Université de Montréal

The Shifting Landscape of Data: Learning to Tame Distributional Shifts

par

Adam Ibrahim

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Discipline

May 3, 2024

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

The Shifting Landscape of Data: Learning to Tame Distributional Shifts

présentée par

Adam Ibrahim

a été évaluée par un jury composé des personnes suivantes :

Simon Lacoste-Julien
(président-rapporteur)

Ioannis Mitliagkas
(directeur de recherche)

Irina Rish
(codirecteur)

Guillaume Rabusseau
(membre du jury)

Razvan Pascanu
(examinateur externe)

(représentant du doyen de la FESP)

Résumé

Les modèles d'apprentissage automatique (ML) atteignent des performances remarquables sur les tâches pour lesquelles ils sont entraînés. Cependant, ils sont souvent sensibles aux changements dans la distribution des données, ce qui peut nuir à leur fiabilité. Cela peut se produire lorsque la distribution des données rencontrées au déploiement diffère de celle vue pendant l'entraînement, entraînant une dégradation considérable des performances. Pire encore, les attaquants peuvent également induire de tels changements afin d'induire les modèles d'apprentissage automatique en erreur. Enfin, cela peut même arriver si l'entraînement est effectué séquentiellement sur des distributions de données différentes. Ces *changements de distribution* sont omniprésents en ML, nuisant à l'équité, à la fiabilité, à la sécurité et à l'efficacité des modèles d'apprentissage automatique. Cette thèse se concentre sur la compréhension et l'amélioration de la robustesse et de l'adaptation des modèles de ML aux changements de distribution, englobant à la fois des travaux théoriques et expérimentaux.

Tout d'abord, nous étudions les limites fondamentales de l'optimisation différentiable à plusieurs objectifs. Une meilleure compréhension de ces limites est importante car les travaux sur les changements de distribution reposent souvent sur des formulations de la théorie des jeux. Nous fournissons de nouvelles bornes inférieures sur la vitesse de convergence d'une large classe de méthodes, ainsi que de nouvelles métriques de conditionnement qui aident à évaluer la difficulté d'optimiser des classes de jeux, et expliquent le potentiel de convergence rapide, même sans forte convexité ou forte concavité.

Deuxièmement, nous abordons le manque de robustesse aux attaques adversarielles contre plusieurs types d'attaques, une limitation courante des méthodes de pointe. Nous proposons une approche inspirée de la généralisation de domaine, utilisant l'extrapolation des risques (REx) pour promouvoir la robustesse à plusieurs attaques. Notre méthode atteint des performances supérieures aux bases de référence existantes, que les attaques aient été vues ou non lors de l'entraînement.

Enfin, nous nous intéressons aux défis du pré-entraînement continu pour les grands modèles de langage (LLM). Ces modèles sont confrontés à un compromis: soit ils oublient de manière catastrophique les connaissances antérieures lorsqu'ils sont mis à jour sur de nouvelles données, soit ils nécessitent un réentraînement complet coûteux en calcul. Nous démontrons qu'une combinaison de réchauffement et de re-décroissance du taux d'apprentissage, et de réutilisation des données précédemment utilisées permet aux LLM d'apprendre continuellement à partir de nouvelles distributions tout en préservant leurs performances sur les données auparavant apprises. Cette approche

permet d'atteindre les performances d'un réentraînement complet, mais à une fraction du coût en calcul.

Dans l'ensemble, cette thèse apporte des considérations importantes pour améliorer la robustesse et l'adaptation aux changements de distribution. Ces contributions ouvrent des voies prometteuses pour relever les défis du ML du monde réel dans l'optimisation multiobjectif, la défense contre les adversaires et l'apprentissage continu des grands modèles de langage.

Mots-clés: intelligence artificielle, apprentissage automatique, changements de distribution, attaques adversarielles, apprentissage continu, grand modèle de langage, LLM, optimisation, théorie des jeux.

Abstract

Machine learning (ML) models achieve remarkable performance on tasks they are trained for. However, they often are sensitive to shifts in the data distribution, which may lead to unexpected behaviour. This can happen when the data distribution encountered during deployment differs from that used for training, leading to considerable degradation of performance. Worse, attackers may also induce such shifts to fool machine learning models. Finally, this can even happen when training sequentially on different data distributions. These *distributional shifts* are pervasive in ML, hindering the fairness, reliability, safety and efficiency of machine learning models. This thesis is focused on understanding and improving the robustness and adaptation of ML models to distributional shifts, encompassing both theoretical and experimental work.

First, we investigate the fundamental limits of differentiable multiobjective optimisation. This investigation is important because works on distributional shifts often rely on game theoretical formulations. We provide new lower bounds on the speed of convergence of a large class of methods, along with novel condition numbers that help assess the difficulty to optimise classes of games, and explain the potential for fast convergence even without strong convexity or strong concavity.

Second, we address the lack of adversarial robustness against multiple attack types, a common limitation of state-of-the-art methods. We propose a domain generalisation-inspired approach, using Risk Extrapolation (REx) to promote robustness across a range of attacks. Our method achieves performance superior to existing baselines for both seen and novel types of attacks.

Finally, we tackle the challenges of continual pretraining for large language models (LLMs). These models face a trade-off: either they catastrophically forget previous knowledge when updated on new data, or they require computationally expensive full retraining. We demonstrate that a combination of learning rate re-warming, re-decaying, and the replay of previous data allows LLMs to continually learn from new distributions while preserving past knowledge. This approach matches the performance of full retraining, but at a fraction of the computational cost.

Overall, this thesis contributes impactful considerations towards improving robustness and adaptation to distributional shifts. These contributions open promising avenues for addressing real-world ML challenges across multiobjective optimisation, adversarial defense, and continual learning of large language models.

Keywords: artificial intelligence, machine learning, distributional shift, adversarial attacks, continual learning, large language model, LLM, optimisation, game theory.

Contents

List of Tables

List of Figures

List of acronyms and abbreviations

AA	AutoAttack
AI	artificial intelligence
AT	adversarial training
Avg	Average [of attacks]
B	billion
CL	continual learning
CW	Carlini Wagner
DF	DeepFool
DACPT	domain adaptive continual pre-training
DNN	deep neural networks

DRO	distributionally robust optimisation
ERM	emperical risk minimization
FGSM	fast gradient sign method
GD	gradient descent
GDA	gradient descent-ascent
Ger	German
IID	independent and identically distributed
IRM	invariant risk minimization
LLM	large language model
LR	learning rate
ML	machine learning
MLM	masked language modelling
MLP	multilayer perceptron

MSD	multi-steepest descent
OoD	out-of-distribution
PGD	projected gradient descent
REx	risk extrapolation
SCLI	stationary canonical linear iterative
SLE	system of linear equations
SGD	stochastic gradient descent
SOTA	state of the art
SP	SlimPajama
T	trillion
WU	warm-up

Acknowledgment

First, I would like to thank Ioannis Mitliagkas for his supervision during my PhD. Ioannis provided great mentorship and feedback throughout the whole PhD, helping me improve my research skills, and in particular, the presentation of my research. Ioannis was encouraging during my thesis and supported me as I explored various areas of machine learning. I also would like to highlight that Ioannis encouraged me to get involved in mentoring MSc students which eventually led to me consulting as an external supervisor in the industry.

I would also like to thank Irina Rish for her trust in having me coorganise workshops with her, assisting with grant proposals, agreeing to cosupervise me in the last year of my PhD, and always being open to discussion. I would also like to highlight my gratitude for the unprecedented, unique opportunity to gain experience with LLM pretraining thanks to her hard work and vision in realising very early the potential of scaling, and obtaining compute grants to study large models.

I would like to express my gratitude to all the colleagues I met while mentoring and supervising projects in the industry: names are kept confidential at Apple, but you know who you are! Maysam Mokarian and Pretesh Patel at Microsoft, Allen Bao, Saurabh Bodhe and Vinay Pandit at AMD, Jean-Sébastien Grondin, Megha Roshan and Arthur Boschet at Optina Diagnostics, Aman Dalmia and Senthilkumar Chandramohan at Staples, Yi Cong Li and Stefan Schmid at Bosch. Beyond these, I would like to also thank Robert Rizk from Blackbox AI, and the whole team at Zyphra with whom I have been having a lot of fun brainstorming and working on the whole pipeline from theory, to data selection, optimisation, kernels, etc.

I would also like to extend my heartfelt thanks to the many people in the industry who have been supportive and encouraging, including Amy Zhang, Behnam Neyshabur, David Stutz, Helen King, Jeffrey Dean, Nicholas Carlini.

In particular, special thanks to: Michal Valko for the fun times, camaraderie and advice; Sven Gowal for being very encouraging and enthusiastic to collaborate; and Arthur Mensch for sharing how much fun he was having working on LLMs in 2022, convincing me to start working on large models.

I would like to extend my appreciation to the great people I have met, collaborated with, or talked to for hours at Mila: Aaron Courville, Alexia Jolicoeur-Martineau, Benjamin Therien, Brady Neal, Charles Guille-Escuret, Damien Scieur, David Krueger, Divyat Mahajan, Dongyan Lin,

Evgenii Nikishin, Gauthier Gidel, Gwendolyne Legate, Hailey Schoelkopf, Hiroki Naganuma, Karina Wolf, Kartik Ahuja, Kilian Fatras, Kshitij Gupta, Mahta Ramezanian, Manuela Girotti, Mathilde Besson, Mats Richter, Mattie Tesfaldet, Maude Couture, Max Schwarzer, Mido Assran, Mehrnaz Mofakhami, Mohammad Pezeshki, Motahareh Sohrabi, Nicolas Loizou, Olexa Bilaniuk, Pouya Bashivan, Quentin Anthony, Quentin Bertrand, Rachade Hmamouchi, Reyhane Askari, Reza Bayat, Ryan D'Orazio, Samuel Lavoie, Simon Guiroy, Waïss Azizian, Zichu Liu.

I am indebted to all my dear friends who supported me, many of whom since middle school, and in spite of time differences – among whom Etienne and Arnaud need to be singled out for the countless times they lost sleep so we could spend time together, and Daniel for ensuring I kept a healthy lifestyle since my BSc. And of course, my immense gratitude to my parents Assia and Hussein, my sister Nada, her husband Tarek, and my brother Ahmed, who kept supporting me from beyond the seas for the past 12 years.

Chapter 1

Introduction

Machine Learning (ML) has been successfully applied to many areas, ranging from image recognition, to natural language processing, healthcare, etc. At the heart of this success is the ability of Deep Learning (DL) models to learn complex patterns from data. However, current ML methods still are sensitive to distributional shifts – changes in the data distribution.

In the context of generalisation, such distributional shifts may occur after training and deployment of the model due to biases in the training dataset or the data seen after deployment for example. These distributional shifts might induce significant losses of performance (Quionero-Candela et al., 2009). The fields of domain adaptation and domain generalisation have focused on addressing these challenges, in order to improve the ability of models to generalise – that is, perform well – in spite of distributional shifts. Intriguingly, such distributional shifts may be induced intentionally in order to manipulate the predictions of models, which poses a security risk. Adversarial machine learning is concerned with studying such vulnerabilities and addressing them.

Domain adaptation, domain generalisation and adversarial machine learning often use game formulations. The optimisation of games, however, is more complex and is less understood than single-objective optimisation. For example, even on classes of problems as simple as quadratic games, it is not well understood how much methods can be accelerated, or how hard optimising the problem is. Thus, improving the understanding of games may help develop better tools to address distributional shifts.

Beyond the generalisation setting, distributional shifts can also occur during the learning process itself. This is frequently the case in reinforcement learning (Igl et al., 2020), and is central to the field of continual learning. Continual learning is important to the development of efficient machine learning methods that may be able to train on new distributions in order to adapt to shifted distributions. However, training on non-stationary data distributions introduces novel challenges in maintaining performance on old data and learning well on new data. The ability of larger models such as Large Language Models (LLM) to learn from massive datasets (Kaplan et al., 2020) holds significant potential in this domain. Their adaptability to many tasks and scalability may be leveraged to train on large amounts of diverse data to improve robustness against distributional

shifts (Hernandez et al., 2021). Since data gathering is bound to be an ongoing effort, for example, due to temporal shifts (e.g., updating world knowledge), the ability to continually train large models on new distributions may be a promising solution to improving robustness against distributional shifts. However, even in the case of LLMs, continuing to pretrain a model that has already been previously pretrained has proven to be surprisingly difficult, requiring inefficient strategies such as retraining the model from random initialisation on the old and new data.

This thesis aims to delve in several of these aspects of distributional shifts. After giving more background on distributional shifts and the optimisation of games in Chapter 2, we present three papers:

- Chapters 3 and 4 present *Linear Lower Bounds and Conditioning of Differentiable Games* (Ibrahim et al., 2019). This work aims to improve the fundamental understanding of optimising games. Specifically, we provide condition numbers to capture the difficulty of optimising games, and lower bounds to help understand how much algorithms may be accelerated on a given class of games.
- Chapters 5 and 6 present *Towards Out-of-Distribution Adversarial Robustness* (Ibrahim et al., 2022). In this paper, we illustrate how adversarial machine learning can be framed as a distributional shift problem. Then, using this connection with domain generalisation, we explore the potential of using tools for out-of-distribution generalisation to defend against adversarial attacks, obtaining better performance than state-of-the-art methods on the dataset and architectures tests, and against multiple combinations of seen and unseen attacks.
- Chapters **??** and 8 present *Simple and Scalable Strategies to Continually Pre-train Large Language Models* (Ibrahim et al., 2024). In this work, we investigate methods to enable the continual pretraining of Large Language Models (LLM). When new data is released, the common strategy has been to retrain from scratch on the union of old and new data in order to avoid degradation of performance on previously learnt data, or poor adaptation to the new data, at potentially tremendous environmental, financial and timely cost. Specifically, we show how using a combination of learning rate and replay is sufficient to allow continually pretraining an existing on the new data without retraining from scratch in the settings we consider.

Finally, we conclude the thesis in Chapter 9.

Chapter 2

Background

2.1. Distributional shifts at test time

Machine learning (ML) is concerned with algorithms learning from data in order to perform well on certain tasks. These tasks can be very varied: image classification, language generation, decision-making, etc. The most common procedure to *train* a parametric model in deep learning (DL) is to solve an optimisation problem in order to find a suitable set of parameters on a training dataset. This optimisation problem can correspond, for example, to minimising the error in the predictions of the model given labelled training data, with cross-entropy being a common choice of a loss function. The intuition, supported by theory such as Probably Approximately Correct learning (Valiant, 1984), is that if a model performs well enough on its training dataset, it should also perform approximately as well on any data seen at test time – that is, after having been trained – provided that the distributions of the data seen during training and at test time are similar. The ability of a model to *generalise* to new data after having been trained is key to the deployment of machine learning algorithms. Because a common practice has been to split the same dataset into a training, validation and test datasets, the training and validation datasets used during training often allow predictable performance on the test set. This is because this practice of splitting the same dataset ensures implicitly that the training, validation and test datasets are *identically distributed*.

In practice, however, models often fail to perform well when deployed. This is because the distribution of data encountered at test time is different from the distributions seen during training. For example, images used during the design of an object recognition model may come from the same camera, often have ideal exposure, may have been preprocessed such that the entity to be recognised is always in the center of the image or the background might have been removed, etc. In spite of their name, the test datasets used during the conception of machine learning models often share such characteristics with the training dataset, due to methodological biases or scarcity of available data. This is the case with the aforementioned practice of randomly splitting the same dataset to generate a training and a test set. On the other hand, the data encountered in the real world by a deployed model (that is, at test time) may include for instance obstructed objects or

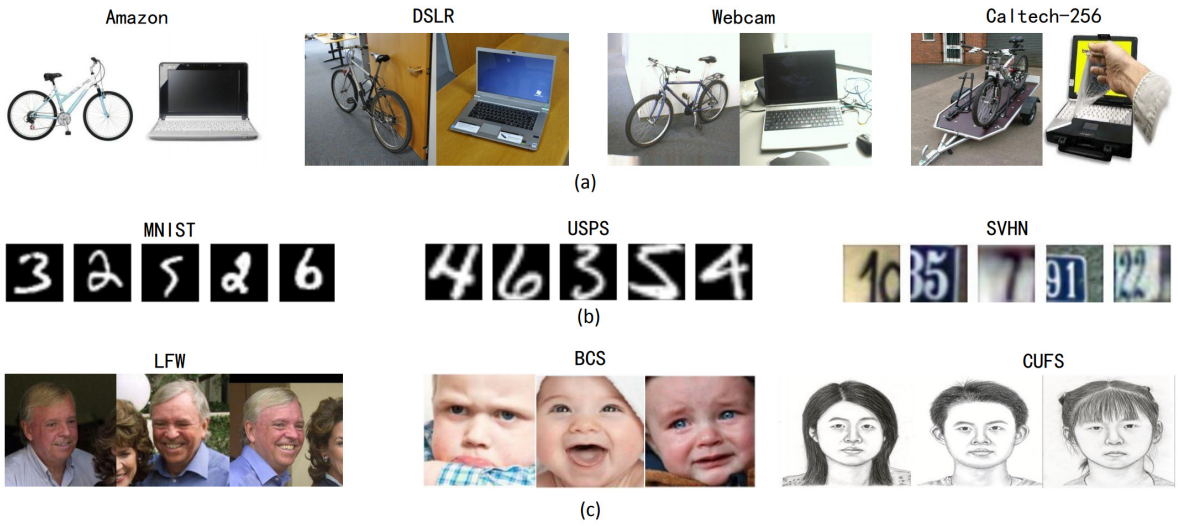


Figure 1. An example of various datasets of images of 1) bicycles and laptops, 2) numbers, 3) faces. In each of these 3 cases, several datasets are provided, illustrating how each dataset encodes particular biases. Source: (Wang and Deng, 2018)

scenes with imperfect lighting conditions. Fig. 1 also highlights how datasets for the same task may vary. Here, as all samples from the LFW dataset are pictures while all samples of CUFS are sketches, it is clear that assumption that samples are identically distributed is not satisfied if one wants to use a model designed from CUFS on LFW. This is because the samples from LFW are *out of the distribution* of sketches of faces. Thus, the performance of the model on the test sets used during the design of the model (e.g. if the model was trained and tested on CUFS during its design) may no longer be representative of its performance on a different distribution of data. However, we want the model to maintain good performance at test time on distributions that may be different from the training distribution. We refer to the ability of the model to maintain its performance on a different distribution as *out-of-distribution (OoD) generalisation*.

For example, in image classification, this may be due to the use a different camera, different lighting conditions, partial obstruction of the object, etc. We refer to changes in the distribution as *distributional shifts*. In this section, we focus on distributional shifts happening at test time; we will discuss in a further section how distributional shifts can also happen during training.

2.1.1. Domain Adaptation

A particular subproblem of OoD generalisation is that of *domain adaptation*. Suppose we have access to labelled training data (this is referred to as the source domain). In domain adaptation, we still want to assess and improve the performance of the model on a different domain (or distribution). However, we assume that the target domain is fixed, and that we have access to

unlabelled (unsupervised) or partially labelled (semi-supervised) data from the target domain during training.

There are several approaches to domain adaptation in the unsupervised case. Common approaches revolve around finding maps between the source and target domains, forcing the distributions of the source and the target to align, ensemble methods, etc (see (Wilson and Cook, 2020) for a survey). In particular, Ganin et al. (2016) present a highly successful methodology based on aligning the distribution of the features from both the source and the target domains, inspired by theoretical work by Ben-David et al. (2007, 2010). The goal is to learn good discriminative features that are invariant across the source and the target domains. To enforce the latter, the authors introduce a domain classifier that attempts to discern from the output of the feature map whether a data sample was from the source or the target domain. In parallel, the feature map feeds into a label classifier that is charged with predicting the correct label given an input. Intuitively, if the feature distributions are invariant across domains, the domain classifier should perform poorly. Therefore, on the one hand, we want to find sets of parameters of both the label and the domain classifiers that minimise their error over the training set. On the other hand, we want to find parameters of the feature map that minimise the loss of the label classifier while maximising the loss of the domain classifier.

Formally, let $G_f(\cdot, \mathbf{w}_f)$ denote the feature map with parameters $\mathbf{w}_f$, $G_y(\cdot, \mathbf{w}_y)$ the label classifier with parameters $\mathbf{w}_y$ and $G_d(\cdot, \mathbf{w}_d)$ the domain classifier with parameters $\mathbf{w}_d$. Consider the augmented dataset

$$U = \{(\boldsymbol{x}_i, 0)\}_{i=1}^{n} \cup \{(\boldsymbol{x}_i, 1)\}_{i=n+1}^{N} \tag{1}$$

where the extra label 0 is appended to the n points of the source domain, and 1 is appended to the $N - n$ points of the target domain. Additionally, for a given datapoint $\boldsymbol{x}_i$, let the loss of the label classifier be $\mathcal{L}_y^i(\mathbf{w}_f, \mathbf{w}_y) = \mathcal{L}_y(G_y(G_f(\boldsymbol{x}_i; \mathbf{w}_f), \mathbf{w}_y), y_i)$, and similarly, let $\mathcal{L}_d^i(\mathbf{w}_f, \mathbf{w}_d) = \mathcal{L}_d(G_d(G_f(\boldsymbol{x}_i; \mathbf{w}_f), \mathbf{w}_d), d_i)$ be the loss of the domain classifier.

We seek to find parameters of the feature map $\mathbf{w}_f$ and the label classifier $\mathbf{w}_d$ such that the empirical risk

$$\hat{\mathcal{R}}(\mathbf{w}_f, \mathbf{w}_y) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y^i(\mathbf{w}_f, \mathbf{w}_y) \tag{2}$$

is minimised. Moreover, drawing from (Ben-David et al., 2007, 2010), we seek parameters of the domain classifier $\mathbf{w}_d$ such that

$$\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_d^i(\mathbf{w}_f, \mathbf{w}_d) + \frac{1}{N-n} \sum_{i=n+1}^{N} \mathcal{L}_d^i(\mathbf{w}_f, \mathbf{w}_d) \tag{3}$$

is minimised[*]. However, we also want to find a feature map such that $\mathbf{w}_f$ maximises this quantity (that is, features that can fool the domain classifier). Since $\arg\min f = \arg\max -f$, this is equivalent to finding $\mathbf{w}_d$ such that

$$F(\mathbf{w}_f, \mathbf{w}_d) = -\left(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{w}_f, \mathbf{w}_d) + \frac{1}{N-n}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{w}_f, \mathbf{w}_d)\right) \tag{4}$$

is maximised (and similarly, we also seek $\mathbf{w}_f$ such that this quantity is minimised).

Since both of these problems involve optimising for $\mathbf{w}_f$, we combine these two objectives into one:

$$E(\mathbf{w}_f, \mathbf{w}_y, \mathbf{w}_d) = \hat{\mathcal{R}}(\mathbf{w}_f, \mathbf{w}_y) + \lambda F(\mathbf{w}_f, \mathbf{w}_d) \tag{5}$$

and the optimisation problem we are solving becomes:

$$\min_{\mathbf{w}_f, \mathbf{w}_y}\max_{\mathbf{w}_d} E(\mathbf{w}_f, \mathbf{w}_y, \mathbf{w}_d) \tag{6}$$

Therefore, we seek to find saddle-points $(\mathbf{w}_f, \mathbf{w}_y, \mathbf{w}_d)$ such that

$$(\mathbf{w}_f^*, \mathbf{w}_y^*) \in \arg\min E(\mathbf{w}_f, \mathbf{w}_y, \mathbf{w}_d^*) \tag{7}$$

$$\mathbf{w}_d^* \in \arg\max E(\mathbf{w}_f^*, \mathbf{w}_y^*, \mathbf{w}_d) \tag{8}$$

This promotes finding a feature map that can yield simultaneously good performance when used in combination with the label classifier, and features that do not differ enough across domains for the domain classifier to be successful, in spite of being trained adversarially on the feature map.

This adversarial approach has since then been improved by several authors (Hoffman et al., 2018; Kumar et al., 2018; Shu et al., 2018; Chen et al., 2019) and expanded to other subfields of machine learning such as active learning (Su et al., 2020), continual learning (Ebrahimi et al., 2020) and transfer learning (Yu et al., 2019), among others.

However, in many cases, we do not have access to the target domain during training; for example, when training an image classification model that will be deployed in the real world. This more general case is the subject of the next subsection.

2.1.2. Domain Generalisation

The general case of dealing with OoD generalisation is often referred to as domain generalisation. As in domain adaptation, domain generalisation is concerned with making the model perform well under distribution shifts between training and test time; however, in domain generalisation, we do not have access in any way to the test distribution.

Several approaches have been proposed to tackle domain generalisation. Some approaches are based on meta-learning (Li et al., 2018; Albuquerque et al., 2019), others on learning invariant

[*]Note that this is close to empirical risk minimisation on U, however this particular weighting allows for the computation of an upper bound on the risk on the target domain (Ben-David et al., 2007, 2010).

features such as Invariant Risk Minimisation (Arjovsky et al., 2019) and its game-theoretic formulation (Ahuja et al., 2020), some use robust optimisation strategies (Krueger et al., 2020), etc. Note that while some of these approaches, notably the meta-learning ones, assume and leverage multiple source domains, some approaches are tailored for a single source domain, for example by adversarially augmenting of the data (Volpi et al., 2018).

For the conciseness of this thesis, we will only briefly discuss Albuquerque et al. (2019)'s approach as an example. The authors propose an approach that assumes multiple source domains, and views those source domains as distributions sampled from a dataset-generating meta-distribution that generates possible, or plausible, datasets. The key idea is that the unseen target domain is a dataset that might be close to the empirical distribution of datasets sampled from the meta-distribution. If that is the case, Albuquerque et al. (2019) show that based on Ben-David et al. (2007, 2010), one may derive guarantees for the error on the target domain. After obtaining an upper bound on the error of the target domain, Albuquerque et al. (2019) propose a methodology very close to the one presented in the previous section. Indeed, they also propose an approach based on a feature map, a label classifier, and one-versus-all domain classifiers for each source domain that assess whether an input belongs to that respective source domain, and the goal is to find a feature map that help the label classifier but hinders the domain classifier.

Formally, let $G_f(\cdot, \mathbf{w}_f)$ denote the feature map with parameters $\mathbf{w}_f$, $G_y(\cdot, \mathbf{w}_y)$ the label classifier with parameters $\mathbf{w}_y$ and $G_d^j(\cdot, \mathbf{w}_d^j)$ the one-versus-all domain classifier with parameters $\mathbf{w}_d^j$ that determines whether an input belongs to source domain j. For a given datapoint $\boldsymbol{x}_i$, let the loss of the label classifier be $\mathcal{L}_y^i(\mathbf{w}_f, \mathbf{w}_y) = \mathcal{L}_y(G_y(G_f(\boldsymbol{x}_i; \mathbf{w}_f), \mathbf{w}_y), y_i)$, and let $\mathcal{L}_d^{i,j}(\mathbf{w}_f, \mathbf{w}_d^j) = \mathcal{L}_d(G_d^j(G_f(\boldsymbol{x}_i; \mathbf{w}_f), \mathbf{w}_d^j), d_i^j)$ be the loss of the domain classifier for source domain j, where $d_i^j = 1$ if $\boldsymbol{x}_i$ belongs to the domain j and 0 otherwise. Suppose there are N_S source domains, and a total of n datapoints across all source domains.

We seek parameters of the feature map $\mathbf{w}_f$ and the label classifier $\mathbf{w}_y$ that minimise the empirical risk

$$\hat{\mathcal{R}}(\mathbf{w}_f, \mathbf{w}_y) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y^i(\mathbf{w}_f, \mathbf{w}_y) \tag{9}$$

Moreover, we want to find parameters for the domain classifier $\mathbf{w}_d^1, ..., \mathbf{w}_d^{N_S}$ such that

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{N_S} \mathcal{L}_d^{i,j}(\mathbf{w}_f, \mathbf{w}_d^j) \tag{10}$$

is maximised, while the feature map parameters $\mathbf{w}_f$ minimise this quantity. As before, one may study the related problem of finding parameters of the domain classifier and the feature map respectively maximising and minimising

$$F(\mathbf{w}_f, \mathbf{w}_d^1, ..., \mathbf{w}_d^{N_S}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{N_S} \mathcal{L}_d^{i,j}(\mathbf{w}_f, \mathbf{w}_d^j) \tag{11}$$

These two problems are combined into one objective

$$E(\mathbf{w}_f, \mathbf{w}_y, \mathbf{w}_d^1, ..., \mathbf{w}_d^{N_S}) = \hat{\mathcal{R}}(\mathbf{w}_f, \mathbf{w}_y) + F(\mathbf{w}_f, \mathbf{w}_d^1, ..., \mathbf{w}_d^{N_S}) \tag{12}$$

and the optimisation problem we are solving becomes:

$$\min_{\mathbf{w}_f, \mathbf{w}_y} \max_{\mathbf{w}_d^1, ..., \mathbf{w}_d^{N_S}} E(\mathbf{w}_f, \mathbf{w}_y, \mathbf{w}_d^1, ..., \mathbf{w}_d^{N_S}) \tag{13}$$

Albuquerque et al. (2019) show that this approach yields state-of-the-art performance on several benchmarks, highlighting the importance of min-max formulations in domain generalisation.

2.1.3. Adversarial Machine Learning

An important concept in machine learning is that of *local smoothness*, which dictates that the predictions $f(\boldsymbol{x})$ of a model should not vary much if the input is perturbed by a small ϵ, that is, $f(\boldsymbol{x}) \approx f(\boldsymbol{x} + \epsilon)$. Assuming neural networks satisfy the local smoothness property, a deep model should be able to generalise beyond the exact data it was trained on, provided the new data is close to the training data.

However, this assumption was observed to be falsified in neural networks for reasonably small perturbations to the input, first by Szegedy et al. (2013), then by Goodfellow et al. (2014b), spanning a whole new research field in the deep learning community. Specifically, it was observed that by adding very small, targeted perturbations to an input image, a classifier would start making wrong predictions with high confidence, in spite of there being no ambiguity for a human classifier, as illustrated in Fig. 2. Goodfellow et al. (2014b) give a generic way to design an adversarial example: the *Fast Gradient Sign Method* (FGSM) computes an adversarially perturbed version of $\boldsymbol{x}_0$ as:

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \epsilon \text{sign} \nabla_{\boldsymbol{x}} \mathcal{L}(\mathbf{w}, \boldsymbol{x}_0, y) \tag{14}$$

where the sign function is applied component-wise, y is the true label of $\boldsymbol{x}_0$, $\mathbf{w}$ are the parameters of the model and where $\mathcal{L}$ denotes the loss of the model. Eq. 14 ensures that the adversarial example $\boldsymbol{x}_1$ is at most ϵ away from the clean datapoint $\boldsymbol{x}_0$ under the distance induced by the max-norm, or L_∞-norm (that is, the absolute value of each component of $\boldsymbol{x}_1 - \boldsymbol{x}_0$ is at most ϵ).

Goodfellow et al. (2014b) show that a particularly efficient way to handle FGSM perturbations, or adversarial attacks, is to use the loss for the FGSM-perturbed input as a regulariser, that is, work with the loss

$$\mathcal{L}_{tot}(\mathbf{w}, \boldsymbol{x}, y) = \alpha \mathcal{L}(\mathbf{w}, \boldsymbol{x}, y) + (1 - \alpha) \mathcal{L}(\mathbf{w}, \boldsymbol{x} + \epsilon \text{sign} \nabla_{\boldsymbol{x}} \mathcal{L}(\mathbf{w}, \boldsymbol{x}, y), y) \tag{15}$$

where α is a hyperparameter set to 0.5 in (Goodfellow et al., 2014b).

As the reader may have noticed, eq. 14 merely consists in taking a step of sign-gradient ascent using the loss function to find an input with a larger value of the loss (that is, the sign of the gradient is used instead of the gradient itself when performing "gradient ascent"). This means that we are perturbing the input to maximise the loss.

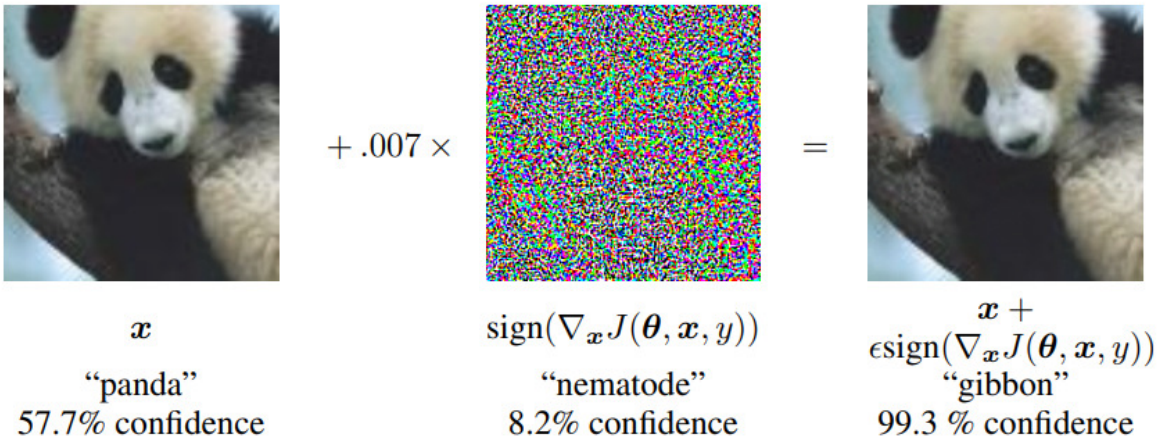


Figure 2. By adding a perturbation imperceptible to the human eye, the model predicts the wrong class with very high confidence. Here, J denotes the objective (loss) function. Source: (Goodfellow et al., 2014b)

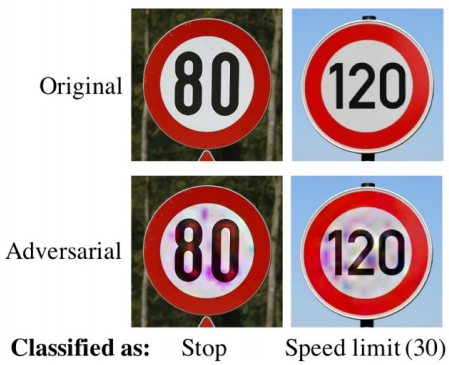


Figure 3. Adversarial attacks on autonomous driving systems by altering traffic signs. Source: Sitawarin et al. (2018)

A concern is that such *adversarial examples* could be induced intentionally by an attacker to induce errors in sensitive applications such as autonomous driving (Sitawarin et al., 2018), as seen in Fig. 3; therefore, it is important to understand *adversarial attacks* in order to be able to design countermeasures. While several other defenses against FGSM attacks have been proposed, such as gradient regularisation (Papernot et al., 2017) or input denoising (Guo et al., 2018), these countermeasures were shown by Athalye et al. (2018) to be ineffective against more advanced attacks.

Indeed, these defenses do not hold against the more advanced *Projected Gradient Descent* (PGD) attacks, which merely consist in taking more iterative steps of sign-gradient ascent to find stronger adversarial attacks. At every step, we project back onto the ball of radius ϵ centered at $\boldsymbol{x}_0$ if we got farther than ϵ from the clean data sample $\boldsymbol{x}_0$ we are perturbing. Formally, PGD adversarial examples are computed as:

$$\boldsymbol{x}^k = \Pi\left(\boldsymbol{x}^{k-1} + \alpha\text{sign}\left(\nabla_{\boldsymbol{x}}\mathcal{L}(\mathbf{w}, \boldsymbol{x}^{k-1}, y)\right)\right) \tag{16}$$

Here, Π denotes the projection operator onto the ϵ-ball centered at $\boldsymbol{x}_0$, and ϵ, α, the norm used to define the ϵ-ball, and the total number of PGD steps are hyperparameters chosen by the attacker. These attacks are deemed to be the strongest first-order attack (Athalye et al., 2018; Wang et al., 2019).

Fortunately, Madry et al. (2018) propose a defense shown to resist well to PGD attacks, and other attacks (Athalye et al., 2018; Guo et al., 2018): *adversarial training*. Adversarial training consists in finding some parameters of the model that minimise the loss of the model on adversarially perturbed

data tailored to the current model. More formally, given a training dataset $\mathcal{S}_{train} = \left(\boldsymbol{x}_i^0, y_i\right)_{i=1,\dots,n}$, adversarial training consists in solving the following problem:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \max_{\left\|\boldsymbol{x}_i - \boldsymbol{x}_i^0\right\|_\infty \leq \epsilon} \mathcal{L}(\mathbf{w}, \boldsymbol{x}_i, y_i) \tag{17}$$

Adversarial training is performed in a repeating sequence. First, we seek the worst adversarial examples at most ϵ away (in the sense of the L_∞-norm) from the original data given the current values of the model parameters $\mathbf{w}$. Then, the model's parameters are updated to improve the performance on those adversarial examples. This replaces the standard training based on empirical risk minimisation, which ignores the inner maximisation problem and uses directly the clean data $\boldsymbol{x}_i^0$ instead of adversarially perturbed data in the loss of eq. 17. In theory, this formulation seeks explicitly to improve the performance on the worst adversarial example within an ϵ distance of each clean training sample. While other norms could be used, the L_∞-norm is traditionally used as the ball of radius ϵ centered at some $\boldsymbol{x}$ induced by the L_∞-norm encompasses all the balls of radius ϵ centered at $\boldsymbol{x}$ induced by L_p norms for any finite p.

In adversarial training, the use of PGD attacks becomes intuitive: Projected Gradient Descent is an optimisation method used to solve constrained minimisation problems. As we mentioned, in the context of adversarial training, finding adversarial examples corresponds to solving the inner constrained maximisation problem, and therefore, using Projected Gradient Descent to generate adversarial examples is well motivated[†].

Remarkably, while this approach of using an adversarial min-max formulation during training has since then been iterated upon, notably by TRADES (Zhang et al., 2019), adversarial training and its variants remain state-of-the-art even against adversarial examples that were generated with different attacks than the ones it was trained on, i.e. FGSM and PGD, as evidenced by Fig. 4. Of note, is the ability of this adversarial training paradigm to lead to models that are more robust even on unseen attacks such as DeepFool (Moosavi-Dezfooli et al., 2016).

However, there are still several open problems in adversarial machine learning: first, unseen attacks still are challenging for defenses (Zhang et al., 2019; Dong et al., 2020; Bashivan et al., 2020). Second, there appears to be a fundamental trade-off between accuracy (defined as performance on a clean test dataset) and robustness (i.e. performance on an adversarially perturbed test dataset) (Tsipras et al., 2018; Zhang et al., 2019; Raghunathan et al., 2020). Third, Cai et al. (2018) and Wang et al. (2019), observe that training early on adversarial examples that are too hard leads to poor performance of the final model, and that curriculum learning (Bengio et al., 2009) in the adversarial training phases helps, where the objective is to use increasingly stronger attacks throughout the adversarial training procedure.

[†]One may note that "projected gradient descent" is a misnomer, as both here and in eq. 16, we are actually performing projected gradient ascent steps.

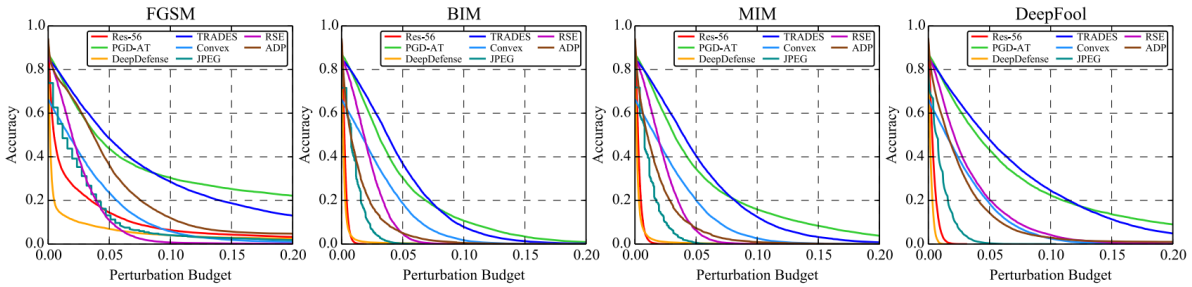


Figure 4. Plot of the robustness (accuracy on adversarial attacks) against various attacks vs perturbation budget ϵ. PGD-AT is standard adversarial training on PGD perturbed data with respect to the L_∞-norm, TRADES is also trained adversarially with respect to the L_∞-norm, Res-56 is a model trained without any kind of defense, DeepDefense is trained on PGD attacks with respect to the 2-norm (Yan et al., 2018), JPEG (Dziugaite et al., 2016), convex (Wong et al., 2018), RSE (Liu et al., 2018) and ADP (Pang et al., 2019) are alternative approaches that do not perform adversarial training. BIM is a variant implementation of PGD, and MIM adds momentum to BIM. Source: (Dong et al., 2020)

We posit that the first two problems are due to the fact that adversarial machine learning *is* in many ways a subproblem of OoD generalisation. Indeed, after training on a particular distribution, e.g. the clean training data, we want the model to perform well on a different distribution at test time. In the case of unseen attacks, this target domain becomes the distribution of adversarial examples from the particular unseen attacks. We also hypothesise that the trade-off between robustness and accuracy comes from the fact that the adversarial and clean distributions are different, and hence that we are just observing the usual pitfalls of machine learning that domain generalisation attempts to address. In the case of curriculum learning, Wang et al. (2019) suspect that the model might be overfitting to noise in strong adversarial attacks in the early phases of training which may prevent the model from learning features that are more generally indicative of the different classes. We believe that this may be true, and that this would require a deeper look into the dynamics of the adversarial training optimisation procedure.

2.2. Optimisation of games

In the previous section, we have seen that min-max formulations are very important components of methods tackling OoD generalisation and its subproblems. As such, it would be good to build a strong understanding of the optimisation of games. It so happens that saddle-point problems and min-max problems are particular kinds of games. Games allow us to model situations where different players are optimising one or several objectives, and where players might be at odds with one another (for example, in min-max problems).

Formally, following the definition of Balduzzi et al. (2018), a *differentiable game* is characterised by n players, each associated with a set of parameters $\mathbf{w}_i \in \mathbb{R}^{d_i}$ and a twice continuously differentiable objective function $l_i : \mathbb{R}^d \to \mathbb{R}$ of all the parameters $\mathbf{w} = (\mathbf{w}_1,...,\mathbf{w}_n) \in \mathbb{R}^d$, where $d = \sum_{i=1}^n d_i$. In particular, if $\sum_{i=1}^n l_i(\mathbf{w}) = 0$, we say that the game is *zero-sum*.

Often, we seek to minimise the objectives l_i, and look for *Nash equilibria* $\mathbf{w}^* = (\mathbf{w}_1^*,...,\mathbf{w}_n^*)$, which satisfy[‡] for all i

$$\mathbf{w}_i^* \in \arg\min_{\mathbf{w}_i} l_i \left(\mathbf{w}_1^*, ..., \mathbf{w}_{i-1}^*, \mathbf{w}_i, \mathbf{w}_{i+1}^*, ..., \mathbf{w}_n^*\right) . \tag{18}$$

In order to find the Nash equilibria, we may look for stationary points, corresponding to the zeros of the vector field $\boldsymbol{v}(\mathbf{w}) = \left(\nabla_{\mathbf{w}_1} l_1(\mathbf{w}) \dots \nabla_{\mathbf{w}_n} l_n(\mathbf{w})\right)^\top$. In single-objective optimisation, which corresponds to a 1-player game, we know that stationary points of $\boldsymbol{v}$ do not necessarily represent minima of the objective function, and higher order information, such as the Hessian, is necessary to determine whether a stationary point is a minimum. The same is true for a game with several players (Balduzzi et al., 2018), where the *Jacobian* of $\boldsymbol{v}$, given by

$$\nabla \boldsymbol{v}(\mathbf{w}) = \begin{pmatrix} \nabla^2_{\mathbf{w}_1} l_1(\mathbf{w}) & \dots & \nabla_{\mathbf{w}_n} \nabla_{\mathbf{w}_1} l_1(\mathbf{w}) \\ \vdots & & \vdots \\ \nabla_{\mathbf{w}_1} \nabla_{\mathbf{w}_n} l_n(\mathbf{w}) & \dots & \nabla^2_{\mathbf{w}_n} l_n(\mathbf{w}) \end{pmatrix} \tag{19}$$

gives sufficient conditions to determine whether a stationary point is a Nash equilibrium.

A good illustration of how min-max problems can be studied as games is to look at quadratic games, which will also be useful when discussing our contributions.

2.2.1. Quadratic games and quadratic min-max problems

2.2.1.1. Quadratic games. In order to gain insight on general games, we focus on quadratic games[§], corresponding to games with quadratic objectives l_i. We will mostly discuss two-player games, where the players respectively control the parameters $\boldsymbol{x} \in \mathbb{R}^{d_1}$ and $\boldsymbol{y} \in \mathbb{R}^{d_2}$. The quadratic objectives take the form

$$\begin{aligned} l_1(\boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{S}_1 \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{M}_{12} \boldsymbol{y} + \boldsymbol{x}^\top \mathbf{b}_1 \\ l_2(\boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{S}_2 \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{M}_{21} \boldsymbol{x} + \boldsymbol{y}^\top \mathbf{b}_2 \end{aligned} \tag{20}$$

[‡]Of course, we could be trying to maximise some players' objectives, but we can without loss of generality work with minima since $\arg\max f = \arg\min(-f)$

[§]Note that quadratic games are inherently relevant; e.g. in reinforcement learning to learn a linear value function from the mean squared projected Bellman error (Du et al., 2017)

with $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ symmetric. In that case the vector field is given by

$$\begin{aligned}\boldsymbol{v}(\boldsymbol{x},\boldsymbol{y}) &= \begin{pmatrix}\boldsymbol{S}_1\boldsymbol{x}+\boldsymbol{M}_{12}\boldsymbol{y}+\mathbf{b}_1\\ \boldsymbol{M}_{21}\boldsymbol{x}+\boldsymbol{S}_2\boldsymbol{y}+\mathbf{b}_2\end{pmatrix}\\ &= \boldsymbol{A}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{pmatrix}+\mathbf{b} \qquad (21)\\ \text{with } \boldsymbol{A} \triangleq \begin{pmatrix}\boldsymbol{S}_1 & \boldsymbol{M}_{12}\\ \boldsymbol{M}_{21} & \boldsymbol{S}_2\end{pmatrix}, &\quad \mathbf{b} \triangleq \begin{pmatrix}\mathbf{b}_1\\ \mathbf{b}_2\end{pmatrix}\end{aligned}$$

where $\boldsymbol{A}$ is the Jacobian of $\boldsymbol{v}$. For n-player quadratic games, the vector field and Jacobian $\boldsymbol{A}$ take the form

$$\begin{aligned}&\boldsymbol{v}(\mathbf{w}) = \boldsymbol{A}\mathbf{w}+\mathbf{b} \qquad (22)\\ &\text{with } \boldsymbol{A} \triangleq \begin{pmatrix}\boldsymbol{S}_1 & \dots & \boldsymbol{M}_{1n}\\ \vdots & & \vdots\\ \boldsymbol{M}_{n1} & \dots & \boldsymbol{S}_n\end{pmatrix}, \quad \mathbf{b} \triangleq \begin{pmatrix}\mathbf{b}_1\\ \vdots\\ \mathbf{b}_n\end{pmatrix}\end{aligned}$$

where the $\boldsymbol{S}_i$ are symmetric. For more details on quadratic games, see Appendix A.

We shall henceforth refer to the Jacobian of the vector field of quadratic n-player games simply as the Jacobian, since our analysis will be solely based on quadratic objectives. Interestingly, the problem of finding $\mathbf{w}$ such that $\boldsymbol{v}(\mathbf{w}) = 0$ consists in solving a system of linear equations (SLE) (Richardson, 1911). In fact, several techniques to precondition systems of linear equation make use of casting the SLE as a game and optimising it with proximal methods, such as Benzi and Golub (2004).

2.2.1.2. Min-max of quadratics as 2-player quadratic games. Consider the family $\mathcal{P}$ of min-max problems of the form

$$\begin{aligned}\min_{\boldsymbol{x}\in\mathbb{R}^{d_1}}\max_{\boldsymbol{y}\in\mathbb{R}^{d_2}} \quad & f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{y}+\frac{1}{2}\boldsymbol{x}^\top\boldsymbol{S}_1\boldsymbol{x}-\frac{1}{2}\boldsymbol{y}^\top\boldsymbol{S}_2\boldsymbol{y}\\ &\qquad +\boldsymbol{x}^\top\mathbf{b}_1-\boldsymbol{y}^\top\mathbf{b}_2+c \qquad (\mathcal{P})\\ \text{where} \quad & \sigma(\boldsymbol{M}\boldsymbol{M}^\top),\sigma(\boldsymbol{S}_1),\sigma(\boldsymbol{S}_2)\subseteq[0,+\infty)\end{aligned}$$

with possible constraints, where $\sigma(\boldsymbol{M})$ denotes the spectrum of $\boldsymbol{M}$ and where the dimension need not be finite, e.g. $\boldsymbol{x},\boldsymbol{y}\in\ell_2 \triangleq \{\mathbf{u}\in\mathbb{R}^{\mathbf{N}} \mid \sum_i^\infty \mathbf{u}_i^2<\infty\}$. The optimisation of such a problem is equivalent to finding a pair $(\boldsymbol{x}^*,\boldsymbol{y}^*)$ such that,

$$\boldsymbol{x}^* \in \arg\min f(\boldsymbol{x},\boldsymbol{y}^*) \;\text{ and }\; \boldsymbol{y}^* \in \arg\max f(\boldsymbol{x}^*,\boldsymbol{y}) \qquad (23)$$

Noting that $\arg\max f = \arg\min(-f)$, we get that this optimisation problem is equivalent to a zero-sum 2-player game with objectives $l_1 = -l_2 = f$ (see eq. 18). This problem can be reduced to searching for the Nash equilibria of the 2-player quadratic game with simplified objectives

$l_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{S}_1 \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{y} + \boldsymbol{x}^\top \mathrm{b}_1$ and $l_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{S}_2 \boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{y} + \boldsymbol{y}^\top \mathrm{b}_2$, where the $\boldsymbol{S}_i$ have been symmetrised (see Appendix A for an explanation of the symmetrisation of $\boldsymbol{S}_i$ and why the objectives can be simplified). Eq. 21 yields the vector field

$$\boldsymbol{v}(\boldsymbol{x},\boldsymbol{y}) = \begin{pmatrix} \boldsymbol{S}_1 & \boldsymbol{M} \\ -\boldsymbol{M}^\top & \boldsymbol{S}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} + \begin{pmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \end{pmatrix} \tag{24}$$

Therefore, the pair $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ from eq. 23 exists if and only if the corresponding games with vector field given in eq. 24 admit a Nash equilibrium since $\boldsymbol{S}_1, \boldsymbol{S}_2 \succeq 0$. Note that we could also go from a quadratic game satisfying the above to a min-max formulation.

2.2.2. Optimising games

There are several methods to optimise games. In machine learning, we frequently resort to using gradient-based methods to optimise games. For example, one might use the gradient descent(-ascent) algorithm to optimise min-max problems $\min_{\boldsymbol{x}} \max_{\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y})$. In that case, the update rules of simultaneous gradient descent(-ascent) are given by

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \eta_1 \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}) \tag{25}$$

$$\boldsymbol{y}_t = \boldsymbol{y}_{t-1} + \eta_2 \nabla_{\boldsymbol{y}} f(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}) \tag{26}$$

where η_1, η_2 are positive learning rates.

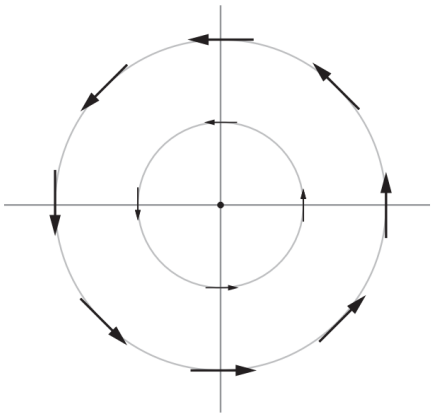


Figure 5. Simultaneous Gradient Descent(-Ascent) can get stuck in cycles. Here, we are optimising $f(x,y) = xy$, where x (resp. y) lies on the x-axis (resp. y-axis), and we observe that the iterates are all on a circle centered at the origin, which is a Nash equilibrium of the game. Source: (Balduzzi et al., 2018)

However, optimising games introduces new challenges. Notably, games have inherent rotational dynamics. Suppose we are optimising a bilinear problem $f(x,y) = xy$. In that case, the Jacobian of the game (see eq. 19) is simply given by

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{27}$$

which has imaginary eigenvalues. This leads to the iterates being stuck in a cycle when optimising with simultaneous gradient descent(-ascent), as in Fig. 5.

Moreover, Balduzzi et al. (2018) observe that rotational dynamics exacerbate challenges with hyperparameter tuning. Indeed, as Fig. 6 illustrates, even when simultaneous gradient descent converges, large step-sizes cause the iterates to diverge from the Nash equilibrium, while the values of the learning rate that allow for convergence lead to very slow progress towards the Nash equilibrium.

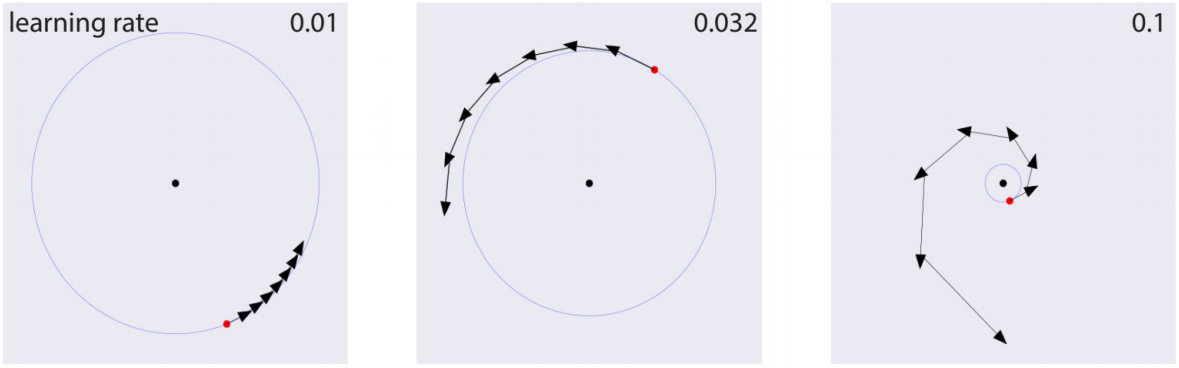


Figure 6. Simultaneous Gradient Descent(-Ascent) requires small step-sizes in order not to diverge because of the rotational dynamics. Source: (Balduzzi et al., 2018)

Some approaches, such as adding negative momentum (Gidel et al., 2019b) or the extragradient method (Korpelevich, 1976; Tseng, 1995; Gidel et al., 2019a) and the Hamiltonian method (Loizou et al., 2020), improve on simultaneous gradient descent(-ascent)'s guarantees and performance on games, and are able to converge linearly on bilinear problems under mild assumptions. However, without even considering general games, it is worth noting that unlike single-objective optimisation where significant progress has been made towards understanding the optimisation dynamics and designing optimal methods for smooth strongly-convex problems (Nesterov, 2004), there are still fundamental questions that are unanswered in the case of games with similarly strong assumptions with respect to each variable. In particular, unlike single-objective optimisation, there was no well-defined metric to assess the difficulty of a game on which linear convergence is possible. Moreover, while several methods and associated upper bounds on their rate of convergence exist, no lower bounds were known for unconstrained smooth strongly-convex-strongly-concave games (a natural generalisation of single-objective optimisation's smooth strongly-convex problems) or bilinear games, which means that there was no way to assess the optimality of the various methods proposed to optimise games on such classes.

Even though the games relevant to OoD generalisation come with much less guarantees, we believe it is important to first address the glaring issues that already exist in the simpler settings where linear convergence is possible. Before discussing how we tackled these issues in our ICML 2020 paper (Ibrahim et al., 2019), we give some background on the last setting of distributional shifts considered in this work: shifts in the training distribution.

2.3. Distributional shifts and continual learning

In section 2.1, we discussed distributional shifts at test time. We also evoked how adversarial training induces shifts during training. It happens that distributional shifts can also occur during training if the training data distributions are *non-stationary*, that is, if they change. This happens in reinforcement learning, and can be emulated in simple supervised classification tasks (Igl et al.,

2020). In the case of adversarial ML, interestingly, the distribution of adversarial attacks is likely non-stationary during adversarial training, as most common attacks use functions of the weights of the model at a given timestep to produce adversarial examples. This is a fortiori the case when using a curriculum like Wang et al. (2019), as a corollary of the fact that different tunings of attacks induce different distributions as illustrated in (Ibrahim et al., 2022), presented in Chapter 6.

Considering distributional shifts during training is relevant because as illustrated in the previous subsections, methods to address distributional shifts at test time often rely on training on multiple domains in order to improve the coverage of data. Hence, it would be ideal to train on as many useful domains as possible, such that the training distribution's support covers as much as possible the support of any distribution encountered at test time. Following the *bitter lesson* argument of Sutton (2019), increasing compute may prove to be a potentially great strategy for more robust and adaptive models. Indeed, scaling model size, dataset size and compute has been shown to be a promising strategy in the case of Large Language Models (LLM) or in vision (Kaplan et al., 2020; Brown et al., 2020; Hernandez et al., 2021; Zhai et al., 2022). However, realistically, it is not always possible to be robust against certain types of shifts without additional training. For example, in language modelling, new knowledge about the world or evolution of the language may need to be learnt. Moreover, it is sometimes desirable to adapt a model to new tasks, e.g. adding new languages and coding abilities in language tasks, adding more classes in classification tasks, etc. While an option would be to retrain the model from scratch on the previous and new data, it would be more efficient to be able to update the model with the new data distribution without incurring such costs, *especially* for large models that may require considerable resources to retrain. In such settings where we consider additional training on different distributions to further improve the test-time performance of a preexisting model, there would be distributional shifts during training.

The field of *continual learning* focuses on learning as the training data undergoes distributional shifts (French, 1999). Such shifts during training bring their own set of challenges. Indeed, two pitfalls encountered in continual learning are that a model may fail to adapt to new data, referred to as a *loss of plasticity* (Ellis and Lambon Ralph, 2000), or may see its performance on previously learnt data or tasks degrade, referred to as *catastrophic forgetting* (McCloskey and Cohen, 1989; Ratcliff, 1990). In fact, the relationship between the plasticity of a model and forgetting has long been studied in the continual learning literature and neuroscience, and referred to as the *plasticity-stability dilemma* (Mermillod et al., 2013a). Some works have attempted to understand the causes of loss of plasticity (Igl et al., 2020; Lyle et al., 2023, 2024). Others have focused on mitigating forgetting by controlling hyperparameters such as the learning rate (Kirkpatrick et al., 2017), architectural choices (Mirzadeh et al., 2021), or replaying past data (Rolnick et al., 2019). Some aspects of continual learning are rather intuitive: for example, a large learning rate may increase adaptation but worsen forgetting, and vice-versa for smaller learning rates. Others are less intuitive: for example, Igl et al. (2020) show how a distributional shift of intermediate strength during training leads to worse adaptation to new data than a stronger or weaker distributional shift. It is worth noting that

inducing distributional shifts during training can be desirable, as is the case in general in curriculum learning (Bengio et al., 2009).

In the interest of avoiding redundancy, we refer readers to Chapter 8 for more discussion of the current state of the continual learning literature.

Chapter 3

Prologue to First Article

3.1. Article Details

Linear Lower Bounds and Conditioning of Differentiable Games. Adam Ibrahim, Waïss Azizian, Gauthier Gidel, Ioannis Mitliagkas. In the 37th International Conference on Machine Learning (ICML 2020).

3.2. Contributions

Ioannis had been working on games (Gidel et al., 2019b), and suggested studying lower bounds to understand limits on accelerating optimisation methods for games as Nesterov did for single-objective optimisation (Nesterov, 2004). Gauthier suggested single-objective optimisation papers to read among which p-SCLI (Arjevani et al., 2016). Adam found and proved the lower bounds and condition numbers, and presented those results in a preliminary version of the work as a project for Ioannis's optimisation class. Waïss reviewed Adam's proofs. Ioannis helped write the introduction, and Adam wrote the rest. Ioannis, Waïss and Gauthier helped review the draft. Ioannis supervised the project.

3.3. Context

Multi-objective optimisation has had a renewal of interest due to adversarial formulations (e.g. generative adversarial networks (Goodfellow et al., 2014a)), by reframing the optimisation problem with game-theoretical formulations and drawing inspiration from physics (Mescheder et al., 2017; Balduzzi et al., 2018). While work had focused on guaranteeing convergence speed of certain algorithms such as extragradient methods (Korpelevich, 1976) or negative momentum (Gidel et al., 2019b), it was not clear whether these guarantees were as accurate as possible. Since those guarantees take the form of upper bounds, complementary lower bounds would allow the research community to know when a method is optimal – that is, when upper bounds and lower

bounds match asymptotically. Moreover, preexisting condition numbers for games (Palaniappan and Bach, 2016) suggested that the absence of strong convexity and strong concavity in min-max games made linear convergence impossible. Works such as (Gidel et al., 2019b) had already shown linear convergence to be possible, highlighting that the existing condition numbers did not properly gauge the difficulty of optimising differentiable games.

Chapter 4

Linear Lower Bounds and Conditioning of Differentiable Games

4.1. Introduction

Game formulations arise commonly in many fields, such as game theory (Harker and Pang, 1990), machine learning (Kim and Boyd, 2008; Goodfellow et al., 2014a), and computer vision (Chambolle and Pock, 2011; Wang et al., 2014) among others, and encompass saddle-point problems (Palaniappan and Bach, 2016; Chambolle and Pock, 2011; Chen et al., 2017).

The machine learning community has been overwhelmingly using gradient-based methods to train differentiable games (Goodfellow et al., 2014a; Salimans et al., 2016). These methods are not designed with game dynamics in mind (Mescheder et al., 2017), and to make matters worse, have been often tuned suboptimally (Gidel et al., 2019b). A recent series of publications in machine learning brings in tools from the minimax and game theory literature to offer better, faster alternatives (Daskalakis et al., 2018; Gidel et al., 2019a,b). This exciting trend begs the question: how fast can we go? Knowing the fundamental limits of this class of problems is critical in steering future algorithmic research.

In order to answer this question, the optimisation literature contains a few different approaches based on the distance between the iterates at a step t and the optimal choice of parameters. Given an optimisation algorithm, it is possible to show under certain assumptions on the objectives that this error is in $\mathcal{O}\left(\rho^t\right)$, where the rate of convergence ρ depends on the algorithm (Nesterov, 2004). If $\rho \in (0, 1)$, we say that the rate of convergence is linear, which corresponds to the error decaying exponentially fast. Hence, a lower bound on the rate of convergence limits how fast an algorithm may converge. This is important as it helps establish the tightness of upper bounds, which happens when they are matched by the lower bounds, and may otherwise indicate possible acceleration of the method considered. For example, in single-objective optimisation, the lower bound on the rate of convergence of first-order black box algorithms is known to be linear for smooth, strongly convex

objectives, and can be derived via a domino-like coverage argument by Nesterov (2004). Another recent, spectral approach by Arjevani et al. (2016) complements these results by proposing linear lower bounds for a large class of optimisers in finite-dimensional settings. As the lower bounds for Nesterov's accelerated gradient obtained by those techniques match the upper bound, we know that Nesterov's accelerated gradient is optimal within a large class of methods for smooth, strongly convex objectives. Additionally, in optimisation, a natural concept of condition number arises to describe the difficulty of μ-strongly convex, L-smooth objectives. This condition number is the only problem-dependent quantity that appears in both the upper and lower bounds and is given by $\kappa = L/\mu$ (Nesterov, 2004). In single-objective optimisation, there is a clear distinction between strongly convex objectives, where the condition number is finite and linear rates are achievable, and general convex objectives where the condition number can be undefined and only sublinear rates are possible in general.

When studying lower bounds for convex-concave min-max, one is faced with a number of distinct challenges compared to the optimisation setting. In particular, there is no universally accepted definition of a condition number. Some commonly used definitions, like the one used in Chambolle and Pock (2011); Palaniappan and Bach (2016), are undefined for bilinear problems, which lack strong convexity and strong concavity in the variables. This is problematic because we know that both extragradient and gradient methods with negative momentum achieve linear convergence in bilinear games (Korpelevich, 1976; Gidel et al., 2019b). *Can we get a condition number that captures the fact that linear rates are possible even in the absence of strong convexity and strong concavity?*

We show that it is possible by providing new lower bounds, obtained by casting saddle-point and min-max problems as games and leveraging existing proof techniques originally designed for smooth strongly convex, single-objective optimisation. These bounds also yield a meaningful condition number for the bilinear case, in the absence of strong convexity and strong concavity in the variables. Our contributions are summarised as follows:

(1) We generalise *Nesterov's domino argument* and design a difficult min-max problem to derive a linear lower bound on the rate of convergence of several first-order black box optimisation algorithms for 2-player games and min-max problems. In order to get an asymptotic rate using the domino bound, one needs to resort to the analysis of infinite-dimensional problems.

(2) We propose a linear lower bound for finite-dimensional problems by generalising the p-SCLI framework proposed by Arjevani et al. (2016) to n-objective optimisation algorithms. This lower bound stems from the spectral properties of the algorithms on quadratics, and is valid for any number of players, and in particular 2-player games and min-max problems. This bound is tight for $n = 1$ since it reduces to the one presented by Arjevani et al. (2016) for strongly convex, smooth single-player optimisation.

(3) We provide a formulation of the condition number of 2-player games consistent with the existing literature on upper bounds for games and min-max problems. In particular, this condition number is finite for bilinear games.

After the results of this work were made available online, several researchers have proposed methods to match some of our bounds in the smooth strongly-convex-strongly-concave setting (Fallah et al., 2020; Lin et al., 2020) and the bilinear setting (Azizian et al., 2020), which is merely convex-concave, thereby establishing the tightness of some of the bounds and the optimality of those methods in those regimes.

The rest of the paper is organised as follows. We purposely discuss preliminaries first in Section 4.2 to introduce the general framework used to present in Section 4.3 the relevant literature in the context of our results. In Section 4.4, we provide lower bounds using Nesterov's domino argument, and in Section 4.5 we improve on those bounds using the spectral technique. We conclude with some discussion.

4.2. Preliminaries

4.2.1. Differentiable games

Following the definition of Balduzzi et al. (2018), a *differentiable game* is characterised by n players, each associated with a set of parameters $\mathbf{w}_i \in \mathbb{R}^{d_i}$ and a twice continuously differentiable objective function $l_i : \mathbb{R}^d \to \mathbb{R}$ of all the parameters $\mathbf{w} = (\mathbf{w}_1,...,\mathbf{w}_n) \in \mathbb{R}^d$, where $d = \sum_{i=1}^n d_i$. In particular, if $\sum_{i=1}^n l_i(\mathbf{w}) = 0$, we say that the game is *zero-sum*.

Often, we seek to minimise the objectives l_i, and look for *Nash equilibria* $\mathbf{w}^* = (\mathbf{w}_1^*,...,\mathbf{w}_n^*)$, which satisfy* for all i

$$\mathbf{w}_i^* \in \arg\min_{\mathbf{w}_i} l_i \left(\mathbf{w}_1^*, ..., \mathbf{w}_{i-1}^*, \mathbf{w}_i, \mathbf{w}_{i+1}^*, ..., \mathbf{w}_n^*\right) . \tag{28}$$

In order to find the Nash equilibria, we may look for stationary points, corresponding to the zeros of the vector field $\boldsymbol{v}(\mathbf{w}) = \left(\nabla_{\mathbf{w}_1} l_1(\mathbf{w}) \dots \nabla_{\mathbf{w}_n} l_n(\mathbf{w})\right)^\top$. In single-objective optimisation, which corresponds to a 1-player game, we know that stationary points of $\boldsymbol{v}$ do not necessarily represent minima of the objective function, and higher order information, such as the Hessian, is necessary to determine whether a stationary point is a minimum. The same is true for a game with several players (Balduzzi et al., 2018), where the *Jacobian* of $\boldsymbol{v}$, given by

$$\nabla \boldsymbol{v}(\mathbf{w}) = \begin{pmatrix} \nabla^2_{\mathbf{w}_1} l_1(\mathbf{w}) & \dots & \nabla_{\mathbf{w}_n} \nabla_{\mathbf{w}_1} l_1(\mathbf{w}) \\ \vdots & & \vdots \\ \nabla_{\mathbf{w}_1} \nabla_{\mathbf{w}_n} l_n(\mathbf{w}) & \dots & \nabla^2_{\mathbf{w}_n} l_n(\mathbf{w}) \end{pmatrix} \tag{29}$$

*Of course, we could be trying to maximise some players' objectives, but we can without loss of generality work with minima since $\arg\max f = \arg\min(-f)$

gives sufficient conditions to determine whether a stationary point is a Nash equilibrium. Note that our lower bound analysis encompasses games with stable stationary points that are not Nash equilibria.

4.2.2. Quadratic games

In order to gain insight on general games, we focus on quadratic games[†], corresponding to games with quadratic objectives l_i. In our analysis, we will mostly discuss two-player games, where the players respectively control the parameters $\boldsymbol{x} \in \mathbb{R}^{d_1}$ and $\boldsymbol{y} \in \mathbb{R}^{d_2}$. The quadratic objectives take the form

$$\begin{aligned} l_1(\boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{S}_1 \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{M}_{12} \boldsymbol{y} + \boldsymbol{x}^\top \mathbf{b}_1 \\ l_2(\boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{S}_2 \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{M}_{21} \boldsymbol{x} + \boldsymbol{y}^\top \mathbf{b}_2 \end{aligned} \tag{30}$$

with $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ symmetric. In that case the vector field is given by

$$\begin{aligned} \boldsymbol{v}(\boldsymbol{x}, \boldsymbol{y}) &= \begin{pmatrix} \boldsymbol{S}_1 \boldsymbol{x} + \boldsymbol{M}_{12} \boldsymbol{y} + \mathbf{b}_1 \\ \boldsymbol{M}_{21} \boldsymbol{x} + \boldsymbol{S}_2 \boldsymbol{y} + \mathbf{b}_2 \end{pmatrix} \\ &= \boldsymbol{A} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} + \mathbf{b} \\ \text{with } \boldsymbol{A} &\triangleq \begin{pmatrix} \boldsymbol{S}_1 & \boldsymbol{M}_{12} \\ \boldsymbol{M}_{21} & \boldsymbol{S}_2 \end{pmatrix}, \quad \mathbf{b} \triangleq \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \end{aligned} \tag{31}$$

where $\boldsymbol{A}$ is the Jacobian of $\boldsymbol{v}$. For n-player quadratic games, the vector field and Jacobian $\boldsymbol{A}$ take the form

$$\begin{aligned} \boldsymbol{v}(\mathbf{w}) &= \boldsymbol{A}\mathbf{w} + \mathbf{b} \\ \text{with } \boldsymbol{A} &\triangleq \begin{pmatrix} \boldsymbol{S}_1 & \dots & \boldsymbol{M}_{1n} \\ \vdots & & \vdots \\ \boldsymbol{M}_{n1} & \dots & \boldsymbol{S}_n \end{pmatrix}, \quad \mathbf{b} \triangleq \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{pmatrix} \end{aligned} \tag{32}$$

where the $\boldsymbol{S}_i$ are symmetric. For more details on quadratic games, see Appendix B.1. Further assumptions on the dimensionality or properties of $\boldsymbol{S}_i$ will be introduced in the rest of the paper as they become relevant (e.g. positive semi-definiteness in the next subsection).

We shall henceforth refer to the Jacobian of the vector field of quadratic n-player games simply as the Jacobian, since our analysis will be solely based on quadratic objectives. Interestingly, the problem of finding $\mathbf{w}$ such that $\boldsymbol{v}(\mathbf{w}) = 0$ consists in solving a system of linear equations (SLE)

[†]Note that quadratic games are inherently relevant; e.g. in reinforcement learning to learn a linear value function from the mean squared projected Bellman error (Du et al., 2017)

(Richardson, 1911). In fact, several techniques to precondition SLE make use of casting the SLE as a game and optimising it with proximal methods, such as Benzi and Golub (2004).

In this paper, we will denote the spectrum of a matrix $\boldsymbol{M}$ by $\sigma(\boldsymbol{M})$, and define the *block spectral bounds* $\mu_1, \mu_2, \mu_{12}, L_1, L_2, L_{12}$ as constants bounding the spectra of the blocks in the Jacobian of eq. 31:

$$\begin{aligned} \mu_1 \leq |\sigma(\boldsymbol{S}_1)| \leq L_1 \qquad \mu_2 \leq |\sigma(\boldsymbol{S}_2)| \leq L_2 \\ \mu_{12}^2 \leq |\sigma(\boldsymbol{M}_{12}\boldsymbol{M}_{12}^\top)| \leq L_{12}^2 \end{aligned} \tag{33}$$

where we assume $\boldsymbol{M}_{12}$ is a wide or square matrix (if it is a tall matrix, $\mu_{12}^2 \leq |\sigma(\boldsymbol{M}_{12}^\top\boldsymbol{M}_{12})| \leq L_{12}^2$ is used to define μ_{12} and L_{12} instead of the last inequality of eq. 33).

4.2.3. Min-max of quadratics as 2-player quadratic games

Consider the family $\mathcal{P}$ of min-max problems of the form

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^{d_1}} \max_{\boldsymbol{y}\in\mathbb{R}^{d_2}} \quad & f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{y} + \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{S}_1\boldsymbol{x} - \frac{1}{2}\boldsymbol{y}^\top\boldsymbol{S}_2\boldsymbol{y} \\ & \qquad + \boldsymbol{x}^\top\mathbf{b}_1 - \boldsymbol{y}^\top\mathbf{b}_2 + c \qquad (\mathcal{P}) \\ \text{where} \quad & \sigma(\boldsymbol{M}\boldsymbol{M}^\top), \sigma(\boldsymbol{S}_1), \sigma(\boldsymbol{S}_2) \subseteq [0, +\infty) \end{aligned}$$

with possible constraints and where the dimension need not be finite, for example, $\boldsymbol{x}, \boldsymbol{y} \in \ell_2 \triangleq \{\mathbf{u} \in \mathbb{R}^{\mathbf{N}} \mid \sum_i^\infty \mathbf{u}_i^2 < \infty\}$. The optimisation of such a problem is equivalent to finding a pair $(\boldsymbol{x}^*,\boldsymbol{y}^*)$ such that,

$$\boldsymbol{x}^* \in \arg\min f(\boldsymbol{x}, \boldsymbol{y}^*) \;\text{ and }\; \boldsymbol{y}^* \in \arg\max f(\boldsymbol{x}^*, \boldsymbol{y}) \tag{34}$$

Noting that $\arg\max f = \arg\min(-f)$, we get that this optimisation problem is equivalent to a zero-sum 2-player game with objectives $l_1 = -l_2 = f$ (see eq. 28). This problem can be reduced to searching for the Nash equilibria of the 2-player quadratic game with simplified objectives $l_{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{S}_1\boldsymbol{x} + \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{y} + \boldsymbol{x}^\top\mathbf{b}_1$ and $l_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^\top\boldsymbol{S}_2\boldsymbol{y} - \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{y} + \boldsymbol{y}^\top\mathbf{b}_2$, where the $\boldsymbol{S}_i$ have been symmetrised (see Appendix B.1 for an explanation of the symmetrisation of $\boldsymbol{S}_i$ and why the objectives can be simplified). Eq. 31 yields the vector field

$$\boldsymbol{v}(\boldsymbol{x}, \boldsymbol{y}) = \begin{pmatrix} \boldsymbol{S}_1 & \boldsymbol{M} \\ -\boldsymbol{M}^\top & \boldsymbol{S}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} + \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \tag{35}$$

Therefore, the pair $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ from eq. 34 exists if and only if the corresponding games with vector field given in eq. 35 admit a Nash equilibrium since $\boldsymbol{S}_1,\boldsymbol{S}_2 \succeq 0$. Note that we could also go from a quadratic game satisfying the above to a min-max formulation. Hence, any lower bound on quadratic games of the form of eq. 35 is a lower bound on min-max problems (in $\mathcal{P}$), and vice versa.

4.3. Background

4.3.1. Existing bounds for 2-player quadratic min-max problems

Some upper bounds on the rate of convergence of certain optimisation algorithms exist for unconstrained problems in $\mathcal{P}$. These upper bounds on the rate of convergence ρ imply that for any problem in $\mathcal{P}$, the iterates will converge to a solution in $\mathcal{O}\left(\rho^t\right)$. For clarity's sake, we reformulate these upper bounds to be consistent with the notation of $\mathcal{P}$ and eq. 33. Letting $\kappa = \frac{L_{12}}{\sqrt{\mu_1\mu_2}}$, Chen and Rockafellar (1997) analyse the forward-backward algorithm, and find a convergence in $\mathcal{O}\left(\left(\sqrt{1-\left(\frac{\min(\mu_1,\mu_2,\mu_{12})}{\max(L_1,L_2,L_{12})}\right)^2}\right)^t\right)$. Chambolle and Pock (2011) give an algorithm for which the convergence is in $\mathcal{O}\left(\left(\sqrt{1-\frac{2}{\kappa+2}}\right)^t\right)$. Palaniappan and Bach (2016) present an accelerated version of the forward-backward algorithm with variance reduction with convergence in $\mathcal{O}\left(\left(1-\frac{1}{1+2\kappa}\right)^t\right)$. Note that asymptotically, the Chambolle-Pock and accelerated Forward-Backward rates match up to a factor of 2 on κ. Finally, Gidel et al. (2019b) give an upper bound in $\mathcal{O}\left(\left(1-\frac{1}{4L_{12}^2/\mu_{12}^2}\right)^t\right)$ on the convergence of alternating gradient descent with negative momentum for non-singular *bilinear games*, i.e. quadratic games satisfying eq. 35 with $\boldsymbol{S}_1 = \boldsymbol{S}_2 = 0$ and non-singular Jacobian.

A key problem with those rates of convergence is that the tightness of the upper bound is not established. Such information is important since it may indicate that the algorithm can be accelerated. Ideally, one would use the rate of convergence of the hardest problem (i.e. slowest convergence) in the class of problems, which would be a tight upper bound. If one can only find a lower bound on the rate of convergence of the hardest problem, then any upper bound on the entire problem class must be greater than that lower bound to avoid a contradiction. This is because the (upper bound on the) rate of convergence for a class of problems must apply to *any* problem in the class, and hence be greater than any lower bound derived on any particular problem within that class. Usually, it is not possible to find the problem with the slowest convergence, so one may have to guess a hard enough problem. If the lower bound on that problem matches the upper bound for the problem class, then we have established that not only this problem is one of the hardest problems in the class, but also that the upper bound is tight. Therefore, it is important to remember that the goal of lower bounds generally is not to apply to every problem within the class, unlike upper bounds, but to help estimate how much the upper bounds on the whole class can be improved given the presence of hard problems in the class.

Unlike upper bounds, relevant lower bounds for first-order methods on saddle-point problems are scarcer in the literature. Nemirovsky (1992) gives a lower bound in $\mathcal{O}(1/t)$ for a limited number of steps. Ouyang and Xu (2018) also leverage Krylov subspace techniques, and show lower bounds in $\mathcal{O}(1/t)$ in the monotone case and $\mathcal{O}(1/t^2)$ in the strongly monotone case, assuming the number of iterations is less than half the dimension of the parameters. Note that Ouyang and Xu (2018) do

not assume smoothness of the objective in $\boldsymbol{y}$. A key issue is that since these bounds are only valid for a limited number of steps, they do not yield bounds that can be compared with the upper bounds previously mentioned. In contrast, the lower bounds presented in this work are valid for any number of steps and are linear, and therefore provide a direct limit to the acceleration of methods achieving linear convergence on two-player games. Additionally, our lower bounds also yield condition numbers that give intuition about the difficulty inherent to a problem, and can be computed in a plug-and-play fashion using either bounds on the spectrum of the full Jacobian, or on the spectra of its blocks.

4.3.2. Lower bound techniques for convex optimisation with bounded spectrum

In single-objective optimisation, i.e. a 1-player game, the Jacobian in eq. 29 reduces to the Hessian of the objective, denoted $\boldsymbol{H}(\boldsymbol{x})$. In that case, if there exists $\mu, L \in \mathbb{R}^{++}$ such that for all $\boldsymbol{x}$ in the domain considered

$$\mu \preceq \boldsymbol{H}(\boldsymbol{x}) \preceq L$$

the objective is μ-strongly convex and has L-Lipschitz gradients, and the convergence rates (i.e. upper bounds) are known to be linear in the number of iterations for several classes of algorithms (Nemirovsky and Yudin, 1983; Nesterov, 2004). In the context of convex minimisation, various lower bounds have been derived depending on whether the objective is strongly convex and/or has Lipschitz gradients (see (Bubeck et al., 2015) for an overview).

Nesterov's lower bound In particular, Nesterov (2004) gives an information-based complexity bound for μ-strongly convex objectives with L-Lipschitz gradients, by showing that there is a μ-strongly convex example in $\ell_2 \to \mathbb{R}$ with L-Lipschitz gradients for which first-order black box methods, i.e. methods using only past iterates and gradients of past iterates at every update, converge linearly at a rate at least $\rho = 1 - \frac{2}{\sqrt{\kappa}+1}$, where the condition number is given by $\kappa = L/\mu$. The proof relies on the fact that at iteration t, only the t first components of the estimates $\boldsymbol{x}_t$ have been updated from their initial values, where $\boldsymbol{x}_0 = 0$. This is then used to lower bound the distance to the optimum. Since an infinite number of iterations is required to converge in ℓ_2 if the solution $\boldsymbol{x}^*$ has an infinite number of nonzero components, we obtain asymptotic rates. An important caveat is that an infinite-dimensional example does not directly yield a lower bound for finite-dimensional problems.

p-SCLI Arjevani et al. (2016) introduce the p-SCLI framework to provide bounds for a large class of methods used for optimising μ-strongly convex objectives with L-Lipschitz gradients. Roughly speaking, an algorithm is p-SCLI if its update rule on quadratics $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}^\top \mathbf{b}$ with $\boldsymbol{A}$ symmetric is a linear combination of the p previous iterates and $\mathbf{b}$, where the coefficients are matrices that depend on $\boldsymbol{A}$ and are assumed to be simultaneously triangularisable. The spectral

properties of the update rule are used to derive lower bounds on the rate of convergence of p-SCLI algorithms. The lower bound on the rate of convergence of p-SCLI methods is given by $\rho = 1 - \frac{2}{\sqrt[p]{\kappa}+1}$ for $\kappa = L/\mu$. This allows us to recover lower bounds for gradient descent ($p = 1$) or Nesterov (1983)'s accelerated gradient descent ($p = 2$) that match the upper bounds. A key advantage is that these bounds are more refined than Nesterov's by introducing the dependence on p (for example, both GD and Nesterov's accelerated gradient descent are black box first-order methods, but this bound is tighter for $p = 1$ methods such as GD), and do not rely on an infinite-dimensional example, but rather on the spectral properties of the methods. Such bounds highlight how lower bounds can suggest potential acceleration: in the example previously given, the addition of momentum to gradient descent turns the method from $p = 1$ to $p = 2$, thereby decreasing the lower bound and allowing for potentially faster convergence. Finally, the p-SCLI framework also yields upper bounds, and the authors also show a general method to accelerate algorithms on quadratics with the hope that the acceleration is relevant to more general classes of objectives, albeit at a cost too prohibitive to be practical.

Interestingly, both techniques produce a tight lower bound from a quadratic objective, indicating that quadratics are asymptotically as hard as any other strongly convex, smooth problem in single-objective optimisation. This motivates the use of quadratic games to derive the lower bounds presented in this paper as we generalise those methods to the multi-objective setting. When there are several players, however, the Jacobian is no longer symmetric, and its spectrum will generally be complex, and hence several of the arguments used in single-objective optimisation fail to apply directly.

4.4. Parametric Lower Bounds from Nesterov's Domino Argument

In this section, we will only discuss min-max problems. The class $\mathcal{F}$ of counterexamples considered is

$$\min_{\boldsymbol{x}\in\ell_2}\max_{\boldsymbol{y}\in\ell_2} \quad f(\boldsymbol{x},\boldsymbol{y}) = c\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{y} - d_1\boldsymbol{x}^\top \boldsymbol{e_1} + d_2\boldsymbol{y}^\top \boldsymbol{e_1} + \frac{\mu_1}{2}\|\boldsymbol{x}\|^2 - \frac{\mu_2}{2}\|\boldsymbol{y}\|^2 \tag{$\mathcal{F}$}$$

where $\boldsymbol{M}$ is an infinite-dimensional bidiagonal matrix i.e. $\forall i, M_{ii} = a_0$ and $M_{i,i+1} = a_1$ with all other entries set to 0, such that $ca_0a_1 \neq 0$ and $\mu_1, \mu_2 \in \mathbb{R}^{++}$. Since these problems are in $\mathcal{P}$, the lower bounds of this section are in particular bounds on the optimisation of min-max problems.

Definition 1 (Two-step linear span assumption)**.** *A first-order black box method for 2-player games satisfies the two-step linear span assumption on $\mathcal{F}$ if for problems in $\mathcal{F}$ with Jacobian $\boldsymbol{A}$ (cf eq. 32):*

$$\mathbf{w}_t \in \mathbf{w}_0 + Span\Big(\mathbf{w}_0, ..., \mathbf{w}_{t-1}, \boldsymbol{A}\mathbf{w}_0, ..., \boldsymbol{A}\mathbf{w}_{t-1}, \boldsymbol{A}^2\mathbf{w}_0, ..., \boldsymbol{A}^2\mathbf{w}_{t-1}, \mathbf{b}, \boldsymbol{A}\mathbf{b}\Big) \tag{36}$$

Examples of such methods include simultaneous gradient descent, negative momentum and extragradient. One way to design challenging problems for these methods is to construct problems with a dense solution $(\boldsymbol{x}^*,\boldsymbol{y}^*)$ for which only one new component of the iterates may change from its initial value at every iteration (Nesterov, 2004), a phenomenon we will refer to as the *domino argument* (see Appendix B.2.1 for some intuition, where we show that the argument also applies to cases where diagonal matrices are used as coefficients in the span, and to alternating implementations of any algorithm satisfying the two-step linear span assumption on $\mathcal{F}$ thanks to the properties of bidiagonal Toeplitz matrices).

4.4.1. A first lower bound for games with block spectral bounds μ_1, μ_2, L_{12}

Proposition 1 (Naive bound). *For any problem class containing quadratic games, there exists a function $f : \ell_2 \times \ell_2 \to \mathbb{R}$ corresponding to a problem in $\mathcal{P}$ with block spectral bounds $\mu_1 = L_1, \mu_2 = L_2, L_{12} \in \mathbb{R}^{++}$ as defined in eq. 33, that has condition number $\kappa = \frac{L_{12}}{\sqrt{\mu_1\mu_2}}$ such that for any number of iterations $t \geq 1$ and any procedure satisfying the two-step linear span assumption (see def. 1), the following lower bound holds:*

$$\|(\boldsymbol{x}_t, \boldsymbol{y}_t) - (\boldsymbol{x}^*, \boldsymbol{y}^*)\| \geq \left(1 - \frac{2}{\sqrt{\kappa^2 + 1} + 1}\right)^{t+1} \cdot \|(\boldsymbol{x}_0, \boldsymbol{y}_0) - (\boldsymbol{x}^*, \boldsymbol{y}^*)\| \tag{37}$$

We invite the reader to consult Appendix B.2.2 for the proof. This lower bound on the distance to a solution also yields a lower bound on the minimum number of steps necessary for all subsequent iterates to be within a target distance — typically referred to as *iteration complexity*. We may interpret the proposition as being a bound for 2-player games with block spectral bounds μ_1, μ_2, and L_{12} as the bound provided holds for a problem sharing the same block spectral bounds. Similarly, we also get a bound on problems with block spectral bounds L_1, L_2 and L_{12} by replacing μ_i by L_i appropriately in κ.

We appear to obtain the same condition number as in the upper bound literature. If we assume this bound and condition number to be representative of a finite-dimensional bound as was the case in convex optimisation, we easily see an apparent contradiction from the upper bound on the rate of convergence of alternating gradient descent with negative momentum for bilinear games given by Gidel et al. (2019b). Indeed, if we let $\mu_1, \mu_2 \to 0$, the rate of convergence in Prop. 1 goes to 1, whereas the upper bound of negative momentum is not affected and may indicate fast

convergence. This illustrates how the condition number of the upper bounds is not general enough to be representative of inherent difficulty: it can be shown that for the problem used in the proof of the proposition, $\mu_{12} = 0$. As such, it is not surprising that the bound failed to hold against the upper bound of negative momentum on bilinear games; they were not comparable as by definition bilinear games have $\mu_{12} > 0$. This shows that μ_{12} encodes critical information that this condition number was not able to capture. Nevertheless, an important point is that the bound itself is correct and represents a problem with slow convergence; it just fails to yield a condition number that accurately captures difficulty as μ_{12} does not appear.

However, by refining our proof technique, we can derive a bound which avoids this issue, and yields tighter bounds for games for which we know $\mu_{12}, L_{12}, \mu_1, \mu_2$.

4.4.2. Improved lower bound for games with block spectral bounds $\mu_1, \mu_2, \mu_{12}, L_{12}$

Theorem 1. *For any problem class containing quadratic games, there exists a function $f : \ell_2 \times \ell_2 \to \mathbb{R}$ corresponding to a problem in $\mathcal{P}$ with block spectral bounds $\mu_1 = L_1, \mu_2 = L_2, L_{12} \in \mathbb{R}^{++}, \mu_{12} \in \mathbb{R}^{+}$ as defined in eq. 33, that has condition number $\kappa = \sqrt{\frac{L_{12}^2+\mu_1\mu_2}{\mu_{12}^2+\mu_1\mu_2}}$, such that for any number of iterations $t \geq 1$ and any procedure satisfying the two-step linear span assumption, the following lower bound holds:*

$$\|(\boldsymbol{x}_t, \boldsymbol{y}_t) - (\boldsymbol{x}^*, \boldsymbol{y}^*)\| \geq \left(1 - \frac{2}{\kappa + 1}\right)^{t+1} \cdot \|(\boldsymbol{x}_0, \boldsymbol{y}_0) - (\boldsymbol{x}^*, \boldsymbol{y}^*)\| \tag{38}$$

The same result holds for any problem class containing bilinear games, if one sets $\mu_1 = L_1 = \mu_2 = L_2 = 0$.

Corollary 2 (Iteration complexity bound). *For the same problem classes and under the same assumptions as Theorem 1, the minimal number of steps t required to reach a target distance ϵ from the solution, that is $\|(\boldsymbol{x}_t, \boldsymbol{y}_t) - (\boldsymbol{x}^*, \boldsymbol{y}^*)\| < \epsilon$, is given by*

$$t \geq \frac{\kappa - 1}{2} \log\left(\frac{\|(\boldsymbol{x}_0, \boldsymbol{y}_0) - (\boldsymbol{x}^*, \boldsymbol{y}^*)\|}{\epsilon}\right) - 1 \tag{39}$$

This generalises Prop. 1. The proof can be found in Appendix B.2.3, where we also show how we used spectral properties of Toeplitz matrices in Banach algebras to create the hard problems yielding the bound. The proof of the iteration complexity bound can be found in Appendix B.4. As with the lower bounds, note that the iteration complexity bound does not necessarily hold for every problem in the class; the idea is that to reach a target error ϵ we know that there is at least one problem that requires at least the number of steps indicated in the bound, and therefore that to optimise over a problem class that satisfies the assumptions, we will need in general at least the number of steps given in the bound, to account for the hard problems.

It is important to emphasize that as is the case with Nesterov's argument for single-objective optimisation, this bound is still based on an infinite-dimensional problem, and that upper bounds generally are proven for finite-dimensional settings. We may hope that this bound also holds in finite dimension, since we are not aware of upper bounds contradicting it, and were not able to generate finite-dimensional 2-player games for which the bound did not hold empirically. The condition number appearing in Thm. 1 is more expressive than the one found in the upper bound literature $\kappa = L_{12}/\sqrt{\mu_1\mu_2}$, and is lower bounded by 1, instead of 0. A limitation, however, is that this κ is not able to dissociate $L_1 \neq \mu_1, L_2 \neq \mu_2$, which is a problem in terms of expressivity of the condition number. It may also threaten the tightness of the lower bound since intuition from convex optimisation would suggest that objectives with matching lower and upper bounds on the spectra are easier to optimise. This stems from the fact that the closed form solution for problems in $\mathcal{P}$ when $\boldsymbol{S}_i$ is non-scalar, which we would need for all block spectral bounds to appear in κ, is complicated and the associated condition number is impractical. Therefore, we leave the matter of deriving a practical bound based on the domino argument involving all μ_i and L_i as future work.

Interestingly, the rate takes the same form as in strongly convex smooth optimisation, suggesting that for general n-player games, we may still get a lower bound of the form $\rho \geq 1 - \frac{2}{\kappa+1}$ for some generalised condition number κ. This intuition will be highlighted in the next section, by deriving lower bounds from the spectral properties of the update operators of a large class of optimisation methods for n-player games. The results we are about to introduce will also address the matter of $L_i \neq \mu_i$, and will be based on finite-dimensional problems.

4.5. p-SCLI-n for n-player Games

4.5.1. Definitions and examples

Let $\mathcal{Q}^{d_1,\dots,d_n}$ denote the set of n-player quadratic games, i.e. games comprised of n quadratic objectives $l_i : \mathbb{R}^{d_i} \to \mathbb{R}$, and $f_{\boldsymbol{A},\mathbf{b}}(\mathbf{w}) \in \mathcal{Q}^{d_1,\dots,d_n}$ be a game with vector field $\boldsymbol{A}\mathbf{w} + \mathbf{b}$ as indicated in eq. 32. The following definition is a direct generalisation of the definition of p-SCLI algorithms given by Arjevani et al. (2016) to n-player games.

Definition 2 (p-SCLI-n optimisation algorithms for n-player games)**.** *Let $\mathcal{A}$ be an optimisation algorithm for n-player quadratic games. Then $\mathcal{A}$ is a p-stationary canonical linear iterative method for n-player games (p-SCLI-n) if there exist functions $C_0, ..., C_{p-1}, N$ from $\mathbb{R}^{d\times d}$ to $\mathbb{R}^{d\times d}$-valued random variables, such that the following conditions are satisfied for all $f_{\boldsymbol{A},\mathbf{b}}(\mathbf{w}) \in \mathcal{Q}^{d_1,\dots,d_n}$:*

(1) Given an initialisation $\mathbf{w}^0, \dots, \mathbf{w}^{p-1} \in \mathbb{R}^d$, the update rule at iteration $t \geq p$ is given by

$$\mathbf{w}^t = \sum_{i=0}^{p-1} C_i(\boldsymbol{A})\mathbf{w}^{t-p+i} + N(\boldsymbol{A})\mathbf{b} \tag{40}$$

(2) $C_0(\boldsymbol{A}), ..., C_{p-1}(\boldsymbol{A}), N(\boldsymbol{A})$ are independent from previous iterations

(3) $\mathbb{E}C_i(\boldsymbol{A})$ are finite and simultaneously triangularisable

We will refer to the C_i as the coefficient matrices and N as the inversion matrix.

An important fact is that if $n = 1$, this definition becomes the same as the one given by Arjevani et al. (2016). A key difference, however, is that the Jacobian $\boldsymbol{A}$ will generally not be symmetric for $n > 1$; only the blocks $\boldsymbol{M}_{ii}$ will be, and hence we may not assume the spectrum $\sigma(\boldsymbol{A})$ to be positive since it will generally be complex. Fortunately, several results from Arjevani et al. (2016) hold nevertheless, as discussed in Appendix B.3.1. Before introducing the results, let us give examples of algorithms used to optimise games that are p-SCLI-n, as evidenced by their update rule on quadratic games.

Simultaneous Gradient Descent (GD) The update rule is given by $\mathbf{w}_i^t = \mathbf{w}_i^{t-1} - \eta_i \nabla_{\mathbf{w}_i} l_i(\mathbf{w}^{t-1})$, which can be rewritten with $\boldsymbol{\eta} = \text{Diag}\,(\eta_1,...,\eta_n)$ as:

$$\begin{aligned}\mathbf{w}^t &= \mathbf{w}^{t-1} - \boldsymbol{\eta}\left(\boldsymbol{A}\mathbf{w}^{t-1} + \mathbf{b}\right) \\ &= (\boldsymbol{I} - \boldsymbol{\eta}\boldsymbol{A})\mathbf{w}^{t-1} - \boldsymbol{\eta}\mathbf{b}\end{aligned} \tag{41}$$

This shows that simultaneous gradient descent is a 1-SCLI-n algorithm.

Simultaneous Momentum GD The update rule is $\mathbf{w}_i^t = \mathbf{w}_i^{t-1} - \eta_i \nabla_{\mathbf{w}_i} l_i(\mathbf{w}^{t-1}) + \beta_i(\mathbf{w}_i^{t-1} - \mathbf{w}_i^{t-2})$ which can rewritten with $\boldsymbol{\beta} = \text{Diag}\,(\beta_1,...,\beta_n)$ and $\boldsymbol{\eta}$ as before:

$$\begin{aligned}\mathbf{w}^t &= \mathbf{w}^{t-1} - \boldsymbol{\eta}\left(\boldsymbol{A}\mathbf{w}^{t-1} + \mathbf{b}\right) + \boldsymbol{\beta}(\mathbf{w}^{t-1} - \mathbf{w}^{t-2}) \\ &= (\boldsymbol{I} - \boldsymbol{\eta}\boldsymbol{A} + \boldsymbol{\beta})\mathbf{w}^{t-1} - \boldsymbol{\beta}\mathbf{w}^{t-2} - \boldsymbol{\eta}\mathbf{b}\end{aligned} \tag{42}$$

Therefore, simultaneous gradient descent with momentum is a 2-SCLI-n, if we assume $\boldsymbol{\beta}$ to be scalar (since we need the coefficient matrices $C_i(\boldsymbol{A})$ to be simultaneously triangularisable).

Extragradient (Korpelevich, 1976) The update rule is $\mathbf{w}_i^t = \mathbf{w}_i^{t-1} - \eta_i \nabla_{\mathbf{w}_i} l_i(\mathbf{w}^{t-1} - \boldsymbol{\eta}\boldsymbol{v}(\mathbf{w}^{t-1}))$, which can be rewritten as:

$$\begin{aligned}\mathbf{w}^t &= \mathbf{w}^{t-1} - \boldsymbol{\eta}(\boldsymbol{A}(\mathbf{w}^{t-1} - \boldsymbol{\eta}(\boldsymbol{A}\mathbf{w}^{t-1} + \mathbf{b})) + \mathbf{b}) \\ &= (\boldsymbol{I} - \boldsymbol{\eta}\boldsymbol{A} + (\boldsymbol{\eta}\boldsymbol{A})^2)\mathbf{w}^{t-1} - (\boldsymbol{I} - \boldsymbol{\eta}\boldsymbol{A})\boldsymbol{\eta}\mathbf{b}\end{aligned} \tag{43}$$

This shows that extragradient is a 1-SCLI-n.

Simultaneous Stochastic Gradient Descent The reasoning is the same as the one presented by Arjevani et al. (2016): we approximate $\nabla f_{\boldsymbol{A},\mathbf{b}}(\mathbf{w}) = \boldsymbol{A}\mathbf{w} + \mathbf{b}$ with stochastic gradients $\boldsymbol{G}_\omega(\mathbf{w})$ and denote the error by $\boldsymbol{e}_\omega(\mathbf{w}) = \boldsymbol{G}_\omega(\mathbf{w}) - (\boldsymbol{A}\mathbf{w} + \mathbf{b})$. Then the update rule for fixed $\boldsymbol{\eta}$ is given by

$$\begin{aligned}\mathbf{w}^t &= \mathbf{w}^{t-1} - \boldsymbol{\eta}\boldsymbol{G}_{\omega_{t-1}}(\mathbf{w}^{t-1}) \\ &= (\boldsymbol{I} - \boldsymbol{\eta}\boldsymbol{A})\mathbf{w}^{t-1} - \boldsymbol{\eta}\mathbf{b} - \boldsymbol{\eta}\boldsymbol{e}_{\omega_{t-1}}(\mathbf{w}^{t-1})\end{aligned} \tag{44}$$

Under certain assumptions, e.g. if $\boldsymbol{e}_\omega(\mathbf{w}) = \boldsymbol{A}_\omega\mathbf{w} + \boldsymbol{N}_\omega\mathbf{b}$ and $\mathbb{E}\boldsymbol{A}_\omega = \mathbb{E}\boldsymbol{N}_\omega = 0$, then the update rule becomes

$$\mathbf{w}^t = (\boldsymbol{I} - \boldsymbol{\eta}(\boldsymbol{A} + \boldsymbol{A}_{\omega_{t-1}}))\mathbf{w}^{t-1} - \boldsymbol{\eta}(\boldsymbol{I} + \boldsymbol{N}_{\omega_{t-1}})\mathbf{b} \tag{45}$$

and we get a 1-SCLI-n.

One last definition is required before we introduce the p-SCLI lower bounds. Our definition generalises that of Arjevani et al. (2016).

Definition 3 (Consistency of p-SCLI-n optimisation algorithms)**.** *Let $\mathcal{Q}_{\boldsymbol{A}}^{d_1,\ldots,d_n} \subseteq \mathcal{Q}^{d_1,\ldots,d_n}$ denote the set of quadratic n-player games with non-singular Jacobian $\boldsymbol{A}$ (see eq. 32). Then $\mathcal{A}$ is consistent with respect to $\boldsymbol{A}$ if for any game $f_{\boldsymbol{A},\mathbf{b}} \in \mathcal{Q}_{\boldsymbol{A}}^{d_1,\ldots,d_n}$ and any initialisation, $\mathcal{A}$ converges to a stationary point of $f_{\boldsymbol{A},\mathbf{b}}$ or equivalently if the sequence of iterates $(\mathbf{w}^t)$ (see eq. 40) satisfies*

$$\mathbf{w}^t \to -\boldsymbol{A}^{-1}\mathbf{b} \tag{46}$$

Equivalently, as Arjevani et al. (2016) argue in their section 3.1, consistency with respect to some invertible Jacobian $\boldsymbol{A}$ is equivalent to having $\mathcal{A}$ converge on $f_{\boldsymbol{A},\mathbf{b}}$ and

$$\sum_{i=0}^{p-1} \mathbb{E}C_i(A) = I_d + \mathbb{E}N(\boldsymbol{A})\boldsymbol{A} \tag{47}$$

Note that all three examples of optimisation algorithms discussed in this subsection satisfy eq. 47.

4.5.2. Parametric lower bound for p-SCLI-n with scalar inversion matrix

We are now ready to introduce the lower bound for p-SCLI-n methods with scalar inversion matrix.

Proposition 2. *Let $\mathcal{A}$ be a p-SCLI-n algorithm with scalar inversion matrix for optimising games over $\mathbb{R}^{d_1} \times \ldots \times \mathbb{R}^{d_n}$. Then for quadratics $f_{\boldsymbol{A},\mathbf{b}} \in \mathcal{Q}^{d_1,\ldots,d_n}$, if $\mathcal{A}$ is consistent with respect to $\boldsymbol{A}$ and if $0 \notin \sigma(\boldsymbol{A})$, we have the following lower bound on the (linear) rate of convergence ρ:*

$$\rho \geq \frac{\sqrt[p]{\kappa} - 1}{\sqrt[p]{\kappa} + 1} = 1 - \frac{2}{\sqrt[p]{\kappa} + 1} \tag{48}$$

where the condition number κ is defined as $\kappa \triangleq \frac{\max|\sigma(\boldsymbol{A})|}{\min|\sigma(\boldsymbol{A})|}$ where $\sigma(\boldsymbol{A})$ is the spectrum of $\boldsymbol{A}$.

While this lower bound is valid for p-SCLI-n methods on *any single quadratic game where the assumptions apply*, it also gives us a more general result for problem classes containing quadratics as an immediate corollary, similarly to the lower bounds of the previous section.

Theorem 3. *For any problem class containing quadratic games, there exists a game such that the lower bound given in eq. 48 holds for a p-SCLI-n method.*

Corollary 4 (Iteration complexity bound). *For the same problem classes and under the same assumptions as Theorem 3, the minimal number of steps t to satisfy $\max_{i=0,\ldots,p-1}\left\|\mathbb{E}\mathbf{w}^{t+i} - \mathbb{E}\mathbf{w}^*\right\| < \epsilon$ is given by*

$$t \geq \left(\frac{\sqrt[p]{\kappa} - 1}{2}\right) \log\left(\frac{C}{\epsilon}\right) \tag{49}$$

for some strictly positive constant C.

See Appendix B.3.1 for the proof of Prop. 2, and Appendix B.4 for the iteration complexity bound's proof. Interestingly, by setting $n = 1$, this bound captures the 1-player case for μ-strongly convex objectives with L-Lipschitz gradients, where $\kappa = \frac{\max \sigma(\boldsymbol{A})}{\min \sigma(\boldsymbol{A})} = L/\mu$, verifying the intuition discussed at the end of the previous section, and showing that this technique yields a tight lower bound for $n = 1$. Moreover, this form is valid for n-player games (and min-max problems) in finite dimension, and κ arises naturally from the spectral properties of the update rules of the p-SCLI-n methods and is lower bounded by 1. Additionally, our bounds are valid for some stochastic methods. In single-objective optimisation (i.e. $n = 1$), while linear rates are not achievable for general stochastic problems, for which the worst-case bounds are sublinear, under certain conditions linear rates are possible (Loizou and Richtárik, 2017). Conditions of this type can be satisfied in over-parameterised neural networks (Vaswani et al., 2019). Hence, our linear lower bounds may be useful even for stochastic problems.

However, while the moduli in the $n > 1$ case allow us to handle complex spectra and matches the classical definition of condition number from linear algebra, several analyses have shown that not only the modulus, but also the relative size of the real and imaginary parts of elements of the spectrum matter (Mescheder et al., 2017; Gidel et al., 2019b). Such an analysis may yield more expressive bounds, but is out of the scope of this work. We will nevertheless give a more explicit form of the bound for 2-player games for which $d_1 = d_2$ that will make the μ_i and L_i appear.

Some explicit bounds for p-SCLI-2 with $d_1 = d_2$ Prop. 2 may be used to derive lower bounds for 2-player games for which $d_1 = d_2$. These bounds depend on the value of the μ_i and L_i defined as in eq. 33. Namely, let

$$\begin{aligned}\Delta_\mu &= (\mu_1 + \mu_2)^2 - 4(\mu_1\mu_2 + \mu_{12}^2) \\ &= (\mu_1 - \mu_2)^2 - 4\mu_{12}^2 \qquad (50)\\ \Delta_L &= (L_1 + L_2)^2 - 4(L_1L_2 + L_{12}^2) \\ &= (L_1 - L_2)^2 - 4L_{12}^2 \qquad (51)\end{aligned}$$

Table 1 gives lower bounds on the condition number that may then be plugged into eq. 48 to get lower bounds on two-players games corresponding to min-max problems (and are therefore lower bounds for general 2-player games).

Table 1. Lower Bounds on the Condition Number

	$\Delta_\mu < 0$	$\Delta_\mu \geq 0$
$\Delta_L < 0$	$\kappa = \sqrt{\frac{L_1L_2+L_{12}^2}{\mu_1\mu_2+\mu_{12}^2}}$	$\kappa \geq 2\frac{\sqrt{L_1L_2+L_{12}^2}}{\mu_1+\mu_2-\sqrt{\Delta_\mu}}$
$\Delta_L \geq 0$	$\kappa \geq \frac{1}{2}\frac{L_1+L_2+\sqrt{\Delta_L}}{\sqrt{\mu_1\mu_2+\mu_{12}^2}}$	$\kappa \geq \frac{L_1+L_2+\sqrt{\Delta_L}}{\mu_1+\mu_2-\sqrt{\Delta_\mu}}$

See Appendix B.3.2 for the counterexample in $\mathcal{P}$ leading to these bounds. This result resolves the issues raised in the discussion of the domino bounds as it uses all of the μ_i and L_i, and is proven from finite-dimensional problems. In fact, if one sets $\mu_1 = L_1$ and $\mu_2 = L_2$ such that μ_1 and μ_2 are small (in particular, smaller than μ_{12}) and L_{12} is large (in particular, larger than L_1, L_2), table 1 yields $\kappa = \sqrt{\frac{\mu_1\mu_2+L_{12}^2}{\mu_1\mu_2+\mu_{12}^2}}$, which coincides with the κ from Thm. 1. More generally, for $p = 1$, if both Δ_L and Δ_μ are negative, we get a tighter bound from the p-SCLI-2 formalism for 1-SCLI-2 methods that also satisfy the two-step linear span assumption than from Thm. 1. Finally, it provides the same plug-and-play convenience as p-SCLI to derive bounds for a large class of algorithms that may not satisfy the first-order black box assumption. On the other hand, the bounds may not be tight for $p \geq 3$, as it was the case for single-objective p-SCLI (Arjevani et al., 2016).

An interesting case is $p = 2$: for 2-SCLI-2 methods that satisfy the two-step linear span assumption such as negative momentum, the rate stemming from the p-SCLI-2 analysis appears to be smaller than the rate from the improved domino bound. This may be because the proof techniques used in our generalisation of p-SCLI yield bounds that can be improved for $p > 1$. In particular, a key difference between Thm. 1 and 3 is that as explained in the background section, lower bounds generally need not apply to every single problem within a class. However, due to the proof technique used, the bound given in Prop. 2 has to apply to every single quadratic game where the assumptions hold. Because some games might be inherently easier than others, such games may bottleneck the bound in Prop. 2 and Thm. 3, and introduce looseness. Indeed, first, the number of players does not appear in Prop. 2's proof, meaning that the proof yielded a bound that should hold for $n = 1$ which might inherently have faster convergence than $n > 1$ due to having to optimise only one objective over one set of parameters. Second, in the proof, because we do not impose a structure for the quadratic games, the bound has to hold for purely cooperative games (that is, the objective of each player does not depend on other players' parameters), in which case the situation is analogous to n single-objective optimisation problems, where we may expect the convergence to be faster than in games with interactions between players. As such, it is worth considering whether fixing the number of players and the structure of the games earlier in the proof could improve the bound.

4.6. Conclusion

In this work, we provide linear lower bounds and condition numbers for any problem class containing quadratic or bilinear games. We give a lower bound on the rate of convergence of first-order black box methods for 2-player games (which directly applies to min-max and saddle point problems) satisfying the two-step linear span assumption by generalising Nesterov's lower bound for the optimisation of strongly convex, smooth convex objectives to 2-player games (R.Q. 1) and constructing a novel class of hard problems using spectral properties of a class of operators in Banach algebras. Moreover, we generalise the framework of p-SCLI, which requires symmetricity

of the Hessian in single-objective optimisation, to provide a bound for a large class of optimisers for n-player games by extending the results of p-SCLI to quadratic games with non-symmetric Jacobian, which for $n = 1$ recovers Arjevani et al. (2016)'s tight bounds. We then give explicit bounds for 2-player games, which apply to min-max and saddle point problems (R.Q. 2). Finally, we derived formulations for the condition number that matched (in the case of the first domino bound), or were more general (in the case of the improved domino bound, and p-SCLI-n and p-SCLI-2 bounds) than the existing ones in the upper bound literature (R.Q. 3). As in the single-objective case, our bounds and condition numbers suggest that optimisers may converge faster on games for which the eigenvalues are at a similar, remote distance from the origin (e.g. on a circle) than on games for which some eigenvalues are close to and others are far from 0.

Following the initial release of this work, several other authors have built upon the bounds presented in this paper. The bound from Theorem 1 is tight on the class of smooth strongly-convex-strongly-concave games, as it is matched by the upper bound presented by Fallah et al. (2020), and up to logarithmic factors, by the upper bound of Lin et al. (2020). Additionally, this same lower bound was also shown to be tight on the class of bilinear games with non-singular Jacobian, which are merely convex-concave, by Prop. 4 of Azizian et al. (2020). As a corollary, our results establish the optimality of those methods on the aforementioned classes of problems.

However, several directions remain to be explored. For example, we raised the question of whether the p-SCLI-n bound could be tightened for $p > 1$ by improving the proof technique, especially given that we are not aware of faster rates of convergence in the literature than those of $p = 1$ or those of the improved domino bound. Moreover, we would like to present a more exhaustive overview of the extension of p-SCLI, and discuss the resulting upper and lower bounds for various commonly used algorithms. In particular, we would like to extend our lower bounds to p-SCLI-n with diagonal inversion matrices, as Arjevani et al. (2016) did in the p-SCLI framework, and provide bounds in the 2-player case when $d_1 \neq d_2$. Furthermore, we believe tighter bounds may be derived, for example by adding constraints on the C_i or by looking not only at the modulus of the eigenvalues but also at their arguments, as done by Gidel et al. (2019b), since we know that the relative size of the imaginary part and real part (even at fixed modulus) affects the dynamics in games (Mescheder et al., 2017). Finally, it would be important to understand when and how linear convergence is possible in non strongly-convex-strongly-concave settings. We plan on exploring several of these directions in future work.

Acknowledgements

The authors would like to thank Damien Scieur for useful discussions and feedback. This work was partially supported by the FRQNT new researcher program (2019-NC-257943), the NSERC Discovery grant (RGPIN-2019-06512), a startup grant by IVADO, a Microsoft Research

collaborative grant and a Canada CIFAR AI chair. Gauthier Gidel was partially supported by a Borealis AI Graduate Fellowship.

Chapter 5

Prologue to Second Article

5.1. Article Details

Towards Out-of-Distribution Adversarial Robustness. Adam Ibrahim, Charles Guille-Escuret, Ioannis Mitliagkas, Irina Rish, David Krueger, Pouya Bashivan.

5.2. Contributions

Adam and Ioannis discussed the idea of framing robustness against multiple attacks as an OoD generalisation problem and Adam mentioned it in his predoctoral presentation under Ioannis's supervision. The specific method from the domain generalisation literature was determined through discussions involving Ioannis, Irina, Pouya and Adam. After variance Risk Extrapolation (REx) (Krueger et al., 2021) was chosen, Irina invited David to the project to give feedback on the use of REx and the OoD literature. Adam designed and ran the experiments, with feedback from Pouya. Charles helped with processing the results and producing plots. Pouya, David and Adam wrote the introduction and background section, and Adam wrote the rest. All authors helped review and improve the writing.

5.3. Context

Adversarially training (Goodfellow et al., 2015; Madry et al., 2018) on samples generated with a given attack (usually PGD L_∞) had been an effective strategy to improve robustness against that attack. However, robustness against a specific norm of PGD was shown theoretically (Khoury and Hadfield-Menell, 2018; Tramèr and Boneh, 2019) and empirically (Maini et al., 2020) not to generalise to PGD of other norms. Worse, several effective attacks, such as Carlini-Wagner (Carlini and Wagner, 2017a) or non-L_p attacks (Xiao et al., 2018; Laidlaw et al., 2021), had not been studied, as most of the analysis focused on PGD and AutoAttack (Croce and Hein, 2020). This prompted some research into robustness against multiple perturbations such as (Tramèr and Boneh,

2019; Maini et al., 2020), but such research still mostly focused on PGD and did not establish or leverage the connection between robustness against multiple and potentially unforeseen adversaries at test time, and out-of-distribution generalisation. The goal of this work was twofold: highlight that different tunings or families of attacks induce different distributions of adversarial examples, and that an OoD generalisation method can be successful at improving adversarial robustness even against attacks that were not seen during training.

Chapter 6

Towards Out-of-Distribution Adversarial Robustness

6.1. Introduction

Vulnerability to adversarial perturbations (Biggio et al., 2013; Szegedy et al., 2014; Goodfellow et al., 2015) is a major concern for real-world applications of machine learning such as healthcare (Qayyum et al., 2020) and autonomous driving (Deng et al., 2020). For example, Eykholt et al. (2018) show how seemingly minor physical modifications to road signs may lead autonomous cars into misinterpreting stop signs, while Li et al. (2020) achieve high success rates with over-the-air adversarial attacks on speaker systems.

Much work has been done on defending against adversarial attacks (Goodfellow et al., 2015; Papernot et al., 2016). However, new attacks commonly overcome existing defenses (Athalye et al., 2018). A defense that has so far passed the test of time against individual attacks is adversarial training. Goodfellow et al. (2015) originally proposed training on examples perturbed with the Fast Gradient Sign Method (FGSM), which performs a step of sign gradient ascent on a sample x to increase the chances of the model misclassifying it. Madry et al. (2018) further improved robustness by training on Projected Gradient Descent (PGD) adversaries, which perform multiple updates of (projected) gradient ascent to try to generate a maximally confusing perturbation within some L_p ball of predetermined radius ϵ centred at the chosen data sample.

Unfortunately, adversarial training can fail to provide high robustness against several attacks, or tunings of attacks, only encountered at test time. For instance, simply changing the norm constraining the search for adversarial examples with PGD has been shown theoretically and empirically (Khoury and Hadfield-Menell, 2018; Tramèr and Boneh, 2019; Maini et al., 2020) to induce significant trade-offs in performance against PGD of different norms. This issue highlights the importance of having a well-defined notion of "robustness": while using the accuracy against individual attacks has often been used as a proxy for robustness, a better notion of robustness, as argued by Athalye et al. (2018), is to consider the accuracy against an ensemble of attacks within a threat model (i.e. a predefined set of allowed attacks). Indeed, in the example of autonomous

driving, an attacker will not be constrained to a single attack on stop signs, and is free to attempt several attacks to find one that succeeds.

In order to be robust against multiple attacks, we draw inspiration from domain generalisation. In domain generalisation, we seek to achieve consistent performance even in case of unknown distributional shifts in the inputs at test time. We interpret different attacks as distinct distributional shifts in the data, and propose to leverage existing techniques from the out-of-distribution generalisation literature.

We choose variance REx (Krueger et al., 2021), which consists in using as a loss penalty the variance on the different training domains of the empirical risk minimisation loss. We choose this method as it is conceptually simple, its iterations are no more costly than existing multi-perturbation baselines', it does not constrain the architecture, and it can be used on models pretrained with existing defenses. We consider robustness against an adversary having access to both the model and multiple attacks.

However, there are multiple potential challenges: first, Gulrajani and Lopez-Paz (2020) show that domain generalisation methods, such as REx, often fail to improve over empirical risk minimisation (ERM) in many settings. Thus, it is possible that REx would fail to improve Tramèr and Boneh (2019)'s defense, which uses ERM. Second, domain generalisation methods are usually designed for stationary settings, whereas in adversarial machine learning, the distribution of adversarial perturbations is non-stationary during training as the attacks adapt to the changes in the model parameters. Finally, the state-of-the-art multi-perturbation defense proposed by Maini et al. (2020), which we intend to improve with REx, does not explicitly train on multiple domains, which REx originally requires.

Therefore, we are interested in the two following research questions:

(1) Can REx improve robustness against multiple attacks seen during training?

(2) Can REx improve robustness against unseen attacks, that is, attacks only seen at test time?

Our results show that the answer to both questions is yes on the ensembles of attacks used in this work. We show that REx consistently yields benefits across variations in: datasets, architectures, multi-perturbation defenses, hyperparameter tuning, attacks seen during training, and attack types or tunings only encountered at test time.

6.2. Related Work

6.2.1. Adversarial attacks and defenses

Since the discovery of adversarial examples against neural networks (Szegedy et al., 2014), numerous approaches for finding adversarial perturbations (i.e. adversarial attacks) have been proposed (Goodfellow et al., 2015; Madry et al., 2018; Moosavi-Dezfooli et al., 2016; Carlini and Wagner, 2017b; Croce and Hein, 2020), with the common goal of finding perturbation vectors with

constrained magnitude that, when added to the network's input, lead to (often highly confident) misclassification.

One of the earliest attacks, the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), computes a perturbation on an input x^0 by performing a step of sign gradient ascent in the direction that increases the loss L the most, given the model's current parameters θ. This yields an adversarial example $\tilde{x}$ that may be misclassified:

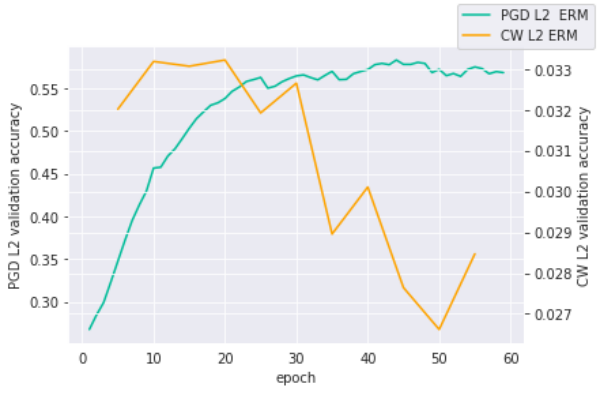
$$\tilde{x} = x^0 + \alpha \operatorname{sgn}(\nabla_x L(\theta, x^0, y)). \tag{52}$$

This was later enhanced into the Projected Gradient Descent (PGD) attack (Kurakin et al., 2017; Madry et al., 2018) by iterating multiple times this operation and adding projections to constrain it to some neighbourhood of x^0, usually a ball of radius ϵ centered at x^0, noted $\mathcal{B}_\epsilon(x^0)$:

$$x^{t+1} = \Pi_{\mathcal{B}_\epsilon(x^0)} \left(x^t + \alpha \operatorname{sgn}(\nabla_x L(\theta, x^t, y)) \right). \tag{53}$$

With the advent of diverse algorithms to *defend* classifiers against such attacks, approaches for discovering adversarial examples have become increasingly more complex over the years. Notably, it was found that a great number of adversarial defenses rely on *gradient obfuscation* (Athalye et al., 2018), which consists in learning how to mask or distort the classifier's gradients to prevent attacks iterating over gradients from making progress. However, it was later discovered that such approaches can be broken by other attacks (Athalye et al., 2018; Croce and Hein, 2020), some of which bypass these defenses by not relying on gradients (Brendel et al., 2019; Andriushchenko et al., 2020).

Figure 7. Validation accuracy of a model adversarially trained on PGD L_2-perturbed CIFAR10 with a ResNet18, evaluated on PGD L_2 and Carlini & Wagner (CW) L_2 attacks. Curves are smoothed with exponential moving averaging (weight 0.7).



A defense that was shown to be robust to such countermeasures is Adversarial Training (Madry et al., 2018), which consists in training on adversarial examples. Adversarial training corresponds to solving a minimax optimisation problem where the inner loop executes an adversarial attack algorithm, usually PGD, to find perturbations to the inputs that maximise the classification loss, while the outer loop tunes the network parameters to minimise the loss on the adversarial examples. Despite the method's simplicity, robust classifiers trained with adversarial training achieve state-of-the-art levels of robustness against various newer attacks (Athalye et al., 2018; Croce and Hein, 2020). For this reason, adversarial training has become one of the most common defenses.

However, Khoury and Hadfield-Menell (2018) and Tramèr and Boneh (2019) show how training on PGD with a search region constrained by a p-norm may not yield robustness against PGD attacks using other p-norms. One reason is that different radii are typically chosen for different norms, leading to the search spaces of PGD with respect to different norms to potentially have some mutually exclusive regions. Another reason is that different attacks, such as PGD and the Carlini and Wagner (Carlini and Wagner, 2017b) attacks, optimise different losses (note that this is also true for PGD of different norms). As an example, Fig. 7 illustrates how, when training adversarially a model on L_2-norm PGD, the accuracy against one attack may improve while it may decrease against another attack, even if the attacks use the same p-norm.

Highlighting the need for methods specific to defending against multiple of perturbations, Tramèr and Boneh (2019) select a set of 3 attacks $\mathcal{A} = \{P_\infty, P_2, P_1\}$, where P_p is PGD with a search region constrained by the L_p norm. They attempt two strategies: the **average (Avg) strategy** consists in training over all attacks in $\mathcal{A}$ for each input (x, y) in the dataset, and the max strategy, which trains on the attack with the highest loss for each sample:

$$L_{\text{Avg}}(\theta, \mathcal{A}) = \mathbb{E}\frac{1}{|\mathcal{A}|}\sum_{A\in\mathcal{A}} \ell(\theta, A(x), y) \tag{54}$$

$$L_{\text{max}}(\theta, \mathcal{A}) = \mathbb{E}\max_{A\in\mathcal{A}} \ell(\theta, A(x), y) \tag{55}$$

Maini et al. (2020) propose a modification to the max method: instead of 3 different PGD adversaries that each iterate over a budget of iterations as in eq. 53, they design an attack consisting in choosing the worst perturbation among L_∞, L_2 and L_1 PGD every iteration through the chosen number of iterations. This attack, **Multi-Steepest Descent (MSD)**, differs from the max approach of Tramèr and Boneh (2019) where each attack is individually iterated through the budget of iterations first, and the one leading to the worst loss is chosen at the end. Note that this implies that technically, unlike (Tramèr and Boneh, 2019)'s Avg approach, *MSD* only consists in training on a single attack.* Maini et al. (2020) show that, in their experimental setup, MSD yields superior performance to both the Avg and Max approaches.

Nevertheless, there is still a very large gap between the performance of such approaches against data perturbed by ensembles of attacks, and the accuracy on the unperturbed data. In order to help address this large gap, we will be exploiting a connection between our goal and domain generalisation.

6.2.2. Robustness as a domain generalisation problem

Domain generalisation – Out-of-Distribution generalisation (OoD) is an approach to dealing with (typically non-adversarial) distributional shifts. In the domain generalisation setting, the training data is assumed to come from several different domains, each with a different data distribution. The goal is to use the variability across training (or seen) domains to learn a model that

*In the rest of the paper, we will use MSD to refer to both the MSD attack, and training on MSD as a defense.

can generalise to unseen domains while performing well on the seen domains. In other words, the goal is for the model to have consistent performance by learning to be invariant under distributional shifts. Typically, we also assume access to domain labels, i.e. we know which domain each data point belongs to. Many methods for domain generalisation have been proposed – see (Wang et al., 2021) for a survey.

Our work views adversarial robustness as a domain generalisation problem, where the domains stem from different adversarial attacks. Because different attacks use different methods of searching for adversarial examples, and sometimes different search spaces, they may produce different distributions of adversarial examples[†]. One might draw an analogy to Hendrycks and Dietterich (2019)'s work on natural perturbations, where both the type and the strength of the perturbations play a similar role as varying the attacks or their tuning, respectively. There are several reasons why the domains we consider may be distributionally shifted with one another (although the distributions may have some overlap). To non-exhaustively name a few, first, we already evoked how different p-norms affect the distributions of adversarial examples yielded by PGD (Khoury and Hadfield-Menell, 2018; Tramèr and Boneh, 2019). Second, different attacks may optimise different losses – for example when comparing P_2 and L_2 CW – which may yield different solutions. Third, the same attack tuned differently (e.g. different ϵ or iteration budget) may yield different distributions of adversarial examples since they do not have the same support. Therefore, robustness to attacks unseen during training means robustness against the corresponding distributional shifts at test time. It is natural to frame adversarial robustness as a domain generalisation problem, as we seek a model that is robust to *any* method to generate adversarially distributional shifts within a threat model, including novel attacks.

In spite of this intuition, it is not obvious that such methods would work in the case of adversarial machine learning. First, recent work demonstrates that domain generalisation methods often fail to improve upon the standard **empirical risk minimisation (ERM)**, i.e. minimising loss on the combined training domains without making use of domain labels (Gulrajani and Lopez-Paz, 2020). On the other hand, success may depend on choosing a method appropriate for the type of shifts at play. Second, a key difference with most work in domain generalisation, is that when adversarially training, the training distribution shifts every epoch, as the attacks are computed from the continuously-updated values of the weights. In contrast, in domain generalisation, the training domains are usually fixed. Non-stationarity is known to cause generalisation failure in many areas of machine learning, notably reinforcement learning (Igl et al., 2020), thereby potentially affecting the success of domain generalisation methods in adversarial machine learning. Third, MSD does not generate multiple domains, which domain generalisation approaches would typically require.

We note that interestingly, the Avg approach of Tramèr and Boneh (2019) can be interpreted as doing domain generalisation with ERM over the 3 PGD adversaries as training domains. Similarly,

[†]Another way to think about this, is that if different attacks or tunings yielded identical distributions, then standard results from statistical learning theory would imply similar performance on the various attacks.

the max approach consists in applying the Robust Optimisation approach on the same set of domains. Furthermore, Song et al. (2018) and Bashivan et al. (2021) propose to treat the clean and PGD-perturbed data as training and testing domains from which some samples are accessible during training, and adopt domain adaptation approaches. Therefore, it is difficult to predict in advance how much a domain generalisation approach can successfully improve adversarial defenses.

In this work, we apply the method of **variance-based risk extrapolation (REx)** (Krueger et al., 2021), which simply adds as a loss penalty the variance of the ERM loss across different domains. This encourages worst-case robustness over more extreme versions of the shifts (here, shifts are between different attacks) observed between the training domains. This can be motivated in the setting of adversarial robustness by the observation that adversaries might shift their distribution of attacks to better exploit vulnerabilities in a model. In that light, REx is particularly appropriate given our objective of mitigating trade-offs in performance between different attacks to achieve a more consistent degree of robustness. We note that our implementation of REx has the same computational complexity per epoch as the MSD, Avg and max approaches, requiring the computation of 3 adversarial perturbations per sample.

6.3. Methodology

Threat model – In this work, we consider white-box attacks, which are typically the strongest type of attacks as they assume the attacker has access to the model and its parameters. Additionally, the attacks considered in the evaluations are gradient-based, with the exception of AutoAttack, which is composite and includes gradient-free perturbations (Croce and Hein, 2020). Because we assume that the attacker has access to all of these attacks, we emphasise that, as argued by Athalye et al. (2018), the robustness against the ensemble of the different attacks is a better metric for how the defenses perform than the accuracy on each individual attack. Thus, using ℓ_{01} as the 0-1 loss, we evaluate the performance on an ensemble of domains $\mathcal{D}$ as:

$$\mathcal{R} = 1 - \mathbb{E} \max_{D \in \mathcal{D}} \ell_{01}(\theta, D(x), y) \tag{56}$$

REx – We propose to regularise the average loss over a set of training domains $\mathcal{D}$ by the variance of the losses on the different domains:

$$L_{\text{REx}}(\theta, \mathcal{D}) = L_{\text{Avg}}(\theta, \mathcal{D}) + \beta \text{Var}_{D \in \mathcal{D}} \mathbb{E}\, \ell(\theta, D(x), y) \tag{57}$$

where ℓ is the cross-entropy loss. We start penalising by the variance over the training domains once the baseline's accuracies on the seen domains stabilise or peak.

Datasets and architectures – We consider two datasets: MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky et al., 2009). It is still an open problem to obtain high robustness against multiple attacks on MNIST (Tramèr and Boneh, 2019; Maini et al., 2020), even at standard tunings of some commonly used attacks. On MNIST, we use a 3-layer perceptron of size [512, 512, 10]. On CIFAR10, we use the ResNet18 architecture (He et al., 2016). We choose two significantly different

architectures to illustrate that our approach may work agnostically to the choice of architecture. We always use batch sizes of 128 when training.

Optimiser – We use Stochastic Gradient Descent (SGD) with momentum 0.9. In subsections 6.4.2 and C.2.2 we do not perform hyperparameter optimisation, to isolate the effect of REx from interactions with hyperparameter tuning, which would differ for each defense. We use a fixed learning rate of 0.01 and no weight decay. We fix the coefficient β in the REx loss. In subsection 6.4.4, we optimise hyperparameters. Based on (Rice et al., 2020) and (Pang et al., 2020), we use in all cases a weight decay of $5 \cdot 10^{-4}$ and a piecewise learning rate decay. For every defense, we search for an optimal epoch to decay the learning rate, with a particular attention to MSD and MSD+REx due to observing a high sensitivity to the choice of learning rate decay milestone. Note that in the case of REx defenses, we always use checkpoints of corresponding baselines before the learning rate is decayed, as we observed this to lead to better performance.

Domains – We consider several domains: unperturbed data, L_1, L_2 and L_∞ PGD (denoted P_1, P_2, P_∞), L_2 Carlini & Wagner (CW_2) (Carlini and Wagner, 2017b), L_∞ DeepFool (DF_∞) (Moosavi-Dezfooli et al., 2016) and AutoAttack (AA) (Croce and Hein, 2020). We use the Advertorch implementation of these attacks (Ding et al., 2019). For L_∞ PGD, CW and DF, we use two sets of tunings, see appendix C.1 for details. The attacks with a $\bullet$ superscript indicate a harder tuning of these attacks that no model was trained on. Those tunings are intentionally chosen to make the attacks stronger. The set of domains **unseen by all models** is defined as $\{P_\infty^\bullet, DF_\infty^\bullet, CW_2^\bullet, \textit{AutoAttack}_\infty\}$, with additionally $\textit{AutoAttack}_2$ in subsection 6.4.4. The set of domains **unseen by a specific model** is the set of all domains except those seen by the model during training, and therefore varies between baselines. We perform 10 attack restarts per sample to reduce randomness in the test set evaluations.

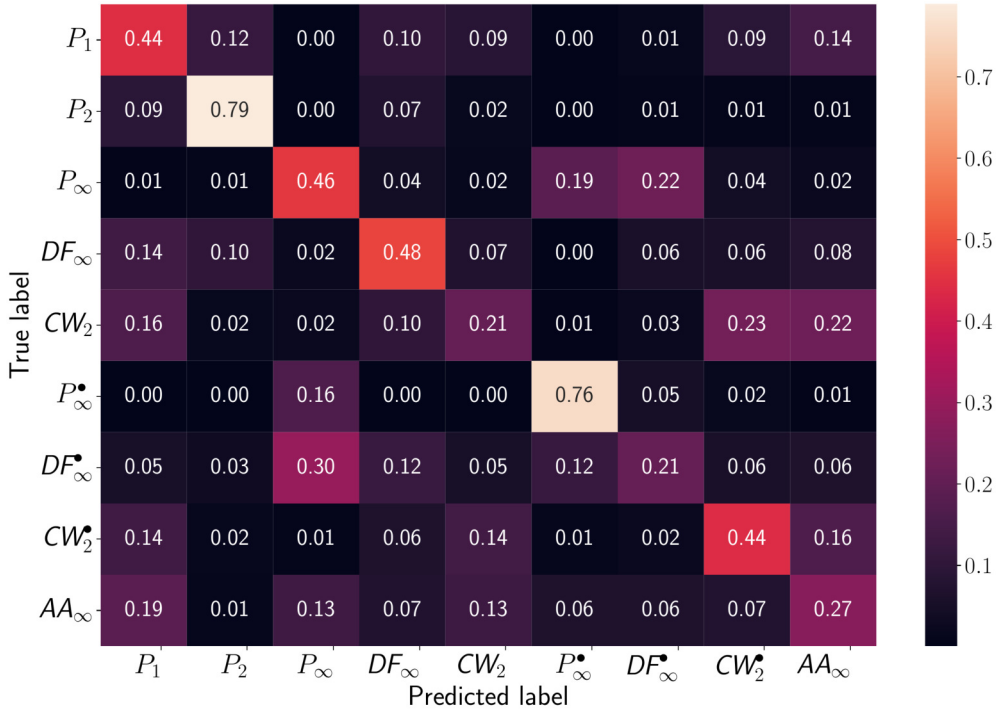
Defenses – Aside from the adversarial training baselines on PGD of L_1, L_2 and L_∞ norms, we define 3 sets of seen domains: $\mathcal{D} = \{\varnothing, P_\infty, DF_\infty, CW_2\}$, $\mathcal{D}_{\textit{PGDs}} = \{\varnothing, P_1, P_2, P_\infty\}$ and $\mathcal{D}_{\textit{MSD}} = \{\textit{MSD}\}$ where $\varnothing$ represents the unperturbed data. We train two Avg baselines: one on $\mathcal{D}$ and one on $\mathcal{D}_{\textit{PGDs}}$. We train the MSD baseline on $\mathcal{D}_{\textit{MSD}}$. We use REx on the Avg baselines on the corresponding set of seen domains. However, when REx is used on the model trained with the MSD baseline, we revert to using the set of seen domains $\mathcal{D}_{\textit{PGDs}}$. While the MSD baseline does not exactly train over P_1, P_2 and P_∞ but rather a composition of these three attacks, we use these attacks when applying REx to the MSD baseline as MSD would only generate one domain, which would not allow us to compute a variance over domains. Note that we chose different sets of seen domains, and different baselines (Avg and MSD), in order to show that REx yields benefits on several multi-perturbation baselines, or within a same baseline with different choices of seen domains. We use **cross-entropy** for all defenses.

See Appendix C.1 for more details about the methodology, such as attack tunings.

6.4. Results

In this section, we first illustrate the differences in distributions stemming from different families or tunings of attacks by training an attack classifier in subsection 6.4.1. We then present our results on MNIST and CIFAR10 in subsections 6.4.2 and 6.4.4. We find that REx consistently improves the baselines over ensembles of seen or unseen attacks. REx often sacrifices the performance on the best performing domains (usually, the unperturbed data, P_1 and P_2) but improves the worst-case accuracy by a much wider margin. This is the case even when using REx on MSD. Also, we find that in general, REx is relatively robust to the choice of β, albeit it can be used to tune relative performance between easier and harder domains (see Appendix C.4). Additionally, more results on the interaction between the advantage in using REx and hyperparameter tuning, the relative performance of REx models on CIFAR10-C and for transfer learning, and more, can be found in Apdx C.2. More implementation details, observations and results can generally be found in the appendix (see Table of Content).

Figure 8. Confusion matrix of a discriminator between attacks (normalised by row).



6.4.1. Attacks as different domains

Before presenting our robustness results, to illustrate how various types or tunings of attacks may correspond to different distributions, we finetune an ImageNet-pretrained vision transformer (ViT) (Dosovitskiy et al., 2021) to predict which attack was used on the training set of CIFAR10, and test it on the perturbed CIFAR10 test set. Fig. 8 illustrates how without much engineering effort, the ViT is able to tell apart the distribution of perturbations induced by the various attacks with 45.1% accuracy, relative to a random chance of $1/9 \simeq 0.11$. To claim unequivocally that all attacks considered are

distributionally shifted with respect to one another, we require $p(A_{true}) > p(A \neq A_{true})$ where A_{true} is the true attack used to generate the sample and $p(A)$ is the probability predicted by the model that a perturbation corresponds to attack A. This is true for all attacks, except two: CW_2 and $DF^{\bullet}_{\infty}$.

The cases of CW_2 and $DF^{\bullet}_{\infty}$ (the latter sharing $\epsilon = 8/255$ with P_{∞} and AA_{∞}) warrant an additional discussion. The former is due to unsuccessful CW_2 iterations stopping early, i.e. $\tilde{x} \simeq x$. This problem disappears when such unsuccessful CW_2 perturbations are no longer involved. This is why the stronger $CW^{\bullet}_2$ is classified more easily. As for $DF^{\bullet}_{\infty}$ confused with P_{∞}, we direct the reader to Apdx C.2.1 for more details, where we show how this problem vanishes when training a binary $DF^{\bullet}_{\infty}$ vs P_{∞} discriminator.

Key observation 1: Different types or tunings of attacks induce different distributional shifts that a discriminator can identify to some extent, even for the same choice of p-norm and ϵ.

6.4.2. MNIST

Table 2. Accuracy on MNIST for different domains. Highlighted cells indicate that the domain (row) was used during training by the defense (column). Bold numbers indicate an improvement of at least 1% accuracy over the baseline used to pretrain REx. Ensembles omit $P^{\bullet}_{\infty}$ due to it being overtuned (i.e. tuned to be too strong.).

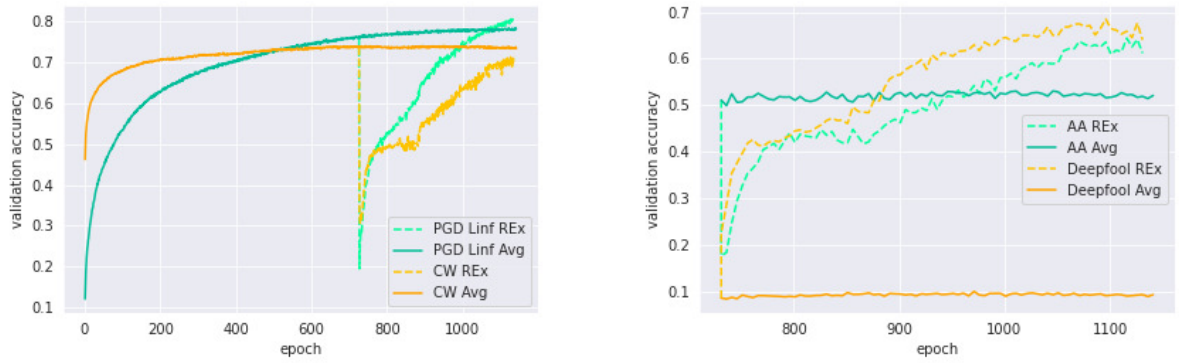
	Defenses									
	None	Adversarial training			Avg	Avg+REx	Avg_{PGDs}	Avg+REx_{PGDs}	MSD	MSD+REx
No attack	98.1	98.5	98.3	84.4	99.0	90.0	98.8	87.3	88.4	**90.2**
P_1	95.5	96.8	96.8	44.0	90.3	72.6	95.6	82.5	82.2	**86.8**
P_2	1.8	17.7	63.5	10.0	53.6	44.0	68.3	**72.8**	61.1	**71.8**
P_{∞}	0.0	0.0	2.2	59.2	67.7	**70.1**	58.0	**70.8**	19.3	**67.4**
DF_{∞}	3.3	5.7	85.9	78.1	92.9	84.6	92.3	80.9	56.7	**82.4**
CW_2	4.4	6.9	56.5	62.3	68.8	68.3	59.9	41.4	77.1	47.3
$DF^{\bullet}_{\infty}$	0.0	0.0	0.0	19.4	7.1	**64.8**	3.7	**58.4**	15.8	**19.9**
$CW^{\bullet}_2$	2.3	2.8	16.0	30.2	23.2	**42.1**	16.4	12.1	40.2	12.9
$\text{AutoAttack}_{\infty}$	0.0	0.0	0.1	55.0	42.3	**58.8**	34.9	**40.6**	1.5	**31.2**
Ensemble (seen)	-	-	-	-	63.2	63.4	55.5	**64.5**	19.3	**60.1**
Ensemble (unseen by all models)	0.0	0.0	0.0	9.3	3.4	**34.6**	1.2	**8.1**	0.6	**3.9**
Ensemble (unseen by this model)	0.0	0.0	0.0	2.7	3.4	**25.9**	1.2	**8.1**	0.6	**3.9**
Ensemble (all)	0.0	0.0	0.0	2.7	3.4	**25.9**	1.2	**8.1**	0.6	**3.9**
$P^{\bullet}_{\infty}$	0.0	0.0	0.0	5.1	0.6	**4.0**	0.9	0.7	0.2	1.0

We report our multiperturbation robustness results on MNIST in Table 2. REx significantly improves robustness against the ensembles of attacks, whether seen or unseen, and in particular on P_{∞} and AutoAttack. REx also yields notable improvements against all ensembles, seen or unseen, when used on the Avg baselines. Note however that as in domain generalisation, when used on all baselines except MSD, REx sacrifices performance on the best performing seen domains in order to

improve the performance on the strongest attacks. We believe that this trade-off may be worth it for applications where robustness is critical, as for example the 9% of clean accuracy lost by using REx on one Avg baseline translates in an increase of robustness from 3.4% to 25.9% on the ensemble of all attacks excluding the overtuned (i.e. tuned too strongly, leading to negligible accuracy) $P_\infty^\bullet$ adversary.

Our test with tuning the $P_\infty^\bullet$ adversary with $\epsilon = 0.4$ instead of the common tuning of 0.3 on MNIST suggests that REx does not rely on gradient masking(Athalye et al., 2018) compared to the baselines, as the accuracy drops to near 0 values for all models, showing that attacks are successfully computed. This is reinforced by the REx models' AutoAttack performance. A second observation is that the MSD baseline performs surprisingly poorly against AutoAttack and P_∞. We note that experiments with a learning rate schedule (not reported here) did not significantly improve performance of MSD, ruling out the absence of schedule as a cause. While we use the original code of Maini et al. (2020), this could be because we did not use the same architecture as them on MNIST. Furthermore, Maini et al. (2020) did not evaluate on AutoAttack as their work predates the publication of Croce and Hein (2020). In any case, the MSD model did not achieve substantial robustness against P_∞ and AutoAttack in our experiments. This leads to poor performance against all ensembles of attacks, whether seen or unseen, as those include either P_∞ or AutoAttack adversaries. Finally, a third observation is that P_∞ training performs remarkably well in this experiment on the ensemble of attacks, compared to the multi-perturbation baselines.

Key observation 2 (MLP on MNIST): REx improves performance of all baselines on MNIST with a multilayer perceptron, from 3.4% with the best baseline to 25.9% accuracy against an ensemble of L_p attacks, sacrificing a little robustness against the weakest attacks.



(a) MNIST seen attacks

(b) MNIST unseen attacks

Figure 9. Validation accuracy of Avg on MNIST with and without REx (dashed line), against seen attacks (left) and unseen attacks (right) (AA=AutoAttack).

6.4.3. Early stopping and REx

In Figure 9, we observe how regularising by the variance over the domains leads to better peak performance when using REx, over different baselines. Like the baselines, we early stop REx models. For all defenses, we use the validation set to choose when to early stop, by selecting the epoch when the performance peaks on the ensemble of seen domains. As shown on the curves, even though we stop training before REx reaches higher performance on the seen attacks, we still get significant improvements on the individual unseen attacks and against the ensembles, as reported in Table 2. See Appendix C.1 for more details on how REx is used, and Appendices C.2.2 and C.3 for more details on how, in contrast with MNIST with the MLP, early stopping is required much earlier and the validation curves no longer behave monotonically on a different dataset and architecture.

6.4.4. CIFAR10 with hyperparameter optimisation

On CIFAR10 with the ResNet18, we use weight decay and search for an optimal learning rate schedule individually for each defense. The results are summarised in Table 3. We chose not to use the Avg_{PGDs} baseline here, due to performing worse compared to the other baselines on CIFAR10 with a ResNet18 (which we confirm to be especially true when tuning hyperparameters in preliminary experiments). We direct the reader to Apdx C.2.2 for CIFAR10 results with an ablation on custom hyperparameter optimisation for each model, which leads to even greater advantages in using REx, and which includes the Avg_{PGDs} baseline.

Table 3. Accuracy on CIFAR10, with hyperparameter tuning. Ensembles omit $CW_2^\bullet$ due to overtuning.

	Defenses				
	P_∞	Avg	Avg+REx	MSD	MSD+REx
No attack	80.8	80.0	76.8	78.6	77.4
P_1	78.2	78.3	74.9	76.6	75.2
P_2	70.0	67.9	68.7	69.8	68.7
P_∞	47.3	34.4	**48.1**	45.8	**48.3**
DF_∞	69.0	64.4	**67.1**	67.1	67.3
CW_2	17.4	14.5	**29.6**	17.9	**20.9**
$P_\infty^\bullet$	28.9	16.9	**28.2**	27.4	**30.7**
$DF_\infty^\bullet$	46.3	35.2	**45.3**	44.9	**46.2**
AutoAttack_∞	44.8	33.5	**43.1**	42.8	**44.8**
AutoAttack_2	57.7	59.2	58.4	61.1	56.6
Ensemble (seen)	-	14.5	**29.2**	45.8	**48.2**
Ensemble (unseen by all models)	28.9	16.9	**27.9**	27.4	**30.3**
Ensemble (unseen by this model)	16.9	16.9	**27.9**	16.5	**19.6**
Ensemble (all)	16.9	14.2	**23.5**	16.5	**19.6**
$CW_2^\bullet$	2.5	4.8	5.3	1.9	**3.7**

The cumulative effect of weight decay and learning rate schedules significantly improves all defenses' robustness, as shown by Rice et al. (2020) and Pang et al. (2020). Once again, we observe that REx improves significantly the seen and unseen ensemble accuracies over the baselines. We also note that P_∞ adversarial training performs better than the

baselines on the ensemble of attacks used in this paper, even with the addition of AutoAttack$_2$ to the ensembles containing unseen attacks. Moreover, only REx is able to perform better than P_∞ adversarial training on P_∞ attacks. In other words, multi-perturbation defenses only perform better than P_∞ against ensembles of attacks when used with REx.

While MSD performs significantly better with a ResNet18 on CIFAR10 than with the MLP on MNIST (likely due to using the same architecture as them on CIFAR10), as suspected when discussing our optimiser methodology in Sec. 6.3, there are interaction effects between hyperparameter tuning and the performance of REx relative to a baseline. Improvements of MSD+REx over MSD are sensitive to hyperparameter tuning, specifically at which epoch to start using REx, and when to decay the learning rate. This sensitivity, and the lower advantage of MSD+REx over MSD, is likely due to the fact that the MSD baseline does not train over multiple domains. REx was originally designed to be used with a baseline using ERM on multiple domains as loss function. Therefore, when it is used in tandem with MSD, REx uses the loss indicated in eq. 57. However, because as mentioned before, the Avg$_{PGDs}$ baseline performs significantly worse than the MSD baseline, it is likely that the advantage in using REx is impacted negatively by the suboptimality of the first (ERM) term in the REx loss. Nevertheless, the variance penalty is beneficial enough to achieve higher robustness with MSD+REx than MSD.

Table 4. Accuracy on two non-L_p perturbations to CIFAR10.

	Defenses				
	P_∞	Avg	Avg+REx	MSD	MSD+REx
RecolorAdv	50.5	24.5	**63.5**	56.0	**58.2**
StAdv	12.1	4.0	**31.8**	17.6	**22.7**

While our results have focused on L_p attacks, we evaluate the tuned CIFAR10 models on two additional non-L_p attacks: RecolorAdv (Laidlaw and Feizi, 2019), and Spatial Transformations (Xiao et al., 2018). We observe in Table 4 that REx provides significantly better robustness against these perturbations than any other baseline. As before, the advantage is particularly more pronounced when starting from an Avg model.

Additionally, we also report results on CIFAR10-C (Hendrycks and Dietterich, 2019) in Table 5. The dataset consists in mimicking several natural corruptions on CIFAR10 images at various strength, which as argued before, can be seen as a non-adversarial analogue of trying both different types, and tunings of adversarial attacks. While this is not an adversarial robustness benchmark, it shows that REx significantly improves robustness of multiperturbation defenses to non-adversarial shifts it is used on, in spite of what REx models' lower in-distribution clean accuracy on CIFAR10 may have suggested in Table 3.

Table 5. Average accuracy of tuned CIFAR10 models on CIFAR10-C corruptions.

	Defenses					
	None	P_∞	Avg	Avg+REx	MSD	MSD+REx
Average	21.8	52.3	26.5	48.2	42.1	51.2

We invite readers interested in more results or details to consult Apdx C.2.

Key observations 3 (ResNet18 on CIFAR10):

- REx improves the performance of all baselines on CIFAR10 with a ResNet18, from 16.9% with the best baseline to 23.5% accuracy against an ensemble of L_p attacks, by sacrificing a little robustness against the weakest individual attacks.
- Multi-perturbation defenses only achieve higher P_∞ and worst-case performance than P_∞ adversarial training when they are used in conjunction with REx.
- REx also provides better robustness on some common non-L_p attacks, and significantly improves robustness of baselines it is used on against CIFAR-C's non-adversarial shifts.

6.5. Conclusion

An attacker seeking to exploit a machine learning model is liable to use the most successful attack(s) available to them. Thus, defenses against adversarial examples should ideally provide robustness against any reasonable attack, including novel attacks. In particular, the worst-case robustness against the set of available attacks is most reflective of the performance achieved against a dedicated and sophisticated adversary.

We achieve state-of-the-art worst-case robustness by applying the domain generalisation technique of V-REx (Krueger et al., 2021), which seeks to equalise performance across attacks used at training time. Our approach is simple, practical, and effective. It produces consistent performance improvements over baselines across different datasets, architectures, training attacks, test attack types and tunings. One limitation, as often in adversarial machine learning, is that our results make no *guarantees* about attacks that were not used in the evaluation. Another limitation lies in the slight loss of accuracy on the unperturbed data, albeit we believe the improvements in adversarial and non-adversarial robustness and promising research directions are significant enough to be of interest to the community.

Indeed, our work demonstrates the potential of applying domain generalisation approaches to adversarial robustness. Future work could investigate other domain generalisation methods, such as Distributionally Robust Optimisation (DRO) (Sagawa et al., 2019) or Invariant Risk Minimisation (IRM) (Arjovsky et al., 2019), and evaluate their effectiveness against more and newer attacks.

Our results are particularly interesting in light of REx's potential failure to improve performance over well-tuned baselines on non-adversarial domain generalisation benchmarks (Gulrajani and Lopez-Paz, 2020).

Acknowledgements

Ioannis Mitliagkas acknowledges support by an NSERC Discovery grant (RGPIN-2019-06512), a Samsung grant and a Canada CIFAR AI chair. Irina Rish acknowledges the support from Canada CIFAR AI Chair Program and from the Canada Excellence Research Chairs Program. Pouya

Bashivan is supported by Healthy-Brains-Healthy-Lives start-up supplement grant, William Dawson Scholar Award, and NSERC grants DGECR-2021-00300 and RGPIN-2021-03035.

We thank Shoaib Ahmed Siddiqui, Jonathan Uesato, Cassidy Laidlaw, Pratyush Maini and Sven Gowal for enriching discussions that helped us refine the paper. We also thank reviewers of ICLR 2023 and ICML 2023, along with attendees of the New Frontiers in Adversarial Machine Learning workshop for their valuable feedback.

Chapter 7

Prologue to Third Article

7.1. Article Details

Simple and Scalable Strategies to Continually Pre-train Large Language Models. Adam Ibrahim*, Benjamin Thérien*, Kshitij Gupta*, Mats L. Richter, Quentin Anthony, Timothée, Eugene Belilovsky, Irina Rish. Published in *Transactions on Machine Learning Research* (TMLR).
* Equal contribution.

7.2. Contributions

Irina had obtained a significant compute grant through the US Department of Energy INCITE 2023 allocation. The project was the fruit of extensive brainstorming between Benjamin, Kshitij, Mats, Timothée, Irina, and Adam. Quentin did the preliminary groundwork to set up and run the GPT-NeoX codebase on Summit, with help from Mats, and later, Adam. Quentin optimised the configuration of the models. Kshitij handled the initial data processing (tokenisation and splits) for an ICML 2023 workshop submission of a preliminary version of the work (Gupta et al., 2023a) which used RedPajama instead of SlimPajama as the "new" dataset. Benjamin reprocessed the data for the experiments of the camera-ready version of the workshop paper which used SlimPajama instead of RedPajama. Benjamin wrote the script to convert the Pythia models from HuggingFace to Megatron format, with Adam helping to debug it. For the initial workshop version of the experiments, Benjamin and Kshitij trained the smaller models, and Benjamin re-trained all models for an updated camera ready version as the second dataset was changed to SlimPajama. In the meanwhile, Adam debugged the evaluation pipeline with EleutherAI's help. After it was decided to start training on Pile ourselves instead of using Pythia models and continuing to train them, all models were retrained. Adam (10B union, all German, domain incremental replay, and pile to pile models) and Benjamin (other 10B, all replay, and 405M union models) ran most of the experiments, with Mats (replication of workshop paper models) and Kshitij (infinite LR models and domain incremental models) training the remainder. Adam fixed transient issues causing node failure and

crashes, and modified the data pipeline to support replay, which was pushed to the GPT-NeoX GitHub repository. Benjamin ran evaluations, and handled plots and the majority of the writing of the paper*. In the meanwhile, Adam helped with most of the remaining writing, fixed code-support and other HPC issues with Quentin on the new 2024 "SummitPlus" allocation, and finished training a dozen incomplete or missing 405M language models. Kshitij helped with the writing.

7.3. Context

The INCITE grant allocated to Irina was aimed at enabling open-source foundation models. Pretraining models with billions of parameters had mostly been possible with resources only available to the industry. After the codebase became functional on Summit, Together AI and Ontocord proposed to collaborate using our code and compute to train models on their new RedPajama dataset. The model was released at two sizes as RedPajama-INCITE. While the initial goal of the project was to train on a long sequence of different tasks, the projected refocused after noticing the demand for better learning rate schedules and a better strategy than retraining from scratch as multiple rounds of finetuning degrade performance of production models such as GPT (Chen et al., 2023). For the open-source community, our work is a step towards enabling continual pretraining of existing open-weight models such as (Azerbayev et al., 2023), which significantly reduces training costs of LLMs. Finally, this work informs strategies for curriculum learning – a form of continual learning – at a scale where experimentation is prohibitively expensive.

*Note that as a consequence, Chapter 8 will use American English spellings and "distribution shifts" instead of "distributional shifts".

Chapter 8

Simple and Scalable Strategies to Continually Pre-train Large Language Models

8.1. Introduction

Over the past few years, large pre-trained models have enabled massive performance improvements in language modeling (Brown et al., 2020; Zhao et al., 2023), visual understanding (Radford et al., 2021; Alayrac et al., 2022; Kirillov et al., 2023), text-to-image generation (Rombach et al., 2022; Pernias et al., 2024), and text-to-video generation (Brooks et al., 2024)—to name a few. Large language models (LLMs) are at the center of all these improvements, providing an intuitive means for humans to interface with machine learning algorithms through language.

While LLMs are the cornerstone of current generative AI technology, they are expensive to train and keep up to date. However, as new and higher-quality datasets continue to become available (Gao et al., 2020; Soboleva et al., 2023; Computer, 2023; Soldaini et al., 2024), organizations will need to update their models to stay abreast of the competition. Currently, LLMs are re-trained on a combination of old and newly collected data. Existing works aim to reduce these training costs by enabling low-cost hyperparameter optimization (Yang et al., 2022) or providing guidelines for maximizing performance under a given compute budget (Hoffmann et al., 2022). However, these works assume that models will be *trained from random initialization*, raising the following question: Should practitioners always combine existing datasets and *train from random initialization* to obtain the best performance? Doing so for every update of the models quickly becomes expensive.

To avoid complete re-training, we explore simple and scalable continual learning strategies for continuing to pre-train LLMs (up to 10B parameters) on large amounts of new data (200B+ tokens). We refer to our setting as "continual pre-training" and highlight that it is *distinct* from existing settings in the literature (Gururangan et al., 2020; Ke et al., 2022; Scialom et al., 2022; Xie et al., 2023) due to the large amount of incoming data we consider. In this work, we do not intend to improve on the performance of models trained from a random initialization on all of the

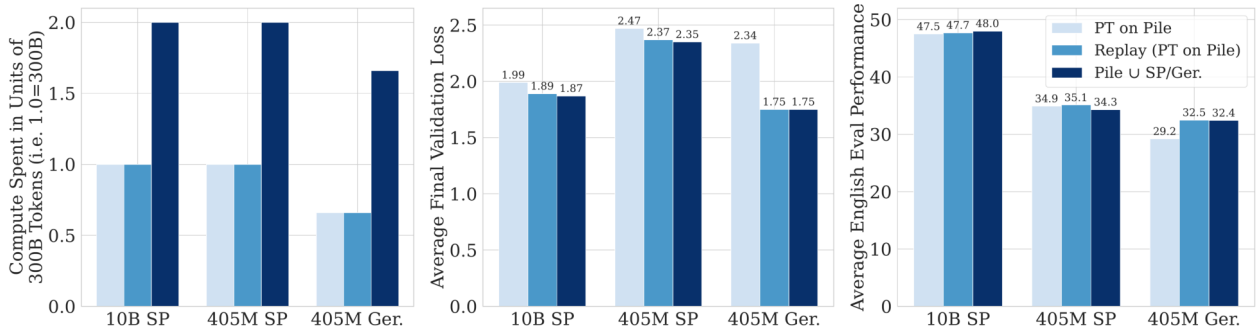
available data. Instead, we consider models trained on the union of existing datasets as baselines whose performance we seek to match using a combination of continual learning strategies at scale.

Naively continuing to train the model on new data, however, tends to lead to performance far below re-training on all available data, often due to 1) poor adaptation (failure to optimize the new dataset) or 2) catastrophic forgetting (significant capability loss on the previous dataset). Firstly, the question of adaptation is central to our setting as training on large datasets is costly. One would presumably not choose to spend considerable computational resources training on a new dataset only to minimally adapt to it. However, most performant open-source LLMs (Touvron et al., 2023a,b; Jiang et al., 2023; Gemma Team et al., 2024) decay their learning rate to a small value by the end of training. We hypothesize, therefore, that the learning rate must be re-increased and re-decayed to improve adaptation per compute spent when training on a new dataset. We note that this has not been thoroughly studied in the continual learning literature. Secondly, catastrophic forgetting is a key difficulty to overcome if one is to realize the full potential of continual pre-training. Adapting to hundreds of billions of new tokens is important, but it must not come at the cost of erasing most existing knowledge in the LLM. Recent work (Scialom et al., 2022) shows, in an LLM fine-tuning setting, that replaying previous data (as little as 1%) is sufficient to mitigate forgetting to a large extent. While continually pre-training on large amounts of new data will almost surely lead to more forgetting than fine-tuning, we hypothesize that an appropriate amount of replay could mitigate forgetting—even in our setting. Moreover, recent works show that pre-training (Cossu et al., 2022; Ramasesh et al., 2022; Mehta et al., 2023) and increasing model size (Mirzadeh et al., 2022) both help to reduce the effects of forgetting. We, therefore, expect the trend of increasing language model capacity and pre-training dataset size in tandem (Kaplan et al., 2020; Hoffmann et al., 2022; Touvron et al., 2023b) will yield models increasingly capable of continual learning (Scialom et al., 2022), suggesting that our experimental results should only improve with models scale.

Given the great potential for continual learning to considerably reduce costs associated with re-training models and the potential for LLMs to be strong continual learners, we ask ourselves the following question: *when simple and scalable continual learning techniques are applied, what is the performance difference between continually pre-trained LLMs relative to LLMs pre-trained from random initialization on the union of all data?* To answer this question, we conduct a large-scale empirical study of continual learning techniques for LLM pre-training. Our empirical evaluation spans large (10B parameters) and small (405M parameters) decoder-only transformer models as well as weak (English $\rightarrow$ English) and stronger (English $\rightarrow$ German) distribution shifts. Our main contributions can be summarized as follows:

(1) We establish the effect of learning rate re-warming and re-decaying for decoder-only transformer-based LLMs pre-trained using a cosine schedule, showing that re-warming and re-decaying is necessary for adaptation during continual pre-training.

(2) We establish the effect of replaying previous data while keeping compute constant across two distribution shifts and many replay percentages. We find that, even when updating



(a) Training compute expended to update/re-train the model

(b) Average final validation loss ↓

(c) Average evaluation performance ↑

Figure 10. Continual pre-training decreases computational costs of updating the model while maintaining similar final validation and average evaluation performance. We report results for the Pile ∪ SlimPajama(SP)/German(Ger.) baseline model trained on the union of both datasets which we consider to be an upper bound on performance. We also report performance for two continually pre-trained models. "PT on Pile" starts from a pre-trained Pile checkpoint and only uses learning rate re-warming and re-decaying, while "Replay (PT on Pile)" re-warms the learning rate, re-decays it, and uses 5% replay for SlimPajama and 25% replay for German. We observe that the combination of LR re-warming, re-decaying, and replay allows our continually pre-trained model to attain similar average performance to the baseline model while requiring substantially less compute. We note that this setting assumes that a pre-trained model is available (e.g., via HuggingFace hub or an in-house model designed to be continually pre-trained).

decoder-only transformer-based LLMs on hundreds of billions of new tokens, it is possible to significantly mitigate forgetting with an appropriate amount of replay.

(3) We demonstrate, across two model sizes and distribution shifts, that a simple and scalable combination of LR re-warming, LR re-decaying, and compute-equivalent replay allows continually pre-trained decoder-only transformer-based LLMs to attain similar performance on average to models re-trained on the union of all data while using significantly less compute.

(4) We propose infinite learning rate schedules (schedules allowing smooth transition across datasets) for the continual pre-training of LLMs as a promising way to circumvent optimization difficulties associated with learning rate re-warming.

Our code is available at `https://github.com/EleutherAI/gpt-neox` through pull requests 1194 and 1200. Model checkpoints throughout continual pre-training for most of our models are available at `https://huggingface.co/collections/cerc-aai/continual-pre-training-661f4af4379b82d9617a9401`. A preliminary version of this work was made available as an ICML 2023 workshop paper in (Gupta et al., 2023b).

8.2. Main Findings and Takeaways and Examples our Method's Practicality

Our experimental results assume that continually pre-trained LLMs undergo two or more pre-training phases sequentially. That is, our results apply to situations where a continually pre-trained

LLM is randomly initialized and pre-trained on datasets $\mathcal{D}_0$,$\mathcal{D}_1$, . . . ,$\mathcal{D}_{N-1}$ in sequence where $N \geq 2$ and $\textit{tokens}(\mathcal{D}_i) \geq 100$B. We note that this includes situations where the LLM in question is an open-source model (Touvron et al., 2023a,b; Jiang et al., 2023; Gemma Team et al., 2024) which has already been pre-trained on $\mathcal{D}_0$ and situations where organizations may wish to train an initial LLM with the intention of continually pre-training it on new data. The new data may be similar to the previous data, corresponding to a weak distribution shift (e.g., the latest web-scrape of different domains), or quite different from previous data, corresponding to a strong distribution shift (e.g., data from a completely new language). Our experimental evaluation accounts for these difficulties, finding that appropriately applying LR re-warming, LR re-decaying, and replay is sufficient to match the performance of re-training across weak and strong distribution shifts and two model sizes (see Fig. 10). To make our findings as accessible to the community as possible, we now provide *Rules of thumb* for applying our findings:

Rules of thumb for continual pre-training

Caveat—The following guidelines are written to the best of our *current knowledge*.

Learning rate schedule:

- If the learning rate was cosine-decayed from a large value η_{max} to a small value η_{min} during pre-training on the initial dataset, the following guidelines can help to continually pre-train your model:
  - Re-warming and re-decaying the learning rate from $\mathcal{O}(\eta_{max})$ to $\mathcal{O}(\eta_{min})$ improves adaptation to a new dataset, e.g. compared to continuing from small learning rates $\mathcal{O}(\eta_{min})$.
  - Decreasing the schedule's maximum learning rate can help reduce forgetting, whereas increasing it can improve adaptation.
- Infinite LR schedules are promising alternatives to cosine decay schedules. They transition into a high constant learning rate across tasks, helping prevent optimization-related forgetting by avoiding re-warming the LR between tasks. They also avoid committing to a specific budget of tokens as a final exponential decay can be used to train the model to convergence at any point during training.

Replay:

- We find that even small amounts of replay are good at mitigating forgetting. We recommend experimenting with different replay fractions since relative differences between them appear very early during training. For example, one may experiment with different replay fractions for a limited token budget, using evaluations relevant to their use case, to find a sweet spot between adapting to the new data and mitigating performance loss due to the distribution shift.

Recent works employing our techniques. Two notable recent works (Glorioso et al., 2024; DeepSeek-AI et al., 2024) have successfully applied combinations of the techniques proposed herein to continually pre-train LLMs at scale, providing further evidence of their efficacy. Glorioso et al. (2024) apply LR re-warming, LR re-decaying, and 60% replay in the context of a decay phase over 50B tokens of high-quality data, applied after their initial pre-training phase. The authors observe improvements in their model's performance without suffering from catastrophic forgetting. DeepSeek-AI et al. (2024) select a non-decayed checkpoint from the initial pre-training phase to ensure a smooth LR transition into continual pre-training (e.g., as suggested in Figure 18), use a decay, and use 30% replay of pre-training data, to continually pre-train DeepSeek-V2 (DeepSeek-AI, 2024) on 6T tokens. The resulting model significantly improves its code generation abilities, while retaining most of its natural language generation abilities. Together, these works highlight the generality the techniques we propose herein: applying the appropriate combinations work to continually pre-train LLMs on small and large continual pre-training datasets (e.g., 50B and 6000B, respectively) and for architectures beyond the dense transformer (e.g., hybrid SSM-transformers and sparse Mixture of Experts models, respectively).

8.3. Related Work

8.3.1. Continual learning

Continual learning (CL) approaches aim to learn from an evolving data distribution, adapting to novel data while retaining knowledge gathered through prior training (French, 1999; Rolnick et al., 2019; Caccia et al., 2020; Lesort et al., 2021). The key challenge of continual learning is to avoid forgetting past information, while also adapting to novel information. This trade-off is known as the rigidity-plasticity dilemma (Mermillod et al., 2013b; Ostapenko et al., 2019; Riemer et al., 2019).

CL approaches are convenient even in small-scale settings to avoid re-training from scratch or to bridge the data availability issue (Smith et al., 2021). However, at scale, CL is more than a convenience; it may be necessary to process huge amounts of continually gathered data. The recent increase in training scale, most notably for LLMs (Scao et al., 2022; Brown et al., 2020; Zhao et al., 2023), offers new opportunities for CL to reduce the cost of re-training and increase efficiency for memory, computing, and storage (Prabhu et al., 2023; Aljundi et al., 2019; Harun et al., 2023a; Veniat et al., 2021; Harun et al., 2023b). Just as federated learning can enable the sharing of compute and data between different agents co-located in space (McMahan et al., 2017; Reddi et al., 2021; Douillard et al., 2023; Ryabinin et al., 2021), continual learning allows the sharing of compute and data progressively through time and could be a useful tool for large-scale training.

Recent work shows that optimizers such as SGD and Adam have interesting knowledge retention properties in DNNs that could be beneficial at scale for CL (Lesort et al., 2023) and that just a small amount of replay could be sufficient to boost knowledge accumulation (Scialom et al., 2022). In this

work, we want to benefit from the efficiency of those approaches in the context of large language models pretraining and boost them with the right learning rate scheduling and replay policy.

8.3.2. Pre-training, Model Scale, and Continual Learning

Several existing works evaluate the impact of pre-training and model scale on continual learning. Cossu et al. (2022) investigate pre-training scenarios for language and vision. They find that unsupervised and self-supervised pre-training plays a fundamental role in mitigating forgetting, while supervision hurts performance. Similarly, Mehta et al. (2023) find that pre-trained models forget less than randomly initialized models, due to their weights lying in flatter regions of the loss landscape. They also find that larger models forget less which is connected to the findings of Ramasesh et al. (2022); Mirzadeh et al. (2022). The former finds that pre-trained models forget less as they are scaled up, suggesting that it may be due to the hidden representations growing more orthogonal with scale. The latter finds that wider neural networks forget less compared to their parameter-equivalent deeper counterparts. Hernandez et al. (2021) establish scaling laws for transfer: equations that can predict the performance of a neural network on a new task as a function of its parameter count and pre-training dataset size. The authors find that this positive transfer consistently improves as the parameter count increases. Finally, Scialom et al. (2022) show that autoregressive LLMs have a strong ability to learn continually which they hypothesize is related to their pre-training objective.

8.3.3. Domain Adaptive Continual Pre-training (DACPT)

Existing work considers Domain Adaptive Continual Pre-training (DACPT), a setting where a series of unlabelled domains become available to the LM sequentially and practitioners wish to train on each domain in a self-supervised fashion while retaining performance across each of them. While the objective is similar to our own, we consider general-purpose pre-training datasets that mix many domains as opposed to domain-specific datasets. Ke et al. (2022) assume data from previous domains is not available when training on new domains and develop a new technique for this setting which involves an importance mask of parameters for all previous tasks to prevent forgetting when pre-training with a masked language modeling (MLM) objective. Gururangan et al. (2020) investigated domain and task adaptive pre-training of RoBERTa (also MLM) and contributed a sample selection strategy for efficient continual pre-training. Similarly, Xie et al. (2023) also propose a data selection strategy that reduces the computational cost of continual pre-training (shown for autoregressive LMs). Qin et al. (2023) investigate re-cycling fine-tuned adapter layers of previous base LMs as the initialization of new adapters for adapting continually updated versions of the base LM to specific tasks. Recently, Wu et al. (2024) proposed LLaMA Pro, a method for the continual pre-training of LLMs that enables learning new tasks without forgetting previous

knowledge. However, unlike our work which considers adapting all existing weights, LLaMA Pro requires growing the size of the model for each new update and only adjusting the new weights.

8.3.4. Continual Learning for LMs Applied to Specific Domains

Several related works apply continual pre-training to specific tasks and domains (Sun et al., 2020; Jang et al., 2022a,b; Gong et al., 2022; Zan et al., 2022; Yadav et al., 2023a; Ma et al., 2023; Yang et al., 2024). While these works also utilize continual pre-training techniques, they differ from our work by focusing on particular domains instead of general pre-training techniques, on smaller-scale datasets < 10B tokens with smaller models. The only existing work that approaches our dataset scale is (Gogoulou et al., 2023), which explores continual autoregressive language modeling across English, Danish, Icelandic, and Norwegian datasets (73B each). While they do not use replay they do re-warm and re-decay the learning rate. The only existing work that approaches our model scale is (Yang et al., 2024). They continually pre-train and instruction tune LLaMA2 on small-scale academic plant science data. This concurrent work uses a very similar continual learning setup to the one we propose: replay, LR re-warming, and LR re-decaying. While, unlike our work, they *do not* build a controlled experimental framework to systematically evaluate the validity of these approaches for continual pre-training, it is nice to see further experimental evidence validating our approach.

8.3.5. Learning Rate Schedules

Several studies have examined the impact of different learning rate (LR) schedules on the training stability and final performance of neural networks. Goyal et al. (2018) found that a gradual warm-up of LR early on in training can help overcome optimization challenges, particularly with large mini-batch sizes. Additionally, Popel and Bojar (2018) emphasized the importance of a warm-up stage when training Post-LN Transformers. On the other hand, Xiong et al. (2020) discovered that Pre-LN Transformers are more stable and may not require a warm-up stage. You et al. (2019) explored the role of the LR decay and found that a large initial LR prevents the network from memorizing noisy data, whereas a small LR helps learn complex patterns. Kaplan et al. (2020) explored LR schedules for pre-training Large Language Models (LLMs) and found that schedule choice did not significantly impact performance. Correcting this erroneous finding, Hoffmann et al. (2022) found that the LR schedule does play an important role. Hoffmann et al. (2022) and Rae et al. (2021) established best practices for using a cosine schedule when pre-training LLMs, which have become widely adopted. In contrast, Raffel et al. (2023) and Zhai et al. (2022) explore LR schedules that follow the inverse square root decay for large-scale pre-training. Raffel et al. (2023) utilized an inverse square root decay for training LLMs, allowing flexibility in adjusting the number of training steps. In Zhai et al. (2022), authors use these schedules referred to as "infinite learning rate schedules" to train vision transformers. These schedules enable indefinite training and the

evaluation of multiple training durations in a single run. We note that our proposed infinite learning rate schedules for LLMs (Sec. 8.7.4) are inspired by this idea.

8.4. Background & Methodology

In this section, we provide appropriate background and methodology as it relates to continual pre-training in the context of LLMs.

8.4.1. Linear Warmup and Cosine Decay Schedule

Hoffmann et al. (2022) and Rae et al. (2021) established best practices for using a cosine schedule when pre-training LLMs. Specifically, they recommend starting with a linear warmup phase and decaying the learning rate to $10\times$ its maximum value such that the end of the cosine cycle is set to match the number of tokens. While the linear warmup duration differs, most works have a duration between 0.1% and 0.5% of training steps (Zhao et al., 2023). Given that many popular open-source models (Touvron et al., 2023b,a; Almazrouei et al., 2023) follow this learning rate schedule recipe, it is critical to understand its nuances for continually pre-training such models. The schedule first linearly increases the learning rate over T_{warmup} timesteps, or equivalently until some timestep $t_{ann} = T_{warmup}$:

$$\eta_t = \eta_{max} \cdot \frac{t}{T_{warmup}} \tag{58}$$

where η_t is the value of the learning rate at iteration t, and η_{max} is the maximum learning rate. The schedule then transitions into a cosine annealing phase over T_{ann} timesteps, equivalently until some timestep $t_{end} = T_{ann} + t_{ann}$:

$$\eta_t = \eta_{min} + \frac{(\eta_{max} - \eta_{min})}{2} \cdot \left(\cos\left(\pi \cdot \frac{t - t_{ann}}{t_{end} - t_{ann}} \right) + 1 \right) \tag{59}$$

where η_{max} is the maximum learning rate and η_{min} is the minimum learning rate. Fig. 11 illustrates these two phases.

8.4.2. Compute-equivalent Replay

In many of our experiments, we compare models trained with replay to models trained without it. When making such comparisons, we keep the amount of compute constant for training both models. That is, we correspondingly reduce the number of tokens seen from the new dataset to accommodate the additional tokens seen from the replay buffer. We refer to this use of replay as *compute-equivalent replay*. For instance, suppose datasets $\mathcal{D}_0$ and $\mathcal{D}_1$ each contain 100B tokens. We wish to compare model (a) trained sequentially on $\mathcal{D}_0$ and $\mathcal{D}_1$ to model (b) trained sequentially on $\mathcal{D}_0$ and $\mathcal{D}_1$ with 5% compute equivalent replay. Model (a) will see all tokens from both datasets for a total of 200B unique tokens. Model (b) will see 100B unique tokens of $\mathcal{D}_0$ and 95B unique tokens

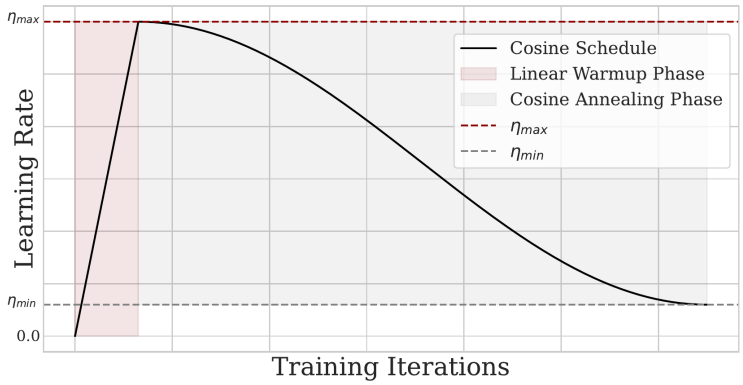


Figure 11. Linear warmup and cosine annealing schedule. For illustration purposes, the schedule uses linear warmup for 10% of training iterations. However, most works have a duration between 0.1% and 0.5% of training steps (Zhao et al., 2023).

of $\mathcal{D}_1$ plus 5B replayed tokens from $\mathcal{D}_0$ for a total of 200B tokens. In this way, both compared models expend the same amount of compute.

For instance, in our settings that span only two datasets $(\mathcal{D}_0, \mathcal{D}_1)$, we use replay of data from $\mathcal{D}_0$ when training on $\mathcal{D}_1$. We replay the data in the order it was seen when pretraining on $\mathcal{D}_0$, as we did not observe noticeable differences when reshuffling the replay data in preliminary experiments. The use of methods for selecting replay samples is left as future work. We refer to models using replay as "$\mathcal{D}_1$ $x\%$ Replay", where x is the percentage of data in each training batch that comes from $\mathcal{D}_0$. Conversely, $(100\% - x)\%$ of the samples in each training batch will be sampled from $\mathcal{D}_1$. When comparing models trained with replay to other configurations, we ensure that the compute is *equivalent* by reducing the number of $\mathcal{D}_1$ tokens to accommodate replay tokens from $\mathcal{D}_0$.

8.5. Experimental Setup

To empirically evaluate the effectiveness of continually pre-training LLMs in comparison to training LLMs from a random initialization, we select recent pre-training datasets from the literature, outline practical continual pre-training settings for investigation, and select several baselines to compare with our proposed techniques. Our goal is to fairly compare our continual pre-training techniques to baselines in a controlled setting. We *do not* seek to obtain state-of-the-art performance or compare with models out of the scope of this paper.

8.5.1. Datasets

We use three datasets for training and validation: SlimPajama (Soboleva et al., 2023), German CommonCrawl (Laippala et al., 2022), and Pile (Gao et al., 2020). For all datasets, use the same tokenizer as Black et al. (2022) trained specifically on the Pile. To create our training set for SlimPajama, we randomly sub-sample the dataset (606B Total Tokens) to form a ∼299B token

subset (see Table 6*) that is of comparable size to Pile. We also further sub-sample this SlimPajama subset to create three $\sim$ 100B token splits of the dataset (see Sec. 8.7.4 for details). For each of these datasets, we follow standard practice in LLM pre-training and select sampling percentages proportionally to the amount of data available in each domain such that one pass over the dataset does not repeat samples from any domain. To create the SlimPajama validation set we simply tokenize the default validation set that has been extensively deduplicated (Soboleva et al., 2023). To create the German training and validation sets, we split and tokenized the German Common Crawl scrape, available as part of the Oscar Dataset (Laippala et al., 2022), into a 195.43B token training set and a 982.6M token validation set. The Pile dataset comes pre-shuffled and mixed, we simply used the default training and validation sets. The training set is $\sim$ 330B tokens total, though in our experiments we only train on a 300B token subset.

Table 6. Domain sizes of the 300B token training set of SlimPajama. We sub-sampled the SlimPajama dataset (606B total tokens) into a 300B token split to make it of comparable size to Pile. We report the size of the subsampled domains that make up SlimPajama and the sampling percentage used at training time (e.g., the percentage of samples in each batch that come from a certain domain).

Dataset	Size (Tokens)	Sampling (%)
Wikipedia	11.96B	4.00
Book	12.58B	4.20
C4	79.87B	26.69
Stack Exchange	10.09B	3.37
GitHub	15.63B	5.22
Common Crawl	155.89B	52.09
Arxiv	13.25B	4.43
Total	299.28B	100.00

8.5.2. Continual Learning Settings

We consider three realistic continual pre-training settings in the main body and provide results for a third which we believe is less warranted in the appendix. Each setting was carefully selected to expose different challenges and strengths of continual pre-training. Our setups assume that continually pre-trained LLMs undergo two or more pre-training phases sequentially. At the start of each phase, we reset the optimizer states, since optimizer states may not always be available, e.g. when using open-weight models from HuggingFace. That is, our results apply to situations where a continually pre-trained LLM is randomly initialized and pre-trained on datasets $\mathcal{D}_0$,$\mathcal{D}_1$,...,$\mathcal{D}_{N-1}$ in sequence where $N \geq 2$. For the realistic settings we consider *tokens*$(\mathcal{D}_i) \geq$ 100B. In each case, we consider the following natural baselines:

- A model trained from random initialization on the union of all datasets i.e. $\bigcup_{i=0}^{N-1} \mathcal{D}_i$, and

*We refer readers to the Pile paper (Gao et al., 2020) for its composition.

- A model trained from random initialization on individual dataset $\mathcal{D}_i$, $0 \leq i \leq N$.

$N = 2$ **settings –** Here we assume a model is available (e.g. via hugging face or pre-trained in-house) that has been pre-trained for autoregressive language modeling on a dataset ($\mathcal{D}_0$) using a linear warmup and cosine decay LR schedule. We also assume that the schedule follows existing conventions in the literature (e.g. decaying to the token budget; see Sec. 8.4 for details) as it is the case for most performant pre-trained LLMs (Rae et al., 2021; Hoffmann et al., 2022; Touvron et al., 2023a,b). Given a model pre-trained on $\mathcal{D}_0$, we now assume that a practitioner wants to update this model on a new dataset $\mathcal{D}_1$ using the same self-supervised objective. We consider the following concrete variations of the **two-dataset setting**:

- **Two datasets, weak shift**: In this variation, we consider $\mathcal{D}_0$ to be the Pile (Gao et al., 2020) and $\mathcal{D}_1$ to be pre-training on SlimPajama (Soboleva et al., 2023). SlimPajama is an extensively deduplicated version of RedPajama (Computer, 2023) which is built based on the LLaMA dataset (Touvron et al., 2023a). We consider this to be a weak but realistic distribution shift as both datasets are English-language and contain overlapping domains (CommonCrawl, GitHub, Arxiv, Wikipedia, StackExchange, Book, and C4), but SlimPajama (2023) is a newer dataset than Pile (2020) and is, therefore, likely to have newer data within these overlapping domains. Therefore, despite the potential for significant overlap, we believe this transition is realistic and is likely to be of interest to practitioners wishing to update an LLM on a similar distribution to pre-training (e.g., newly collected data of the same sources with higher quality filtering).
- **Two datasets, stronger shift**: In this variation, we consider $\mathcal{D}_0$ to be pre-training on the Pile (Gao et al., 2020) and $\mathcal{D}_1$ to be pre-training on German Common Crawl. German Common Crawl is a $\sim$ 200B token dataset taken from the Oscar dataset (Laippala et al., 2022). We note that this constitutes a stronger shift given the change of language. This setting is of particular interest for practitioners wishing to augment an LLM with a new natural language, programming language, or specific domain that is notably different in vocabulary from pre-training. We note, however, that as the domain strays farther and farther away from the tokenizer's training corpus, the tokenizer may become a key bottleneck to performance. We leave the treatment of the tokenizer to future work.

$N > 2$ **settings –** We also consider the following settings with more dataset transitions to investigate how well the methods considered scale with more datasets:

- **Three datasets, no shift** : We consider an $N = 3$ setting, where $\mathcal{D}_0$,$\mathcal{D}_1$,$\mathcal{D}_2$ are each district 100B token splits of SlimPajama. This setting is primarily used to evaluate the ability of our techniques to scale to many future updates and to assess the performance of our proposed infinite learning rate schedules.
- **Domain incremental continual pre-training**: This setting considers consuming the tokens of SlimPajama sequentially ordered by domain. That is, we train on a sequence of N future datasets $\{\mathcal{D}_0,\mathcal{D}_1,\ldots,\mathcal{D}_{N-1}\}$ each of is a distinct domain of SlimPajama 300B. We note that

this is similar to DACPT (Ke et al., 2022), however, we consider much larger datasets for each domain. This setting is particularly challenging due to the distribution shift experience at the transition between each domain. While it is certainly interesting, we believe it is unnecessarily difficult compare to mixing the SlimPajama data before training on it. The poor results in this setting (Sec. D.1.1 of the appendix) suggest that general-purpose LLMs should be continually pre-trained on a mixture of domains if possible, not updated per domain.

8.5.3. Training Setup

Using GPT-NeoX (Andonian et al., 2021) based on Megatron-DeepSpeed (Shoeybi et al., 2019; Microsoft, 2020), we train autoregressive decoder-only transformers with a causal language modeling objective. The models use Pre-LN. Each model is trained using the same tokenizer as Black et al. (2022), which was trained exclusively on the Pile via the BPE algorithm (Sennrich et al., 2016). For all models, we train with the AdamW optimizer (Loshchilov and Hutter, 2019) using a batch size of 1104 and a sequence length of 2048. An epoch of training approximately corresponds to $132{,}366$ total training steps. As mentioned in the previous section, we reset the optimizer states between datasets. We consider two model sizes 405M and 9.6B parameters (referred to as 10B in this work) including embeddings. We train the smaller models using data parallelism across 46 6-GPU nodes using a micro-batch size of 4. The larger model is trained using tensor parallelism(Shoeybi et al., 2020) spanning six GPUs within a node and pipeline parallelism(Huang et al., 2019) spanning four nodes; that is, each model replica spans 24 GPUs across four nodes. We train this model on 276 nodes using gradient accumulation of 4 steps. Each model uses optimizer sharding via ZeRO-1 (Rajbhandari et al., 2020), activation checkpointing (Chen et al., 2016), activation partitioning across tensor parallel ranks, and mixed precision FP16/FP32 to reduce GPU memory consumption and fully utilize NVIDIA tensor cores during training. We provided an extended description of all hyperparameters in the appendix (Table. 26).

8.5.4. German and English LM Evaluation Benchmark

We measure performance on a wide variety of downstream tasks, which can be broadly categorized as follows.

English Benchmarks

- **Commonsense Reasoning (0-shot):** HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2019), PIQA (Bisk et al., 2019), OpenBookQA (Mihaylov et al., 2018), ARC-Easy, ARC-Challenge (Clark et al., 2018)
- **World Knowledge (5-shot):** NaturalQuestions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017)
- **Reading Comprehension (0-shot):** BoolQ (Clark et al., 2019)

- **Math:** MathQA (Amini et al., 2019)
- **Popular Aggregated Results:** MMLU (5-shot) (Hendrycks et al., 2021)

German Benchmarks from (Plüster, 2023), which translated their English counterparts using GPT 3.5 API

- **Commonsense Reasoning (0-shot):** HellaSwag-DE (Zellers et al., 2019), ARC-Challenge-DE (Clark et al., 2018)
- **World Knowledge (5-shot):** TriviaQA-DE (Joshi et al., 2017)
- **Popular Aggregated Results:** MMLU-DE (5-shot) (Hendrycks et al., 2021)

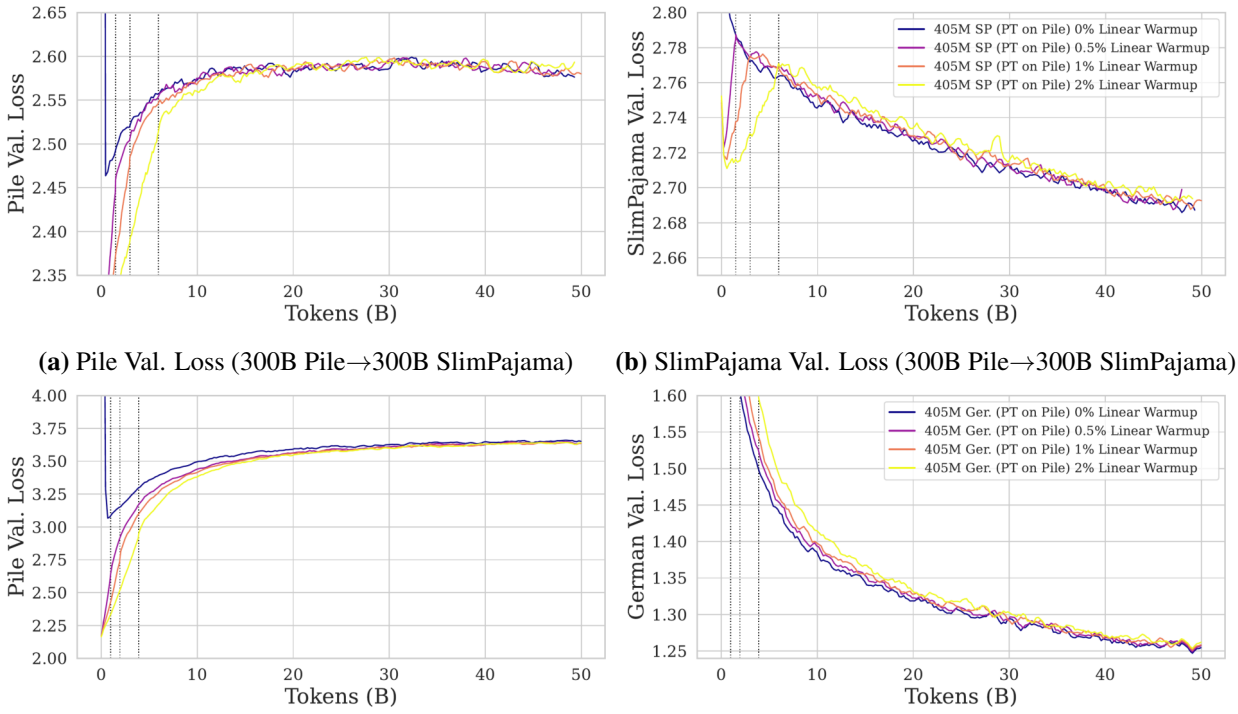
8.6. Results

We focus on continual pre-training when incoming datasets are large (200B tokens+). In such settings, training is expensive, thus, it is critical to efficiently adapt to the large amount of incoming data. However, most performant LLMs (Rae et al., 2021; Hoffmann et al., 2022; Zhao et al., 2023; Touvron et al., 2023b,a) are trained with a linear warmup and cosine decay schedule with a relatively low minimum learning rate. We hypothesize that **re-warming** this learning rate to a relatively high value and subsequently re-decaying it is needed to efficiently adapt to the new dataset. To this end, in section 8.6.1 we study the effect of linear warmup duration, re-warming the LR, re-decaying the LR, and maximum learning rate magnitude on adaptation and forgetting. Finding that re-warming and re-decaying increases both adaptation and forgetting, in section 8.6.2 we investigate whether replay can help mitigate forgetting when the learning rate is re-warmed and re-decayed. Subsections 8.6.3 and 8.6.4 combine the strategies studied in the previous two sections and report their performance relative to baselines for weak and strong distribution shifts and at large model scale. Finally, in section 8.7, we illustrate LR re-warming can cause unwanted forgetting, introduce infinite learning rate schedules as a promising way to circumvent it, and compare these schedules to baselines.

8.6.1. Learning Rate Schedule

Given the influence that the learning rate can have on adaptation and the low final LR values of prominent LLMs (Rae et al., 2021; Hoffmann et al., 2022; Zhao et al., 2023; Touvron et al., 2023b,a), we hypothesize that the LR should be re-warmed and re-decayed to promote adaptation during continual pre-training. In this section, we investigate the effect of linear warmup duration, re-warming the LR, re-decaying the LR, and the magnitude of the η_{max} when continuing to pre-train. Specifically, we evaluate their respective effects in the **two-dataset weak shift** setting (300B Pile $\rightarrow$ 300B SlimPajama) and the **two-dataset stronger shift** setting (300B Pile $\rightarrow$ 300B SlimPajama). Notably, the model trained on $\mathcal{D}_0$ (300B tokens of Pile) follow a linear warmup and cosine decay schedule[†], simulating many common open-source pre-trained LLMs.

[†]For all cosine decays in this paper, unless otherwise specified, we fit the cosine annealing phase to the token budget, set the linear warmup duration (T_{warmup}) to 1% of training iterations, and set $\eta_{min} = 0.1 \cdot \eta_{max}$



(a) Pile Val. Loss (300B Pile→300B SlimPajama)

(b) SlimPajama Val. Loss (300B Pile→300B SlimPajama)

(c) Pile Val. Loss (300B Pile→200B German)

(d) German Val. Loss (300B Pile→200B German)

Figure 12. The effect of linear warmup for weak and strong distribution shifts. (a),(b) and (c),(d) have the same legends respectively, shown in the right figures. We train 405M parameters models following a linear warmup and cosine decay schedule with varying linear warmup durations: 0%,0.5%,1%, and 2% of training iterations. Each learning rate schedule decays to $0.1\eta_{max}$ by the end of training based on the size of the dataset. We report results for the first 50B tokens of training. In the settings explored, we observe that the duration of the warm-up phase does not appear to be impactful when continuing to pre-train.

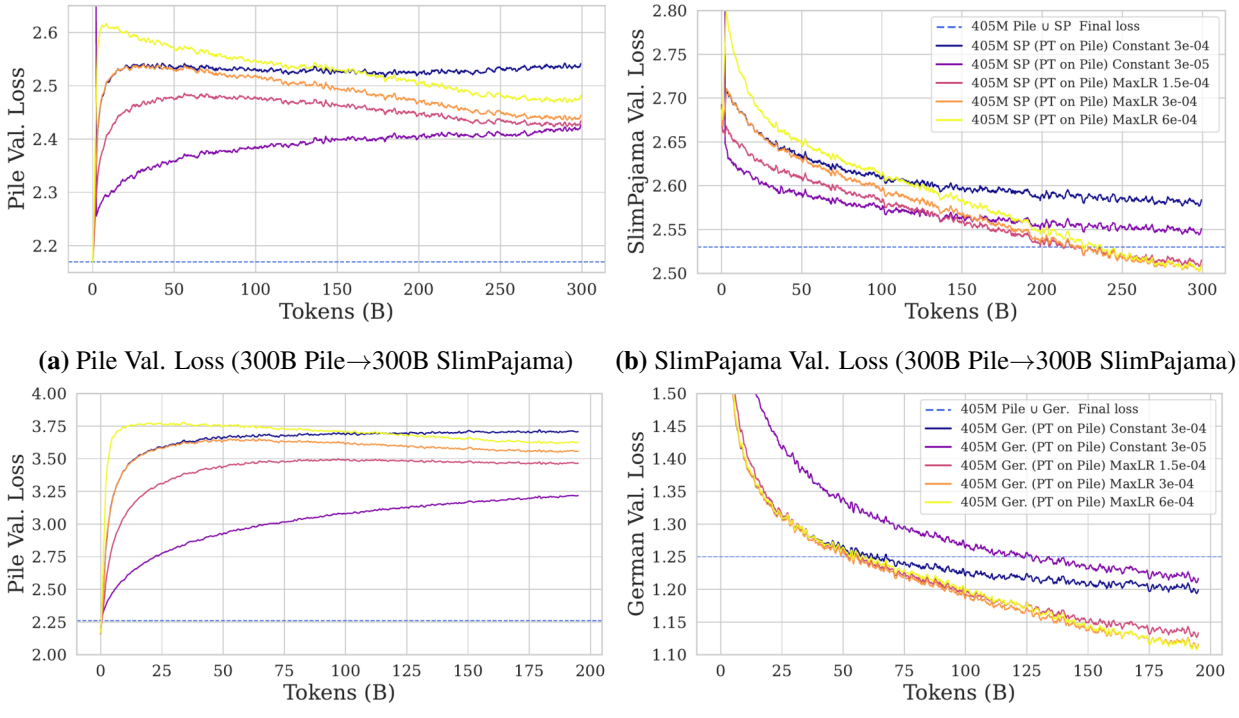
8.6.1.1. The Effect of Linear Warmup for Weak and Strong Distribution Shifts. We first investigate the effect of linear warm-up duration on forgetting and adaptation in the **two datasets, weak shift** and **two datasets, stronger shift** settings (see Sec. 8.5.2 for details). The models are pre-trained on 300B tokens of Pile (Gao et al., 2020) ($\mathcal{D}_0$). We continue to pre-train the models on SlimPajama (weak shift) and German Common Crawl (stronger shift) for the first 50B tokens of training. We re-warm and re-decay the learning rate using a cosine learning rate schedule set to reach its minimal value ($\eta_{min} = 0.1 \cdot \eta_{max}$) at 300B and 200B tokens, respectively. We consider warming up the learning rate for 0.5%, 1%, and 2% of $\mathcal{D}_1$'s total training iterations (132366 and 86000 iterations, respectively). Since the decay happens over the remaining budget of iterations (so resp. $99.5\%, 99\%$ and 98% of the total iterations), note that this implies that the decay phase of longer warmups happens marginally faster. Additionally, we train a model with no linear warm-up (0%) that immediately decays the LR from η_{max} . All experiments are conducted on a 405M parameter model.

Figure 12 reports the validation losses for $\mathcal{D}_0$ and $\mathcal{D}_1$ for all models throughout the first 50B tokens of continued pre-training on $\mathcal{D}_1$. The top row reports results for the weak distribution shift, while the bottom row reports results for the stronger distribution shift. Across both distribution

shifts, we observe that models using shorter linear warmup initially forget and adapt faster than their longer warmup counterparts. This happens because they increase the LR faster which leads to faster forgetting and adaptation. In particular, the model without any warmup adapts and forgets the fastest—even undergoing an initial chaotic phase (as seen in the continual learning literature (De Lange et al., 2022)). Indeed, coupled with noisy gradients due to adapting to a new distribution and the resetting of optimizer states, its large initial learning rate causes a transient spike in validation loss across both shifts. In all scenarios, however, these initial differences diminish throughout training, leaving all models with relatively similar forgetting and adaptation after 50B tokens.

Thus, in the settings explored, the duration of the linear warm-up phase does not appear to affect forgetting or adaptation as measured by the validation loss when continuing to pre-train, although it can prevent initial transient spikes in the loss.

With this in mind, we set a linear warmup duration of 1% of training iterations for all subsequent experiments.



(a) Pile Val. Loss (300B Pile→300B SlimPajama)

(b) SlimPajama Val. Loss (300B Pile→300B SlimPajama)

(c) Pile Val. Loss (300B Pile→200B German)

(d) German Val. Loss (300B Pile→200B German)

Figure 13. The effect of re-warming and re-decaying the learning rate on adaptation and forgetting. We consider two constant baselines and three models that re-warm and re-decay. One baseline continues training from η_{min} of pre-training ($3 \cdot 10^{-4}$) while the other warms up to η_{max} from pre-training ($3 \cdot 10^{-4}$). For the models that re-warm and re-decay we vary $\eta_{max} \in \{1.5 \cdot 10^{-4}, 3 \cdot 10^{-4}, 6 \cdot 10^{-4}\}$. All models except the η_{min} baseline use linear warmup for 1% training iteration. The non-baseline models cosine decay the learning to reach $0.1 \cdot \eta_{max}$ by the end of training. We observe that re-warming and re-decaying the learning rate is needed to best adapt to the new dataset. Small increases or decreases in η_{max} allow to trade-off between more or less adaptation. A stronger distribution shift seems to be a catalyst for both forgetting and adaptation.

8.6.1.2. The effect of re-warming, re-decaying, and varying η_{max} for Weak and Strong Distribution Shifts. We now investigate the benefits of re-warming and re-decaying the learning rate (e.g., following a cosine schedule) for different values of η_{max} . Specifically, we compare these models to two natural baselines: a model that does not re-warm, staying constant at η_{min} ($3 \cdot 10^{-5}$), and a model that re-warms to the pre-training η_{max} ($3 \cdot 10^{-4}$) but does not re-decay. We use the same two two-dataset settings: we first pre-train on the Pile ($\mathcal{D}_0$) for 300B tokens and continually pre-train our model on SlimPajama (weak shift) or German Common Crawl (strong shift) as our $\mathcal{D}_1$ datasets. The continual pre-training is conducted for the full size (300B and 200B tokens, respectively) of the datasets. The models that re-warm and re-decay the LR consider three strategies: re-warming to half the pre-training's η_{max} ($1.5 \cdot 10^{-4}$), re-warming to the same η_{max} as pre-training ($3 \cdot 10^{-4}$), and re-warming to twice the η_{max} of pre-training ($6 \cdot 10^{-4}$). In all cases, the learning rate is cosine-decayed after linear warmup to reach $\eta_{min} = 0.1 \cdot \eta_{max}$ by the end of training. Finally, we consider models trained on $\mathcal{D}_0 \cup \mathcal{D}_1$ as a third baseline (union-trained) to provide an upper bound on performance.

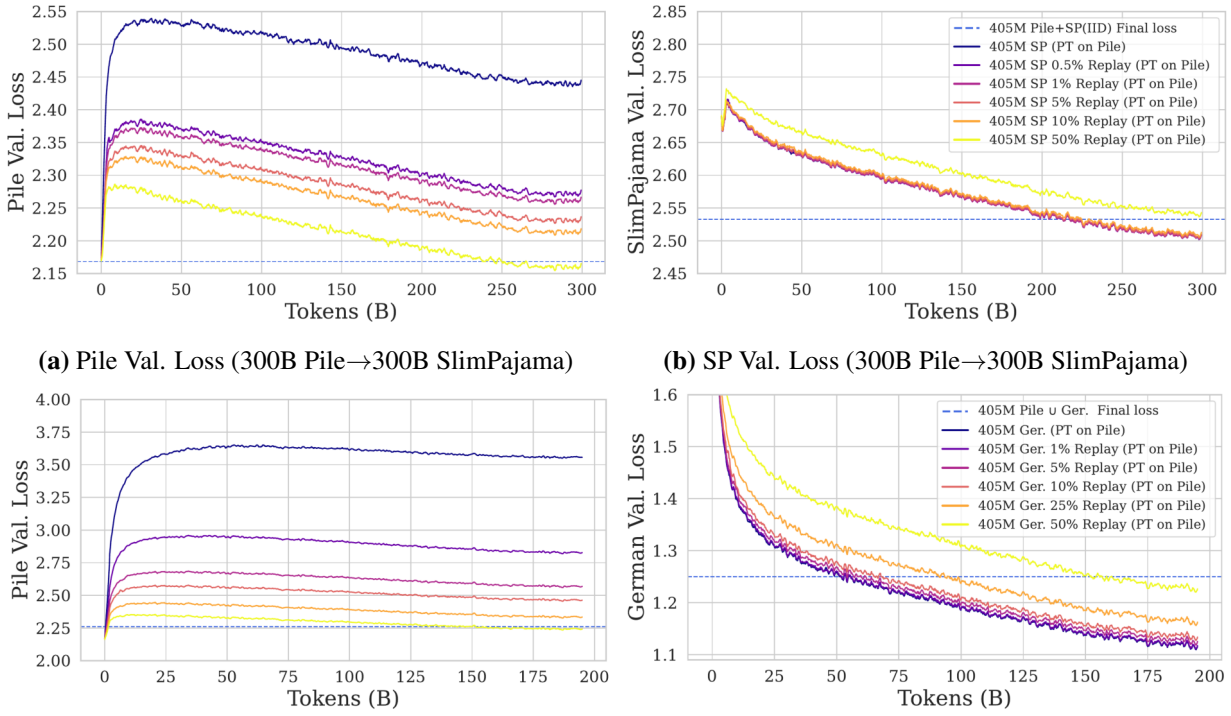
Figure 13 reports validation losses for the $\mathcal{D}_0$ and $\mathcal{D}_1$ datasets throughout the continual pre-training of all models. The top row of plots reports results for the weak distribution shift (300B Pile→300B SP), while the bottom row reports results for the stronger distribution shift (300B Pile→200B Ger.). For both shifts, the constant η_{min} learning rate model achieves the least forgetting on $\mathcal{D}_0$. It also adapts the least on $\mathcal{D}_1$ for the stronger shift, however, for the weak shift it adapts more than the constant η_{max} baseline. When comparing these constant LR baselines to the models that re-warm and re-decay on both shifts considered, we observe that the latter models adapt better to the new dataset by a significant margin for both distribution shifts. This shows that re-warming and re-decaying are necessary to maximize adaptation to the new dataset when continually pre-training LLMs. Among the models that re-warm and re-decay the LR, we observe that varying the learning rate causes small differences in adaptation and forgetting: higher values of η_{max} lead to more forgetting and more adaptation while the opposite is true for lower values. When comparing the constant LR baselines to the union-trained baseline, we observe that the final validation loss for $\mathcal{D}_0$ is significantly higher than the union-trained model's on both distribution shifts. This is also the case for $\mathcal{D}_1$ on the weak distribution shift, but interestingly for the stronger distribution shift, the constant baselines achieve lower $\mathcal{D}_1$ validation loss than the union-trained model. The stronger distribution shift appears to exacerbate the relative forgetting and ability of the models to adapt in the context of continually pretrained LLMs. When comparing models continually pre-trained with re-warming and re-decaying to the union baseline, we note that these models adapt better (lower final validation loss) to $\mathcal{D}_1$ than the union baseline. However, these models experience significant forgetting on $\mathcal{D}_0$, showing the need for replay to make these models competitive with the union baseline.

In summary, continually pre-training LLMs, both re-warming and re-decaying are necessary to maximize adaptation to the new dataset; small increases or decreases in η_{max} allow to trade-off

between more or less adaptation; a stronger distribution shift between $\mathcal{D}_0$ and $\mathcal{D}_1$ exacerbates forgetting and enhances adaptation; and the duration of linear warm-up phase does not appear to be impactful on forgetting or adaptation.

8.6.2. The Effect of Replay

In this subsection, we explore the effect of compute-equivalent replay when continually pre-training models that re-warm and re-decay the learning rate.



(a) Pile Val. Loss (300B Pile→300B SlimPajama)

(b) SP Val. Loss (300B Pile→300B SlimPajama)

(c) Pile Val. Loss (300B Pile→200B German)

(d) German Val. Loss (300B Pile→200B German)

Figure 14. The effect of replay at 405M scale for weak and strong distribution shifts. We report Pile validation loss (left) and SlimPajama/German validation (right top/bottom) during training. Each model is trained from a checkpoint pre-trained on 300B tokens of Pile. The blue dotted line reports the final validation loss for models trained on Pile∪SlimPajama or Pile∪German data, totaling 600B and 500B tokens datasets respectively. We observe that replay significantly reduces forgetting across both shifts, however, the stronger shift requires more replay to mitigate forgetting to the same extent.

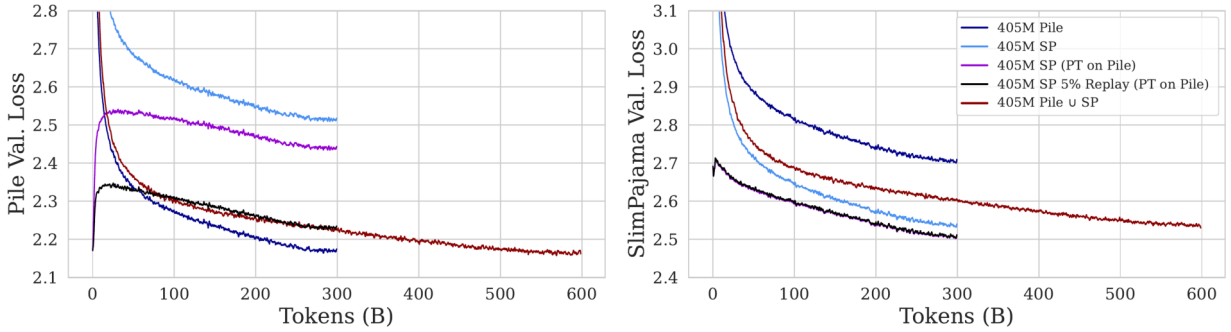
Given the need to mitigate forgetting when re-warming and re-decaying, we move on to investigate the effects of replay in our weak and strong-shift continued pre-training scenarios. Specifically, we use compete equivalent replay (see Sec. 8.4.2 for details) where replay tokens from $\mathcal{D}_0$ are added at the cost of removing the equivalent number of $\mathcal{D}_1$ tokens from the budget. Following the same two dataset settings, the model is pre-trained on $\mathcal{D}_0$ (Pile) for 300B tokens. This is followed by continual pre-training on a SlimPajama (weak shift) or German Common Crawl (strong shift). For more details regarding the setup, please see Section 8.5.2. Our continued pre-training is conducted for the full size of the respective datasets, which is 300B tokens for SlimPajama (weak shift) and 200B

Table 7. Final loss of English-only 405M parameter models trained with varying amounts of replay. The loss is averaged over the last 100 iterations of training sampled at intervals of 10 iterations. The standard error for these measurements was computed but is not reported as it was < 0.001 for all models. We observe that models using more replay achieve a better adaptation-forgetting trade-off (AVG Loss). Interestingly, the model using 50% replay archives nearly identical loss values while seeing 150B fewer tokens on SlimPajama.

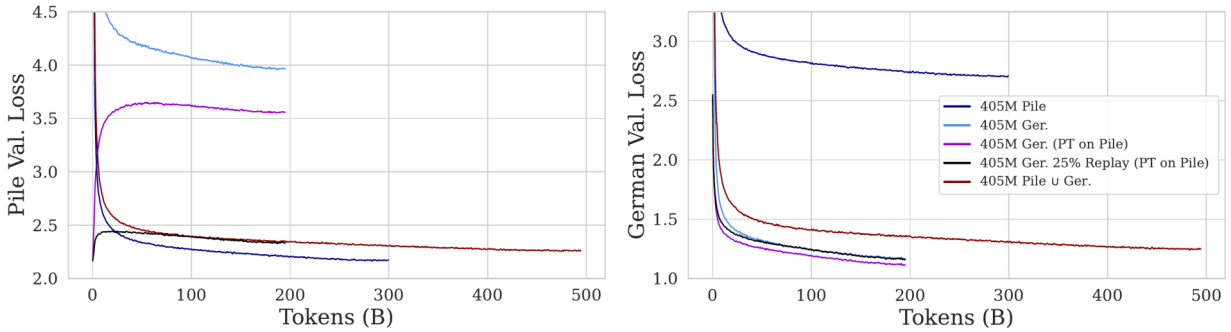
Training Tokens	Validation Loss		
	$\mathcal{D}_0$ Pile	$\mathcal{D}_1$ SlimPajama/German	AVG
300B Pile → 300B SP	2.44	2.50	2.47
300B Pile → 300B SP (0.5% Replay)	2.27	2.50	2.39
300B Pile → 300B SP (1% Replay)	2.26	2.50	2.38
300B Pile → 300B SP (5% Replay)	2.23	2.51	2.37
300B Pile → 300B SP (10% Replay)	2.21	2.51	2.36
300B Pile → 300B SP (50% Replay)	2.16	2.54	**2.35**
600B Pile ∪ SP	2.17	2.53	**2.35**
300B Pile → 200B Ger.	3.56	1.11	2.34
300B Pile → 200B Ger. (1% Replay)	2.83	1.12	1.97
300B Pile → 200B Ger. (5% Replay)	2.57	1.12	1.85
300B Pile → 200B Ger. (10% Replay)	2.46	1.13	1.80
300B Pile → 200B Ger. (25% Replay)	2.33	1.16	1.75
300B Pile → 200B Ger. (50% Replay)	2.24	1.22	**1.73**
500B Pile ∪ Ger.	2.26	1.25	1.75

tokens for German Common Crawl (strong shift). We consider 1%, 5%, 10%, and 50% replay for both shifts and add 0.5% and 25% replay runs for the weak and strong distribution shifts respectively. We consider two baselines to put these results into a broader context. The first baseline is a model trained on $\mathcal{D}_1$ without replay. The second baseline model is trained from random initialization on a union of $\mathcal{D}_0$ and $\mathcal{D}_1$ for 600B tokens (SlimPajama) and 500B tokens (German Common Crawl). The latter baseline reflects the practice of fully re-training the model to update it instead of continually pre-training the existing model. All models re-warm and re-decay the learning rate using a cosine decay schedule fit to their token budget with the same η_{max} ($3 \cdot 10^{-4}$) and η_{min} ($3 \cdot 10^{-5}$) values as during pre-training on $\mathcal{D}_0$.

Validation Loss Comparison. The results in Fig. 14 (top and bottom) show the evolution of the validation loss during continual pre-training on the respective $\mathcal{D}_1$ datasets. Table 7 reports the average final validation loss for each of these models. The final loss is averaged over the last 100 iterations of training sampled at intervals of 10 iterations. We consistently observe across both distribution shifts that even the lowest tested replay of 1% significantly reduces forgetting on Pile compared to the no-replay baselines. This effect is more pronounced in the strong-shift scenario due to the larger amount of forgetting in this setting. We observe little impact on downstream performance for 1%, 5%, and 10% replay when compared to the 0% baseline, showing that the forgetting benefits of replay come at little cost in our setting. However, when using an extreme amount of replay (50%), we observe that the model adapts relatively significantly worse to $\mathcal{D}_1$. Interestingly, for both datasets, the 50% replay models attain or surpass the final average validation performance of the baseline training on $\mathcal{D}_1 \cup \mathcal{D}_0$. This is curious as these model have seen 150B (for SlimPajama) and 100B (for German) fewer tokens of $\mathcal{D}_1$ than their respective baselines.



(a) 405M Pile Val. Loss (300B Pile→300B SlimPajama) **(b)** 405M SP Val. Loss (300B Pile→300B SlimPajama)



(c) 405M Pile Val. Loss (300B Pile→200B German) **(d)** 405M German Val. Loss (300B Pile→200B German)

Figure 15. Final loss of 405M parameter models trained on two distribution shifts. Figures (a) and (b) are duplicated from Fig. 16 for convenient comparison. we provided three baselines and two continually pre-trained models. The baselines (light blue, dark blue, and maroon) are trained from random initialization on 300B tokens of SlimPajama, 300B tokens of Pile, and the union of both datasets (600B tokens). The continually pre-trained models (black and violet) start from a checkpoint pre-trained on 300B tokens of Pile (dark blue curve) and use 0% and 5% replay, respectively. We observe that for both distribution shifts, the combination of re-warming the learning rate and using a small percentage of replay helps to strike a balance between forgetting and adaptation. Importantly, we note that the use of replay minimally affects downstream performance compared to the models using 0% replay.

In summary, we find that, when re-warming and re-decaying the LR in a continual pre-training context, replay is a useful tool for reducing forgetting. For both distribution shifts, using an appropriate amount of replay yields similar final validation loss to the $\mathcal{D}_1 \cup \mathcal{D}_0$ baseline. Moreover, for both shifts, the use of replay seems to negligibly affect adaptation to the downstream dataset, showing that reducing forgetting via replay comes at very little cost when continually pre-training LLMs.

8.6.3. Continual Pre-training Final Performance for Weak and Strong Distribution Shifts.

In this subsection, we compare two continually pre-trained 405M parameter models to several baselines in the *two dataset weak shift* (Pile → SlimPajama) and *two dataset strong shift* (Pile → German) settings. Our main goal is to determine how the differences in distribution shift affect final performance.

Continually Pre-trained Models. To ablate the performance of combining LR re-warming and re-decaying with replay, we opt to train one model that exclusively re-warms and re-decays the learning rate and another that combines both techniques. Given results from the previous section showing that many replay percentages obtain similar average validation loss, we select 5% replay for the weak shift setting and 25% replay for the stronger shift setting because these percentages allow us to see more new tokens than their higher replay counterparts (due to compute-equivalent replay) with a similar average final validation loss. For both models, we re-warm to the η_{max} of pre-training ($3 \cdot 10^{-4}$) and re-decay it using a cosine decay schedule set to reach η_{min} by the end of continual pre-training. More hyperparameters are reported in Table 26 of the appendix.
Baselines. We also train several baselines. Two baselines are trained on $\mathcal{D}_0$ and $\mathcal{D}_1$ respectively while the third is trained on the union of each dataset $\mathcal{D}_0 \cup \mathcal{D}_1$. We consider the model trained on $\mathcal{D}_0 \cup \mathcal{D}_1$ to be an upper bound on performance as it represents an expensive full re-training. The baselines trained on individual datasets can be seen as compute-equivalent alternatives to continual pre-training (e.g., one could opt to train a model from random initialization on $\mathcal{D}_1$ instead of continually pre-training it).

Table 8. Final loss of continually pre-trained English-only & English-German models. All models have 405M parameters. The loss is averaged over the last 100 iterations of training sampled at intervals of 10 iterations. The standard error for these measurements was computed but is not reported as it was < 0.001 for all models. We observe that even for starker distribution shifts, the combination of LR warmup and 25% replay helps to match the average performance of the Pile ∪ German model.

Training Tokens	Validation Loss			LM Eval. Acc.	
	$\mathcal{D}_0$ Pile	$\mathcal{D}_1$ German/SP	AVG	English	HellaSwag-DE
300B Pile	2.17	2.70	2.44	33.95	27.09
300B SP	2.51	2.53	2.52	34.11	27.03
300B Pile → 300B SP	2.44	2.50	2.47	34.93	27.43
300B Pile → 300B SP (5% Replay)	2.23	2.51	**2.37**	35.14	27.09
600B Pile ∪ SP	2.17	2.53	**2.35**	34.30	27.36
300B Pile	2.17	2.70	2.44	33.95	27.09
200B German	3.97	1.17	2.57	27.74	29.53
300B Pile → 200B German	3.56	1.11	2.34	29.20	31.23
300B Pile → 200B German (25% Replay)	2.33	1.16	**1.75**	32.48	31.04
500B Pile ∪ German	2.26	1.25	**1.75**	32.43	30.45

8.6.3.1. Final Performance Evaluated by Loss. Figure 15 reports the validation loss during continual pre-training of 405M parameter models for weak (top) and strong (bottom) shifts. Table 8 reports the average (over the last 100 iterations) final loss value for these models. Since the transition from English to German represents a starker distribution shift than Pile to SlimPajama, training on German leads to significantly more forgetting on Pile ($\mathcal{D}_0$) for the continually pre-trained model without replay (0.27 vs 1.39 for weak and strong shifts respectively). However, choosing 25% replay to handle the starker shift significantly reduces the amount of forgetting on Pile, a reduction of 1.23 in terms of final loss. When comparing continually pre-trained models to baselines trained exclusively on $\mathcal{D}_1$, we observe that the continually pre-trained models always have lower validation loss across both distribution shifts. When comparing the continually pre-trained models with the

$\mathcal{D}_0 \cup \mathcal{D}_1$ baselines we find that both models achieve nearly identical (weak shift) or identical (strong shift) average final validation losses. This shows that for strong and weak distribution shifts, a simple and scalable combination of LR re-warming, LR re-decaying, and replay can achieve similar performance to the $\mathcal{D}_0 \cup \mathcal{D}_1$ baseline.

8.6.3.2. Final Performance Evaluated by Zero-shot and Few-shot Results on Popular LM Benchmarks. While final accuracy provides a good measure of performance on the pre-training objective, LLMs' abilities are typically judged by their performance on evaluation tasks. With the caveat that we use base models, that is our models have not been instruction-tuned, fine-tuned, or adapted to human preferences in any way, we present their evaluation on popular benchmarks in this section. Furthermore, we also provide a qualitative evaluation of German-trained models. We refer the reader to Sec. 8.5.4 of the main manuscript and Sec. D.1.6 of the appendix for a more detailed description of the chosen evaluation tasks.
Table 8 reports the average accuracy of each model for our English evaluation tasks and the normalized accuracy for the German HellaSwag evaluation task. We do not report the average German evaluation score as it is not informative due to evaluations having near-random chance accuracy (see Table 24). We observe that English models consistently outperform German models on the English evaluations. However, the strong replay used with the 25% replay German model helps to reduce this gap. English models' English evaluation performance is very similar with a range of 1.19 between the highest and lowest values. We suspect that there is significant noise in the evaluation process for base models of this size and believe that the differences are likely not significant. That being said, the continually pre-trained model with LR re-warming, LR re-decaying, and replay does improve on the $\mathcal{D}_0 \cup \mathcal{D}_1$ model. When evaluating German-trained models on English evaluation tasks, we see consistent improvements for models using more replay. We note that once again the model trained with LR re-warming, LR re-decaying, and replay does improve on the $\mathcal{D}_0 \cup \mathcal{D}_1$ model. Turning to the German HellaSwag results we observe that German models consistently outperform their English counterparts. Among German-trained models, the continually trained models outperform the union-trained model and the model trained exclusively on German. Given the poor performance of German models on all German evaluation tasks except HellaSwag (the same as English models on average), we further investigated their understanding of German by conducting a short qualitative study of model generations. In section D.1.5 of the appendix, we select five German prompts that contain various peculiarities of the German language (see Tab. 21 of the appendix). We then generate a fixed token-length response for each of the models trained German Common Crawl. As a baseline, we also evaluate the model trained only on the Pile. Despite the poor quality of generations at small model scale, we find that there is an observable improvement in the generative quality of German-language outputs from the models trained on German Common Crawl when compared to the Pile baseline, which tends to be systematically off-topic. This suggests that while our German-trained models have learned about the language, the evaluation tasks are too

difficult to pick it up at the 405M parameter scale. Another reason is that the German dataset is smaller than the English datasets considered, and contains only web-scraped data, as opposed to the more sophisticated English datasets used in this work.

In summary, for weak and stronger distribution shifts alike, it is possible to achieve competitive performance to a model trained on $\mathcal{D}_0 \cup \mathcal{D}_1$ by utilizing a simple and scalable combination of LR re-warming, LR re-decaying, and replay. This is true for final validation loss and averaged language model evaluation scores, showing that this powerful combination of simple techniques can equip language models with new knowledge with little compromise to existing knowledge.

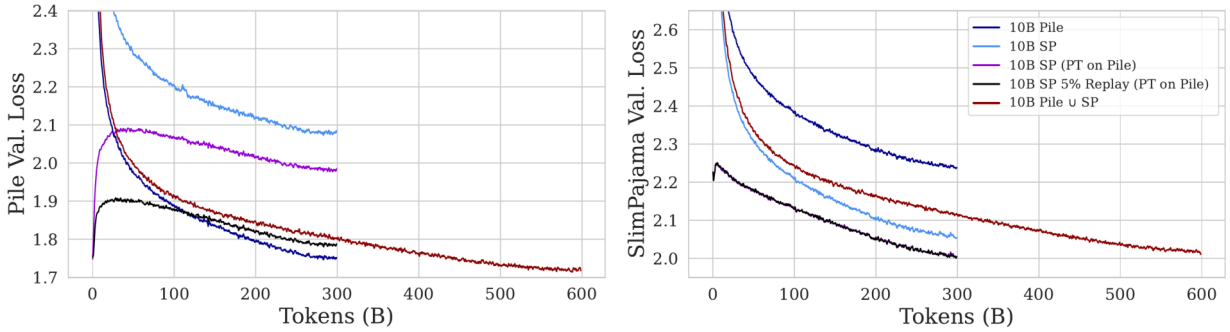
8.6.4. Continual Pre-training Final Performance at Different Model Scales

In this subsection, we establish the effect of increasing parameter count by an order of magnitude on the final performance of continual pre-training. To accomplish this we compare two continually pre-trained models to several baselines at 405M and 10B parameter model sizes in the *two dataset weak shift* (Pile $\rightarrow$ SlimPajama) and *two dataset strong shift* (Pile $\rightarrow$ German) settings.

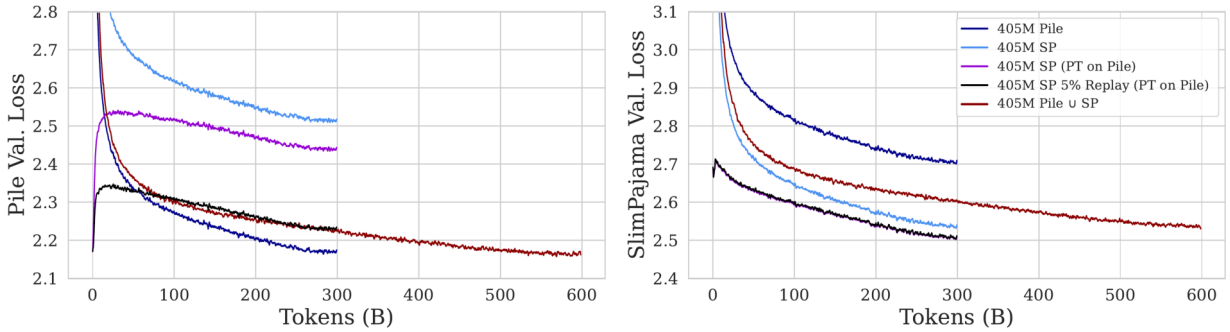
Continually Pre-trained Models. To ablate the performance of combining LR re-warming and re-decaying with replay, we opt to train one model that exclusively re-warms and re-decays the learning rate and another that combines both techniques. Given results from (Sec. 8.6.2) for the weak distribution shifts, showing that many replay percentages obtain similar average validation loss, we select 5% replay for both model scales because these percentages allow us to see more new tokens than their higher replay counterparts (due to compute-equivalent replay) with a similar average final validation loss. For both models, we re-warm to the η_{max} of pre-training ($3 \cdot 10^{-4}$) and re-decay using cosine annealing set to reach η_{min} by the end of continual pre-training. More hyperparameters are reported in Table 26 of the appendix.

Baselines. We also train several baselines. Two baselines are trained on $\mathcal{D}_0$ and $\mathcal{D}_1$ respectively while the third is trained on $\mathcal{D}_0 \cup \mathcal{D}_1$. We consider the model trained on $\mathcal{D}_0 \cup \mathcal{D}_1$ to be an upper bound on performance as it represents an expensive full re-training. The baselines trained on individual datasets can be seen as compute-equivalent alternatives to continual pre-training (e.g., one could opt to train a model from random initialization on $\mathcal{D}_1$ instead of continually pre-training it).

8.6.4.1. Final Performance Evaluated by Loss. Figure 16 reports the validation loss during continual pre-training for 405M and 10B models, while Table 9 reports the average (over the last 100 iterations) final loss value for each model. As expected, we observe that all baselines and continually pre-trained models consistently improve in perplexity on both datasets from increasing parameter count. For the 405M models, we observe that Pile $\cup$ SP achieves identical validation loss on each dataset to the baselines trained individually on them. In contrast, the 10B parameter model trained on Pile $\cup$ SP outperforms the models trained individually on each. We hypothesize that this happens due to larger models having more capacity, thus being capable of learning at a



(a) 10B Pile Validation Loss (300B Pile→300B SlimPajama) **(b)** 10B SP Validation Loss (300B Pile→300B SlimPajama)



(c) 405M Pile Val. Loss (300B Pile→300B SlimPajama) **(d)** 405M SP Val. Loss (300B Pile→300B SlimPajama)

Figure 16. Validation loss during continual pre-training of 10B (top) and 405M (bottom) parameter models. At each model scale we provided three baselines and two continually pre-trained models. The baselines (light blue, dark blue, and maroon) are trained from random initialization on 300B tokens of SlimPajama, 300B tokens of Pile, and the union of both datasets (600B tokens). The continually pre-trained models (black and violet) start from a checkpoint pre-trained on 300B tokens of Pile (dark blue curve) and use 0% and 5% replay, respectively. We observe that for both model sizes, the combination of LR re-warming, LR re-decaying, and using a small percentage of replay helps to strike a balance between forgetting and adaptation. Importantly, we note that the use of replay minimally affects downstream performance compared to the models using 0% replay (black and violet curves overlap in figures (b) and (d)).

Table 9. Final loss of 10B and 405M parameter models. The loss is averaged over the last 100 iterations of training sampled at intervals of 10 iterations. The standard error for these measurements was computed but is not reported as it was < 0.001 for all models. We observe that at both model scales, learning rate re-warming combined with 5% replay approaches the average loss value of joint training.

		Validation Loss		
Model Size	**Training Tokens**	$\mathcal{D}_0$**Pile**	$\mathcal{D}_1$**SlimPajama**	**AVG**
10B	300B Pile	1.75	2.24	1.99
	300B SP	2.08	2.05	2.07
	300B Pile → 300B SP	1.98	2.00	1.99
	300B Pile → 300B SP (5% Replay)	1.79	2.00	**1.89**
	600B Pile ∪ SP	1.72	2.02	**1.87**
405M	300B Pile	2.17	2.70	2.44
	300B SP	2.51	2.53	2.52
	300B Pile → 300B SP	2.44	2.50	2.47
	300B Pile → 300B SP (5% Replay)	2.23	2.51	**2.37**
	600B Pile ∪ SP	2.17	2.53	**2.35**

higher rate for longer. We observe that replaying 5% pile data when continuing to pre-train on SlimPajama reduces forgetting on Pile validation by 0.19 and 0.21 for 10B and 405M parameter

models respectively. The negligible difference in forgetting-reduction from replay despite the order of magnitude difference in parameters between both models suggests that model scale has a limited negative influence on forgetting-reduction from replay. We believe this is because larger models forget less by default. Indeed, the models trained without replay from a pre-trained Pile checkpoint forget 0.23 and 0.27 nats of Pile perplexity for 10B and 405M respectively. While the difference is small, this suggests that larger models forget less, confirming our hypothesis. When comparing the average final validation loss of the models with 5% replay and baselines trained on the union of both datasets, we notice that there is only a difference of 0.02 for both model sizes. This shows that for weak but realistic distribution shifts at two model scales, continual pre-training can achieve similar performance to the expensive re-training baseline.

8.6.4.2. Final Performance Evaluated by Zero-shot and Few-shot Results on Popular LM Benchmarks. While final accuracy provides a good measure of performance on the pre-training objective, LLMs abilities are typically judged by their performance on evaluation tasks. With the caveat that we use base models, that is our models have not been instruction-tuned, fine-tuned, or adapted to human preferences in any way, we present their evaluation on popular benchmarks in this section. We refer the reader to Sec. 8.5.4 of the main manuscript and Sec. D.1.6 of the appendix for a more detailed description of the chosen evaluation tasks.

Table 10. All Zero-shot and Few-shot results on popular LM benchmarks. Normalized accuracy is reported for HellaSwag and exact match (EM) is reported for NaturalQuestions and TriviaQA. All other tasks report unnormalized accuracy. MMLU and TriviaQA are evaluated 5-shot, while all other tasks are zero-shot. We observe **on average**, as expected, that 10B parameter models outperform their 405M counterparts and that the English-only 405M models outperform their German-trained counterparts.

Model Size	Training Tokens	HellaSwag	ARC-c	ARC-e	BoolQ	MathQA	MMLU	OBQA	PIQA	WG	TfQA1	TfQA2	NQ	TrQA	AVG
10B	300B Pile	68.46	34.81	69.49	68.20	27.34	27.28	27.20	76.82	62.51	20.44	33.68	6.65	41.92	43.45
	300B SP	70.38	36.77	71.93	68.04	24.76	27.42	28.20	76.99	65.04	22.40	33.99	11.25	52.63	45.37
	300B Pile → 300B SP	73.66	37.37	73.02	73.18	26.43	29.94	30.20	78.51	66.30	23.26	35.04	12.99	57.94	47.53
	300B Pile → 300B SP (5% Replay)	73.24	39.42	74.24	70.80	26.83	28.79	30.60	78.02	68.67	23.01	35.02	13.32	57.86	47.68
	600B Pile ∪ SP	73.39	39.25	73.57	72.05	26.83	37.78	27.80	77.58	67.32	23.13	36.16	12.41	56.73	48.00
405M	300B Pile	40.95	22.01	51.77	59.24	24.12	26.18	19.80	66.59	53.83	24.85	42.11	0.91	8.97	33.95
	300B SP	44.22	21.76	54.08	59.63	22.71	26.18	19.60	68.23	49.80	22.64	38.63	1.69	14.18	34.11
	300B Pile → 300B SP	46.22	22.70	54.04	57.43	24.22	25.28	21.20	69.26	54.46	23.13	38.91	2.02	15.23	34.93
	300B Pile → 300B SP (5% Replay)	46.55	23.55	55.01	57.92	24.22	25.94	20.60	69.37	54.22	23.38	38.35	1.99	15.70	35.14
	600B Pile ∪ SP	45.06	23.55	52.99	55.57	23.12	26.65	18.20	69.37	52.72	23.50	38.81	1.72	14.63	34.30

TfQA: Truthful QA, WG: WinoGrande, NQ: Natural Questions, OBQA: OpenBook QA, TrQA:TriviaQA

Table. 10 reports English-language LM evaluation results for our english-only continually pre-trained LLMs. Normalized accuracy is reported for HellaSwag and exact match (EM) is reported for NaturalQuestions and TriviaQA. All other tasks report unnormalized accuracy. As expected, we observe that the larger (10B) models achieve stronger performance than their smaller counterparts and that models trained on more tokens always achieve better performance than models trained on fewer tokens. For both model scales, we observe that the models pre-trained continually using a combination of learning rate re-warming and 5% replay approach (10B) or surpass (405M) the performance of the models trained on the union of both datasets in terms of average accuracy. When comparing union-trained models to continually pre-trained models for different tasks, we

observe for the 10B parameter models that the 5% replay model and union-trained model exchange best performance on different tasks with notable differences being OpenBookQA in favor of the replay model and MMLU in favor of the union model. While this degradation in MMLU performance between both models could be cause for concern, we suspect it is due to the limited amount of training data used in our study. Following the initial release of this work, Glorioso et al. (2024) successfully applied our techniques without MMLU performance degradation; in fact, their performance on MMLU is improved during continual pre-training. For the 405M parameter models, the 5% replay model and union-trained model exchange best performance on different tasks with no notable differences. At both model scales, the replay model improves over the model only using re-warming though differences are small and may be attributable to noise.

In summary, we find that models continually pre-trained with a combination of LR re-warming, LR re-decaying, and replay exceed the average performance (e.g., w.r.t. final validation loss and evaluation accuracy) of baselines trained from random initialization on individual datasets and achieve comparable evaluation performance on average to the expensive re-training baseline (trained on the union of both datasets). These results show that the benefits of continual pre-training hold at the $10B$ parameter scale, suggesting that this may also be the case for models with an order of magnitude more parameters (e.g. for $100B+$ parameters).

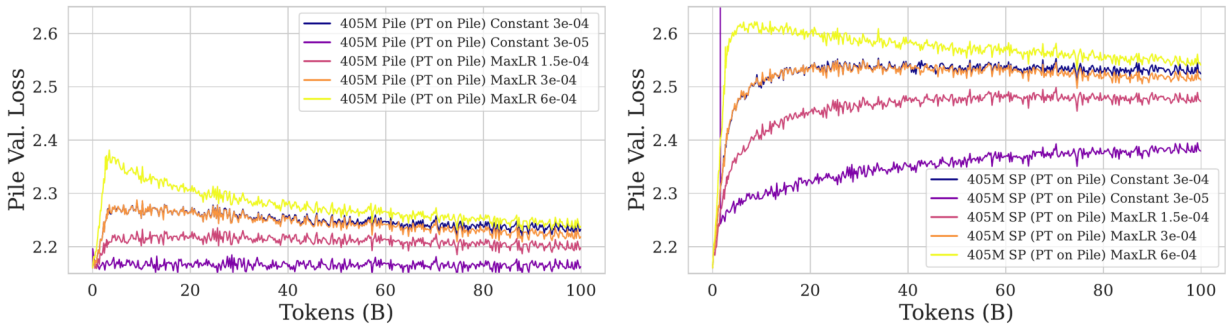
8.7. Understanding and Circumventing the Pathologies of Re-warming

In this section, find that LR re-warming causes unwanted forgetting, introduce infinite learning rate schedules as a promising way to circumvent it, and compare these schedules to baselines from the literature.

8.7.1. Re-warming on the Same Data

In section 8.6.1, we have seen that continuing to pre-train on new data initially leads to a quick increase of the loss on past data, which motivated the use of replay. The increase of the loss was, in particular, more pronounced for greater η_{max} values. One hypothesis for the increase in loss is that it is mostly due to a distribution shift between the pre-training datasets and associated negative transfer. To assess this hypothesis, we re-warm and re-decay over 300B tokens in a setting with no distribution shift. That is, we follow a similar methodology as in our experiments from Fig. 13 but continue to pre-train on Pile as $\mathcal{D}_1$.

As seen in Fig. 17, independently of the distribution shift, rewarming the learning rate appears to be a significant cause of the increase in loss seen previously in Fig. 13 when starting to continue to pre-train, as evidenced by the increase in perplexity when re-warming the learning rate while training on the same distribution. For example, the re-warming leads to a peak increase of the Pile validation loss of 0.1 relative to its initial value with a $\eta_{max} = 3 \cdot 10^{-4}$ as we continue pre-training on



(a) Re-warming (300B Pile→300B Pile) **(b)** Re-warming (300B Pile→300B SlimPajama)

Figure 17. Pile validation loss when continuing to pre-train on Pile (a) and SlimPajama (b). Each curve starts from the same checkpoint pre-trained on 300B tokens of Pile but is trained with a different maximum learning rate. As we focus on the effect of re-warming the learning rate, we only show curves for the first 100B tokens. We observe that every model that re-increases its learning rate from the minimum learning rate of the initial pre-training (e.g., all models except constant) sees an increase in loss.

Pile, which might be contrasted with the Pile validation loss increase of 0.35 with the same learning rate schedule when continuing to pre-train on SlimPajama as in Fig. 13. It is noteworthy that the higher the re-warming, the more pronounced this effect is, as seen with the $\eta_{max} = 6 \cdot 10^{-4}$ curve when continuing to pre-train on Pile (with a peak loss increase of 0.2) vs continuing to pre-train on SlimPajama (peak loss increase of 0.45).

In particular, after re-warming, models fail to recover quickly from the performance hit due to rewarming the learning rate even when training on the same dataset. This motivates finding alternatives to learning rate schedules requiring re-warming in order to improve the efficiency of continual pre-training.

8.7.2. Infinite Learning Rate Schedules

In this subsection, we investigate the use of learning rate schedules that intrinsically may not require re-warming. The motivations are twofold. On the one hand, a cosine decay schedule requires us to know the total number of tokens we want to pre-train on in advance. This limits the ability to continue to pre-train a converged checkpoint. On the other hand, we saw in the previous section that when continuing to pre-train a model that was initially pre-trained with a cosine decay schedule ending with a small learning rate, re-warming the learning rate from its minimum value is needed to best adapt to the new dataset. However, as seen in the previous subsection, we observe that re-warming the learning rate can exacerbate forgetting.

Thus, we explore "Infinite Learning rate schedules" (Zhai et al., 2022) which keep the learning rate at a constant value across all new tasks. This can help prevent forgetting by avoiding re-warming the learning on new tasks. Additionally, this schedule is independent of the total number of tokens making it more suitable for continual learning setups compared to repeating the cosine decay schedule cyclically for each new dataset. As we saw, since a high constant learning rate is also

suboptimal, we opt to perform a fast annealing of the learning rate at the end of pre-training, over a limited amount of tokens. We hope that this will recover the performance advantage of re-decaying the learning rate, while allowing the use of a pre-annealing checkpoint when continuing to pre-train. The infinite learning rate schedules considered have 4 phases:

(1) **Linear warm-up phase –** As before, the learning rate is initially increased to some maximum value η_{max} over T_{warmup} timesteps, or equivalently until timestep $t_{cd} = T_{warmup}$. The learning rate undergoes a warm-up only once (during the first task) and does not require re-warming for future tasks.

(2) **Cooldown phase –** During this stage the learning rate undergoes a cooldown phase where the learning rate is gradually decayed to constant value η_{const} according to some decay function f_{cd} over T_{cd} timesteps from timestep t_{cd} to $t_{const} = t_{cd} + T_{cd}$. This stage also occurs only once during the first task.

(3) **Constant phase –** The learning rate then remains constant for all future tasks over T_{const} timesteps from timestep t_{const} to $t_{ann} = t_{const} + T_{const}$. The checkpoint obtained at the end of this phase is the one one should resume from when continuing to pretrain on a new dataset.

(4) **Annealing phase –** The learning rate is annealed to a small value η_{min} over T_{ann} timesteps from timestep t_{ann} to $t_{end} = t_{ann} + T_{ann}$, helping train the model to convergence before being deployed.

Thus, the infinite learning rate schedules considered here can be written as:

$$\eta_t = \begin{cases} \eta_{max} \cdot \dfrac{t}{T_{warmup}} & t \in [0, t_{cd}] \quad (\textit{warm-up}) \\ f_{cd}(t) & t \in (t_{cd}, t_{const}] \quad (\textit{cooldown}) \\ \eta_{const} & t \in (t_{const}, t_{ann}] \quad (\textit{constant}) \\ \eta_{const} \cdot \left(\dfrac{\eta_{min}}{\eta_{const}}\right)^{\frac{t - t_{ann}}{t_{end} - t_{ann}}} & t \in (t_{ann}, t_{end}] \quad (\textit{annealing}) \end{cases}$$

In this work, we consider the two following functions for the cooldown phase's decay f_{cd}:

(1) Cosine decay

$$f_{cd}(t) = \eta_{const} + \frac{\eta_{max} - \eta_{const}}{2} \cdot \left(1 + \cos\left(\pi \left(\frac{t - t_{cd}}{t_{const} - t_{cd}}\right)\right)\right) \tag{60}$$

(2) Inverse Square Root decay

$$f_{cd}(t) = \eta_{max} + \frac{\eta_{const} - \eta_{max}}{h(1)} \cdot h\left(\frac{t - t_{cd}}{t_{const} - t_{cd}}\right) \tag{61}$$

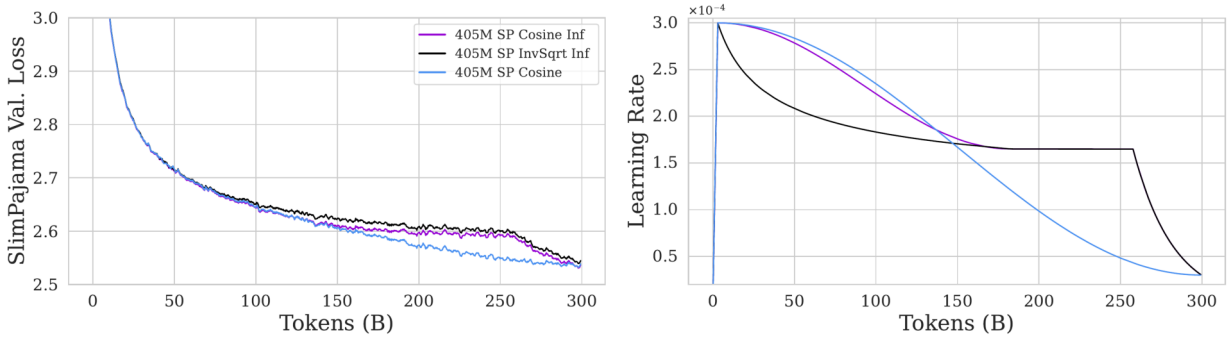
where

$$h(x) = \frac{1}{\sqrt{1 + \alpha x}} - 1$$

with α controlling the steepness of the inverse square root decay. We shift and stretch the Inverse Square root decay to adapt to the interval $(t_{cd}, t_{const}]$.
The three different schedules are seen in Fig. 18 (b).
We now compare infinite learning rate schedules to a cosine decay schedule. We first explore a simple single-dataset pre-training setup to evaluate the feasibility of the schedule for LLM pre-training. Subsequently, we explore its benefits in our *three datasets, no shift* setting.

8.7.3. Comparing Cosine Decay to Variants of our Infinite Schedules

Here we compare a cosine decay schedule with infinite learning rate schedules in the common single-dataset pre-training setting. The aim of these experiments is to test if the infinite learning rate schedules can result in models that perform as well as models trained with a conventional cosine decay schedule.
The models are pre-trained on 300B tokens of SlimPajama from random initialization. Figure 18 shows the training curves of 3 405M parameter models trained on SlimPajama with different learning rate schedules. We observe that all methods reach similar final validation loss showing that infinite learning rate schedules can be used for the common case of pre-training as well. These schedules additionally have the advantage that one can start annealing at any time in the constant phase to efficiently improves the loss when deciding to finalize pre-training, and a pre-annealing checkpoint can be loaded to continue pre-training.



(a) SlimPajama Validation Loss

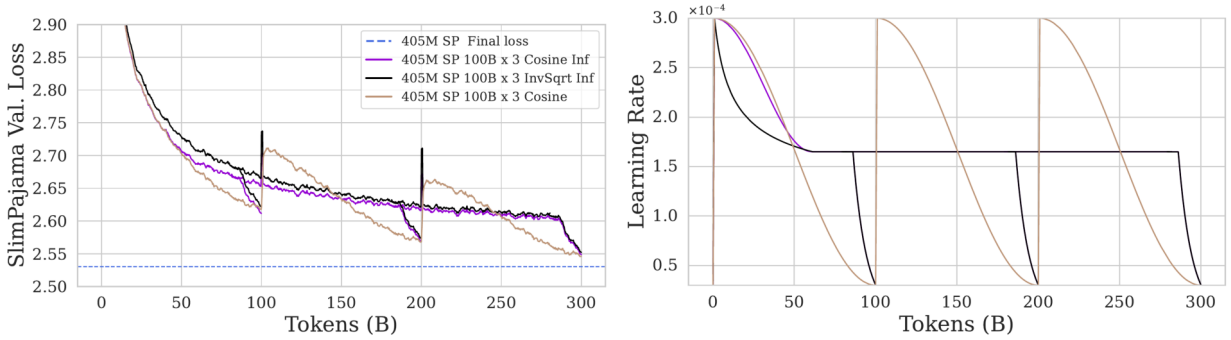
(b) Learning Rate Schedule

Figure 18. Infinite learning rate schedules v.s. Cosine decay. We train a 405M parameter model on 300B tokens of SlimPajama from random initialization with two new schedules, *Cosine Inf* and *InvSqrt Inf*, and compare them to the cosine decay baseline. *Cosine Inf* and *InvSqrt Inf* first decay to a fixed constant LR value and stay constant thereafter until an abrupt final decay. These schedules, therefore, have the advantage that they can smoothly transition between one pre-training phase and the next without re-warming. We find that all methods reach similar final validation loss showing that Cosine decay is not a prerequisite for strong performance.

8.7.4. Infinite Learning Rate Schedules: Scaling to Infinite Future Updates

We now explore the role of the infinite learning rate schedules when multiple new datasets are seen in a continual learning setup. The models are trained from random initialization with different

learning rate schedules on 3 IID 100B subsets of SlimPajama (e.g., our *three datasets no shift* setting; see Sec 8.5.2). We focus on the no shift setting in these preliminary experiments and leave the weak and strong shift cases to future work. This task simulates a setting where large amounts of data from the same distribution are received at time increments and we wish to continue pre-training our models on them (e.g., continuing to pre-train the model on the latest web-scrape). To make our results applicable to situations where previous optimizer states are not available, we do not keep optimizer states across dataset boundaries. Fig. 19 reports training curves for 405M parameter models.



(a) SlimPajama Validation Loss

(b) Learning Rate Schedule

Figure 19. Infinite learning rate schedules evaluated on 3 IID 100B token subsets of SP. The experiment simulates a setting where new data from the same distribution arrives over time and the practitioner wishes to update their model on the new data. The models are trained from random initialization on the first dataset. For each dataset, we train two checkpoints: a checkpoint that continues the constant phase for all data in this dataset and a decayed checkpoint (e.g., phase 4). When transitioning to the new datasets, we select the former. We note that, in figure (b), the black and violet schedules overlap after $\sim$ 80B tokens.

We observe that all schedules perform relatively similarly, however, the two infinite schedules have the advantage that we can start annealing at any time during the constant learning rate phase on each split, while the repeated cosine decays require knowing the number of tokens in advance. Additionally, we see negligible forgetting across dataset boundaries for the infinite LR schedules. While the losses initially increase sharply due to re-initializing the optimizer states, the infinite schedules models immediately recover from this.

In future works, it would be interesting to study the impact of infinite learning rate schedules in continual learning setups with distribution shifts, and investigate the stability of training over large amounts of tokens with a long constant phase of the learning rate.

In summary, we saw that re-warming can hurt performance even when training on the same distribution, but that alternatives to cosine decay schedules might circumvent these issues. Furthermore, these infinite learning rate schedules provide a simple way to end or resume pre-training without being constrained to a particular token budget. That being said, settings with distribution shifts should also be explored to validate these schedules.

8.8. Limitations

While we have conducted a thorough empirical evaluation of continual pre-training for LLMs, there are some limitations to our work. In no particular order: 1) we only studied two model sizes (405M and 10B); 2) we did not run deduplication between the German training and validation datasets created from the German Common Crawl scrape (Laippala et al., 2022); 3) we primarily study the transition between two subsequent tasks; 4) we did not run our experiments over multiple seeds; and 5) our experiments on infinite learning rate schedules are limited to 405M scale with no distribution shift. More explicitly, the first limitation is the number of model scales we consider. While we do consider a 405M and a 10B parameter model (much larger than most works), we could not extend the study to another order of magnitude due to computational limitations (e.g., 100B parameter scale). The second limitation of our work is that the German validation set was not deduplicated from the German training data. While we were careful to take distinct shards for training and validation, there may be some contamination between the two. Given that all baselines have access to the same dataset, however, we believe our results are still valid. The third limitation is that we did not run experiments updating models on more than two subsequent tasks. While we believe that studying this is important, our goal was to focus our compute on different distribution shifts and studying transitions between large datasets, rather than using a large number of datasets. The fourth limitation is that we did not run experiments over multiple seeds due to high computational cost, meaning that there is likely a stochastic element to some results. That being said, our LLMs are trained with a large batch size (2M+ tokens) and, thus, there is little variance in the gradient estimates. Coupled with the fact that the samples from each dataset are processed in the same order in all cases, we believe that our results should be relatively stable to changes in random initialization dictated by the seed. The fifth limitation is that it is very possible that over enough tokens, the infinite schedules may end up being suboptimal due to only having a single phase of warmup and cooldown, as the learning on all subsequent datasets may just be equivalent to using a constant learning rate, which proved to be suboptimal (see Fig. 13). While Fig. 19 showed that the annealing phase helps recover from this suboptimality in the case of IID splits of the same dataset, it is unclear if this would hold over more tokens, or in the case where the different datasets have distribution shifts. Hence, experiments involving distribution shifts, and a larger scale of models and datasets would be important to further test these infinite schedules. Finally, another important consideration to explore at a larger scale is the stability of pre-training with such schedules (in particular, during the constant learning rate phase without μP (Yang et al., 2022)).

8.9. Conclusion

In the context of continual pre-training of autoregressive transformer-based LLMs, we have seen that learning rate re-warming and re-decaying is important for adaptation and found that forgetting is easily mitigated with replay in this setting—at seemingly little cost to adaptation. Given their

powerful ability to enhance adaptation and mitigate forgetting simultaneously, we proposed the simple and scalable combination of LR re-warming, LR re-decaying, and replay for continually pre-training LLMs at scale. We showed that these strategies enable continual pre-training to achieve average performance on par with expensively re-training from scratch on all data, across two distribution shifts (weak & strong) and two decoder-only transformer LLM scales (405M & 10B). Upon further analysis, we identified a pathology of LR re-warming and, inspired by previous work, proposed infinite learning rate schedules for continually pre-training LLMs. In initial experiments, our schedules achieve performance on par with cosine decay while circumventing the need for LR re-warming.

Our findings show that continual pre-training is an efficient and promising alternative to re-training when updating decoder-only transformer LLMs on new data. Equipped with our strategies, practitioners can efficiently update their existing models (Rae et al., 2021; Hoffmann et al., 2022; Touvron et al., 2023b; Jiang et al., 2023; Gemma Team et al., 2024) on newly created higher-quality datasets. These strategies might also be relevant for pre-training curricula such as the ones used by Gemma Team et al. (2024). With the strong incentive for our community to continue creating datasets of increasing quality, we only expect the need for continual pre-training to increase.

In follow-up work, it will be important to further investigate infinite learning rate schedules, growing models during continual pre-training (e.g., mixture-of-experts or block expansion), and adapting the tokenizer to handle drastic changes to the data distribution. Moreover, we would like to explore continual pre-training in the context of multimodal or vision language models and other text-based generative models—we note that recently, Garg et al. (2023) concurrently replicated the success of the techniques discussed in this work in the context of CLIP models instead of LLMs. We also would like to explore replay buffer creating in the continual pre-training setting where an open-weight model does not disclose its dataset; we suspect using the available model for synthetic data or distillation may be a promising direction to build the replay buffer.

Broader Impact Statement. Large language models have seen widespread adoption across a wide range of industry sectors due to their ability to perform very well after being trained on relevant datasets. Moreover, improvements in datasets (better filtering, updating knowledge, etc.) have been crucial to increasing the quality of the output of LLMs. As such, it is reasonable to expect that organizations will spend a significant amount of computing power and, thus, energy to create more powerful models. It is likely that some of this energy will come from non-renewable sources. While the experiments presented in our paper are environmentally costly, as argued in the paper, continuing to pre-train is a promising method to significantly reduce the compute associated with updating a model and, hence, the energy required to maintain foundation models.

Acknowledgements

We acknowledge support from NSERC Discovery Grant RGPIN- 2021-04104 [E.B.], the Canada CIFAR AI Chair Program [I.R.], and the Canada Excellence Research Chairs Program [I.R.]. We would also like to acknowledge funding from the FRQNT Doctoral (B2X) scholarship [B.T.], the scholarship for Artificial Intelligence of Université de Montréal's Études Supérieures et Postdoctorales [A.I.], and a fellowship of the IFI program of the German Academic Exchange Service (DAAD)[M.R.]. This research was made possible thanks to the computing resources on the Summit supercomputer, provided as a part of the INCITE 2023 program award "Scalable Foundation Models for Transferable Generalist AI". These resources were provided by the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. In particular, we thank Jens Glaser for his help with the Summit supercomputer.

Chapter 9

General conclusion

In this thesis, we have striven to propose a more unified view of distributional shifts, and try to leverage those connections to address one of the major flaws of machine learning algorithms: robustness to those shifts.

By contributing to the area of multi-objective optimisation in the publication presented in Chapter 4, we help improve existing and future methods addressing distributional shifts that are based on min-max and other game-theoretical formulations. Since its release, (Ibrahim et al., 2019) has served as a basis to design and establish the optimality of multi-objective optimisers (Fallah et al., 2020; Azizian et al., 2020; Lin et al., 2020; Golowich et al., 2020), with the proof techniques being reused to extend the lower bounds to proximal methods, for example (Zhang et al., 2022).

By showing the links between adversarial robustness and OoD generalisation in the paper presented in Chapter 6, we hope to open a door for new methodology to build safer models. Our experiments showed that leveraging this connection can significantly outperform state-of-the-art defenses. Our work shows, however, that getting truly robust models – i.e., that are robust against multiple unforeseen attacks – is still an open problem, even on simple datasets.

By treating continual pretraining of LLMs as a typical continual learning problem, we show in the paper presented in Chapter 8 how simple strategies with the learning rate and replay of past data mitigate challenges in adaptation and forgetting. In turn, these strategies may help save considerable environmental, energetic, financial and human resources in production. As the bitter lesson suggests, scaling promises to complement well approaches relying on training on a large mixture of data distributions, as evidenced by the success of foundation models as generalists (Achiam et al., 2023). We hope that our contributions will help make the lesson a bit more bitter*sweet*.

Interestingly, my interest in distributional shift started during my initial PhD at UCSB, when my line of work on Human-Computer Interaction ran into them. The project aimed to enable the use of ML algorithms in virtual reality as an educational tool. While preliminary work without ML proved to be very promising (Ibrahim et al., 2018), the “mere” task of object recognition – let alone

multimodality and user personalisation – proved to be too challenging due to obfuscation, lighting conditions, viewing angle, etc. This motivated me to start a PhD in Machine Learning.

The aim of covering several aspects of distributional shifts during my thesis was also to develop a diverse set of skills. My work in optimisation has allowed me to understand better the dynamics of learning in ML. My work on adversarial ML has helped me develop and test a more unified view of distributional shifts and training on non-stationary distributions. Finally, my work on LLMs has allowed me to gain invaluable experience with training very large models on thousands on GPUs, which includes understanding distributed training, optimisation dynamics, and the crucial role of data.

Beyond my academic work, during my PhD, I have consulted in the industry on projects in multimodal machine learning, speech recognition, reinforcement learning, image generation, computer vision, large language models and foundation models for scene understanding. In all cases, distributional shifts were the major challenge to the real world deployment of the models involved. A holistic approach to distributional shifts, drawing inspiration and intuition from one area of ML to tackle problems caused by shifts in another area, has proven successful in many of those projects.

While this thesis marks a step in my research career, it does not mark a stop. Beyond an ongoing collaboration within Mila on improving variational autoencoders, another collaboration with researchers from Stanford and Eleuther aims to make the final performance of LLMs more predictable. Additionally, as a lead of the Open-Sci collective, I will be working with researchers from several institutions and companies (LAION, HuggingFace, Anthropic, Stanford, etc.) in order to research and publish on improving the architecture, data mixture, and training of very large language models, thanks to compute grants on the Jülich and LUMI supercomputers. Last but not least, methodology and intuition developed from working on (Ibrahim et al., 2024) helped us tame and turn distributional shifts into a feature while training Zyphra's Zamba-7B model (Glorioso et al., 2024). Indeed, training only on 1T tokens in a small team of 7 persons, we outperform on standard benchmarks Llama 2 (2T tokens) (Touvron et al., 2023b) by a wide margin, while getting close to the performance of Mistral 7B (several trillions of tokens) (Jiang et al., 2023) and the Gemma Team et al. (2024)'s 8.5B parameter model (6T tokens); a first for open-source models.

Bibliography

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ahuja, K., Shanmugam, K., Varshney, K., and Dhurandhar, A. (2020). Invariant risk minimization games. In *International Conference on Machine Learning*, pages 145–155. PMLR.

Alayrac, J., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J. L., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K. (2022). Flamingo: a visual language model for few-shot learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Albuquerque, I., Monteiro, J., Falk, T. H., and Mitliagkas, I. (2019). Adversarial target-invariant representation learning for domain generalization. *arXiv:1911.00804*.

Aljundi, R., Caccia, L., Belilovsky, E., Caccia, M., Lin, M., Charlin, L., and Tuytelaars, T. (2019). Online continual learning with maximal interfered retrieval. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 11849–11860. Curran Associates, Inc.

Almazrouei, E., Alobeidli, H., Alshamsi, A., Cappelli, A., Cojocaru, R., Debbah, M., Goffinet, É., Hesslow, D., Launay, J., Malartic, Q., Mazzotta, D., Noune, B., Pannier, B., and Penedo, G. (2023). The falcon series of open language models. *CoRR*, abs/2311.16867.

Amini, A., Gabriel, S., Lin, P., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. (2019). Mathqa: Towards interpretable math word problem solving with operation-based formalisms.

Andonian, A., Anthony, Q., Biderman, S., Black, S., Gali, P., Gao, L., Hallahan, E., Levy-Kramer, J., Leahy, C., Nestler, L., Parker, K., Pieler, M., Purohit, S., Songz, T., Phil, W., and Weinbach, S. (2021). GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. (2020). Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer.

Arjevani, Y., Shalev-Shwartz, S., and Shamir, O. (2016). On lower and upper bounds in smooth and strongly convex optimization. *The Journal of Machine Learning Research*, 17(1):4303–4353.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2019). Invariant risk minimization. *arXiv:1907.02893*.

Athalye, A., Carlini, N., and Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR.

Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M. D., McAleer, S., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. (2023). Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Azizian, W., Scieur, D., Mitliagkas, I., Lacoste-Julien, S., and Gidel, G. (2020). Accelerating smooth games by manipulating spectral shapes. *arXiv preprint arXiv:2001.00602*.

Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., and Graepel, T. (2018). The mechanics of n-player differentiable games. In *International Conference on Machine Learning*, pages 363–372.

Bashivan, P., Bayat, R., Ibrahim, A., Ahuja, K., Faramarzi, M., Laleh, T., Richards, B., and Rish, I. (2021). Adversarial feature desensitization. *Advances in Neural Information Processing Systems*, 34.

Bashivan, P., Richards, B., and Rish, I. (2020). Adversarial feature desensitization. *arXiv preprint arXiv:2006.04621*.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2007). Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Benzi, M. and Golub, G. H. (2004). A preconditioner for generalized saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 26(1):20–41.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. (2019). Piqa: Reasoning about physical commonsense in natural language.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B., and Weinbach, S. (2022). Gpt-neox-20b: An open-source autoregressive language model.

Brassard, G. and Bratley, P. (1996). Fundamentals of algorithmics.

Brendel, W., Rauber, J., Kümmerer, M., Ustyuzhaninov, I., and Bethge, M. (2019). Accurate, reliable and fast robustness evaluation. *Advances in neural information processing systems*, 32.

Brooks, T., Peebles, B., Homes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C. W. Y., Wang, R., and Ramesh, A. (2024). Video generation models as world simulators.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 1877–1901.

Bubeck, S. et al. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Caccia, M., Rodriguez, P., Ostapenko, O., Normandin, F., Lin, M., Caccia, L., Laradji, I., Rish, I., Lacoste, A., Vazquez, D., and Charlin, L. (2020). Online fast adaptation and knowledge accumulation: a new approach to continual learning. *NeurIPS*.

Cai, Q.-Z., Liu, C., and Song, D. (2018). Curriculum adversarial training. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3740–3747.

Carlini, N. and Wagner, D. (2017a). Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14.

Carlini, N. and Wagner, D. (2017b). Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE.

Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145.

Chen, C., Xie, W., Huang, W., Rong, Y., Ding, X., Huang, Y., Xu, T., and Huang, J. (2019). Progressive feature alignment for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 627–636.

Chen, G. H. and Rockafellar, R. T. (1997). Convergence rates in forward–backward splitting. *SIAM Journal on Optimization*, 7(2):421–444.

Chen, L., Zaharia, M., and Zou, J. (2023). How is chatgpt's behavior changing over time? *arXiv preprint arXiv:2307.09009*.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. (2016). Training deep nets with sublinear memory cost. *CoRR*, abs/1604.06174.

Chen, Y., Lan, G., and Ouyang, Y. (2017). Accelerated schemes for a class of variational inequalities. *Mathematical Programming*, 165(1):113–149.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. (2019). Boolq: Exploring the surprising difficulty of natural yes/no questions.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge.

Computer, T. (2023). Redpajama: an open dataset for training large language models.

Cossu, A., Tuytelaars, T., Carta, A., Passaro, L., Lomonaco, V., and Bacciu, D. (2022). Continual pre-training mitigates forgetting in language and vision.

Croce, F. and Hein, M. (2020). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR.

Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. (2018). Training GANs with optimism. In *International Conference on Learning Representations*.

De Lange, M., van de Ven, G., and Tuytelaars, T. (2022). Continual evaluation for lifelong learning: Identifying the stability gap. *arXiv preprint arXiv:2205.13452*.

DeepSeek-AI (2024). Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model.

DeepSeek-AI, Zhu, Q., Guo, D., Shao, Z., Yang, D., Wang, P., Xu, R., Wu, Y., Li, Y., Gao, H., Ma, S., Zeng, W., Bi, X., Gu, Z., Xu, H., Dai, D., Dong, K., Zhang, L., Piao, Y., Gou, Z., Xie, Z., Hao, Z., Wang, B., Song, J., Chen, D., Xie, X., Guan, K., You, Y., Liu, A., Du, Q., Gao, W., Lu, X., Chen, Q., Wang, Y., Deng, C., Li, J., Zhao, C., Ruan, C., Luo, F., and Liang, W. (2024). Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *CoRR*, abs/2406.11931.

Deng, Y., Zheng, X., Zhang, T., Chen, C., Lou, G., and Kim, M. (2020). An analysis of adversarial attacks and defenses on autonomous driving models. In *2020 IEEE international conference on pervasive computing and communications (PerCom)*, pages 1–10. IEEE.

Ding, G. W., Wang, L., and Jin, X. (2019). Advertorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*.

Dong, Y., Fu, Q.-A., Yang, X., Pang, T., Su, H., Xiao, Z., and Zhu, J. (2020). Benchmarking adversarial robustness on image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 321–331.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Douglas, R. G. (2012). *Banach algebra techniques in operator theory*, volume 179. Springer Science & Business Media.

Douillard, A., Feng, Q., Rusu, A. A., Chhaparia, R., Donchev, Y., Kuncoro, A., Ranzato, M., Szlam, A., and Shen, J. (2023). Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*.

Du, S. S., Chen, J., Li, L., Xiao, L., and Zhou, D. (2017). Stochastic variance reduction methods for policy evaluation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1049–1058. JMLR. org.

Dziugaite, G. K., Ghahramani, Z., and Roy, D. M. (2016). A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*.

Ebrahimi, S., Meier, F., Calandra, R., Darrell, T., and Rohrbach, M. (2020). Adversarial continual learning.

Ellis, A. W. and Lambon Ralph, M. A. (2000). Age of acquisition effects in adult lexical processing reflect loss of plasticity in maturing systems: insights from connectionist networks. *Journal of Experimental Psychology: Learning, memory, and cognition*, 26(5):1103.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634.

Fallah, A., Ozdaglar, A., and Pattathil, S. (2020). An optimal multistage stochastic gradient method for minimax problems. *arXiv preprint arXiv:2002.05683*.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. (2020). The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Garg, S., Farajtabar, M., Pouransari, H., Vemulapalli, R., Mehta, S., Tuzel, O., Shankar, V., and Faghri, F. (2023). Tic-clip: Continual training of clip models. *arXiv preprint arXiv:2310.16226*.

Gemma Team, T. M., Hardin, C., Dadashi, R., Bhupatiraju, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., Tafti, P., Hussenot, L., and et al. (2024). Gemma: Open models based on gemini research and technology.

Gidel, G., Berard, H., Vignoud, G., Vincent, P., and Lacoste-Julien, S. (2019a). A variational inequality perspective on generative adversarial networks. In *International Conference on Learning Representations*.

Gidel, G., Hemmat, R. A., Pezeshki, M., Priol, R. L., Huang, G., Lacoste-Julien, S., and Mitliagkas, I. (2019b). Negative momentum for improved game dynamics. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019*, pages 1802–1811.

Glorioso, P., Anthony, Q., Tokpanov, Y., Whittington, J., Pilault, J., Ibrahim, A., and Millidge, B. (2024). Zamba: A compact 7b SSM hybrid model. *CoRR*, abs/2405.16712.

Goddard, C., Siriwardhana, S., Ehghaghi, M., Meyers, L., Karpukhin, V., Benedict, B., McQuade, M., and Solawetz, J. (2024). Arcee's mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*.

Gogoulou, E., Lesort, T., Boman, M., and Nivre, J. (2023). A study of continual learning under language shift. *CoRR*, abs/2311.01200.

Golowich, N., Pattathil, S., and Daskalakis, C. (2020). Tight last-iterate convergence rates for no-regret learning in multi-player games. *Advances in neural information processing systems*, 33:20766–20778.

Gong, Z., Zhou, K., Zhao, X., Sha, J., Wang, S., and Wen, J.-R. (2022). Continual pre-training of language models for math problem understanding with syntax-aware memory network.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2018). Accurate, large minibatch sgd: Training imagenet in 1 hour.

Gulrajani, I. and Lopez-Paz, D. (2020). In search of lost domain generalization. In *International Conference on Learning Representations*.

Guo, C., Rana, M., Cisse, M., and van der Maaten, L. (2018). Countering adversarial images using input transformations. In *International Conference on Learning Representations*.

Gupta, K., Thérien, B., Ibrahim, A., Richter, M. L., Anthony, Q., Belilovsky, E., Rish, I., and Lesort, T. (2023a). Continual pre-training of large language models: How to (re) warm your model? *arXiv preprint arXiv:2308.04014*.

Gupta, K., Thérien, B., Ibrahim, A., Richter, M. L., Anthony, Q., Belilovsky, E., Rish, I., and Lesort, T. (2023b). Continual pre-training of large language models: How to (re)warm your model?

Gururangan, S., Marasovic, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. (2020). Don't stop pretraining: Adapt language models to domains and tasks. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8342–8360. Association for Computational Linguistics.

Harker, P. T. and Pang, J.-S. (1990). Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications. *Mathematical programming*, 48(1-3):161–220.

Harun, M. Y., Gallardo, J., Hayes, T. L., and Kanan, C. (2023a). How efficient are today's continual learning algorithms? In *Proceedings of the IEEE/CVF Conference on Computer Vision and*

Pattern Recognition, pages 2430–2435.

Harun, M. Y., Gallardo, J., Hayes, T. L., Kemker, R., and Kanan, C. (2023b). Siesta: Efficient online continual learning with sleep.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). Measuring massive multitask language understanding.

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*.

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. (2021). Scaling laws for transfer. *CoRR*, abs/2102.01293.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, M. X., Chen, D., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., and Chen, Z. (2019). Gpipe: Efficient training of giant neural networks using pipeline parallelism.

Ibrahim, A., Azizian, W., Gidel, G., and Mitliagkas, I. (2019). Linear lower bounds and conditioning of differentiable games. *arXiv preprint arXiv:1906.07300*.

Ibrahim, A., Guille-Escuret, C., Mitliagkas, I., Rish, I., Krueger, D., and Bashivan, P. (2022). Towards out-of-distribution adversarial robustness. *arXiv preprint arXiv:2210.03150*.

Ibrahim, A., Huynh, B., Downey, J., Höllerer, T., Chun, D., and O'donovan, J. (2018). Arbis pictus: A study of vocabulary learning with augmented reality. *IEEE transactions on visualization and computer graphics*, 24(11):2867–2874.

Ibrahim, A., Thérien, B., Gupta, K., Richter, M. L., Anthony, Q., Lesort, T., Belilovsky, E., and Rish, I. (2024). Simple and scalable strategies to continually pre-train large language models. *arXiv preprint arXiv:2403.08763*.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. (2020). The impact of non-stationarity on generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*.

Jang, J., Ye, S., Lee, C., Yang, S., Shin, J., Han, J., Kim, G., and Seo, M. (2022a). Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 6237–6250. Association for Computational Linguistics.

Jang, J., Ye, S., Yang, S., Shin, J., Han, J., Kim, G., Choi, S. J., and Seo, M. (2022b). Towards continual knowledge learning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de Las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. (2023). Mistral 7b. *CoRR*, abs/2310.06825.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Barzilay, R. and Kan, M.-Y., editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Ke, Z., Shao, Y., Lin, H., Konishi, T., Kim, G., and Liu, B. (2022). Continual pre-training of language models.

Khoury, M. and Hadfield-Menell, D. (2018). On the geometry of adversarial examples. *arXiv preprint arXiv:1811.00525*.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada.

Kim, S.-J. and Boyd, S. (2008). A minimax theorem with applications to machine learning, signal processing, and finance. *SIAM Journal on Optimization*, 19(3):1344–1367.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. (2023). Segment anything. *arXiv:2304.02643*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proc. of the national academy of sciences*.

Korpelevich, G. (1976). The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Priol, R. L., and Courville, A. (2020). Out-of-distribution generalization via risk extrapolation (rex). *arXiv preprint arXiv:2003.00688*.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., and Courville, A. (2021). Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR.

Kumar, A., Sattigeri, P., Wadhawan, K., Karlinsky, L., Feris, R., Freeman, B., and Wornell, G. (2018). Co-regularized alignment for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 9345–9356.

Kurakin, A., Goodfellow, I., and Bengio, S. (2017). Adversarial examples in the physical world. *ICLR Workshop*.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Laidlaw, C. and Feizi, S. (2019). Functional adversarial attacks. *Advances in neural information processing systems*, 32.

Laidlaw, C., Singla, S., and Feizi, S. (2021). Perceptual adversarial robustness: Defense against unseen threat models. In *International Conference on Learning Representations*.

Laippala, V., Salmela, A., Rönnqvist, S., Aji, A. F., Chang, L., Dhifallah, A., Goulart, L., Kortelainen, H., Pàmies, M., Dutra, D. P., Skantsi, V., Sutawika, L., and Pyysalo, S. (2022). Towards better structured and less noisy web data: Oscar with register annotations. In *Proceedings of the Eighth Workshop on Noisy User-generated Text, W-NUT@COLING 2022, Gyeongju, Republic of Korea, October 12 - 17, 2022*, pages 215–221. Association for Computational Linguistics.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lesort, T., Caccia, M., and Rish, I. (2021). Understanding continual learning settings with data distribution drift analysis. *arXiv preprint arXiv:2104.01678*.

Lesort, T., Ostapenko, O., Rodríguez, P., Misra, D., Arefin, M. R., Charlin, L., and Rish, I. (2023). Challenging common assumptions about catastrophic forgetting and knowledge accumulation. In Chandar, S., Pascanu, R., Sedghi, H., and Precup, D., editors, *Conference on Lifelong Learning Agents, 22-25 August 2023, McGill University, Montréal, Québec, Canada*, volume 232 of *Proceedings of Machine Learning Research*, pages 43–65. PMLR.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. (2018). Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Li, Z., Shi, C., Xie, Y., Liu, J., Yuan, B., and Chen, Y. (2020). Practical adversarial attacks against speaker recognition systems. In *Proceedings of the 21st international workshop on mobile computing systems and applications*, pages 9–14.

Lin, S., Hilton, J., and Evans, O. (2022). Truthfulqa: Measuring how models mimic human falsehoods.

Lin, T., Jin, C., Jordan, M., et al. (2020). Near-optimal algorithms for minimax optimization. *arXiv preprint arXiv:2002.02417*.

Liu, X., Cheng, M., Zhang, H., and Hsieh, C.-J. (2018). Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages

369–385.

Loizou, N., Berard, H., Jolicoeur-Martineau, A., Vincent, P., Lacoste-Julien, S., and Mitliagkas, I. (2020). Stochastic hamiltonian gradient methods for smooth games. *arXiv preprint arXiv:2007.04202*.

Loizou, N. and Richtárik, P. (2017). Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *arXiv preprint arXiv:1712.09677*.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lyle, C., Zheng, Z., Khetarpal, K., van Hasselt, H., Pascanu, R., Martens, J., and Dabney, W. (2024). Disentangling the causes of plasticity loss in neural networks. *arXiv preprint arXiv:2402.18762*.

Lyle, C., Zheng, Z., Nikishin, E., Avila Pires, B., Pascanu, R., and Dabney, W. (2023). Understanding plasticity in neural networks. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 23190–23211. PMLR.

Ma, S., Huang, S., Huang, S., Wang, X., Li, Y., Zheng, H., Xie, P., Huang, F., and Jiang, Y. (2023). Ecomgpt-ct: Continual pre-training of e-commerce large language models with semi-structured data. *CoRR*, abs/2312.15696.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Maini, P., Wong, E., and Kolter, Z. (2020). Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pages 6640–6650. PMLR.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Mehta, S. V., Patil, D., Chandar, S., and Strubell, E. (2023). An empirical investigation of the role of pre-training in lifelong learning. *J. Mach. Learn. Res.*, 24:214:1–214:50.

Mermillod, M., Bugaiska, A., and Bonin, P. (2013a). The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:54654.

Mermillod, M., Bugaiska, A., and Bonin, P. (2013b). The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4(August):504.

Mescheder, L., Nowozin, S., and Geiger, A. (2017). The numerics of GANs. In *Advances in Neural Information Processing Systems*, pages 1825–1835.

Microsoft (2020). Megatron-DeepSpeed. https://github.com/microsoft/Megatron-DeepSpeed. Accessed: February 28, 2024.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. (2018). Can a suit of armor conduct electricity? a new dataset for open book question answering.

Mirzadeh, S., Chaudhry, A., Yin, D., Hu, H., Pascanu, R., Görür, D., and Farajtabar, M. (2022). Wide neural networks forget less catastrophically. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S., editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 15699–15717. PMLR.

Mirzadeh, S. I., Chaudhry, A., Hu, H., Pascanu, R., Gorur, D., and Farajtabar, M. (2021). Wide neural networks forget less catastrophically. *arXiv preprint arXiv:2110.11526*.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582.

Nemirovsky, A. (1992). Information-based complexity of linear operator equations. *Journal of Complexity*, 8(2):153–175.

Nemirovsky, A. S. and Yudin, D. B. (1983). Problem complexity and method efficiency in optimization.

Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate O (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

Ostapenko, O., Puscas, M., Klein, T., Jahnichen, P., and Nabi, M. (2019). Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11321–11329.

Ouyang, Y. and Xu, Y. (2018). Lower complexity bounds of first-order methods for convex-concave bilinear saddle-point problems. *arXiv preprint arXiv:1808.02901*.

Palaniappan, B. and Bach, F. (2016). Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, pages 1416–1424.

Pang, T., Xu, K., Du, C., Chen, N., and Zhu, J. (2019). Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979.

Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. (2020). Bag of tricks for adversarial training. In *International Conference on Learning Representations*.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE.

Pernias, P., Rampas, D., Richter, M. L., Pal, C., and Aubreville, M. (2024). Würstchen: An efficient architecture for large-scale text-to-image diffusion models. In *The Twelfth International Conference on Learning Representations*.

Plüster, B. (2023). German Benchmark Datasets.

Popel, M. and Bojar, O. (2018). Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Prabhu, A., Cai, Z., Dokania, P., Torr, P., Koltun, V., and Sener, O. (2023). Online continual learning without the storage constraint. *arXiv preprint arXiv:2305.09253*.

Qayyum, A., Qadir, J., Bilal, M., and Al-Fuqaha, A. (2020). Secure and robust machine learning for healthcare: A survey. *IEEE Reviews in Biomedical Engineering*, 14:156–180.

Qin, Y., Qian, C., Han, X., Lin, Y., Wang, H., Xie, R., Liu, Z., Sun, M., and Zhou, J. (2023). Recyclable tuning for continual pre-training.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). Dataset shift in machine learning.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. (2021). Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2023). Exploring the limits of transfer learning with a unified text-to-text transformer.

Raghunathan, A., Xie, S. M., Yang, F., Duchi, J., and Liang, P. (2020). Understanding and mitigating the tradeoff between robustness and accuracy. *arXiv preprint arXiv:2002.10716*.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. (2020). Zero: memory optimizations toward training trillion parameter models. In Cuicchi, C., Qualters, I., and Kramer, W. T., editors, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM.

Ramasesh, V. V., Lewkowycz, A., and Dyer, E. (2022). Effect of scale on catastrophic forgetting in neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. (2021). Adaptive federated optimization. In *International Conference on Learning Representations*.

Rice, L., Wong, E., and Kolter, Z. (2020). Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR.

Richardson, L. F. (1911). Ix. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210(459-470):307–357.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., , and Tesauro, G. (2019). Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. (2019). Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 348–358.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE.

Ryabinin, M., Gorbunov, E., Plokhotnyuk, V., and Pekhimenko, G. (2021). Moshpit sgd: Communication-efficient decentralized training on heterogeneous unreliable devices. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18195–18211. Curran Associates, Inc.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. (2019). Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization. *arXiv:1911.08731*.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. (2019). Winogrande: An adversarial winograd schema challenge at scale.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. In *Advances in neural information processing systems*, pages 2234–2242.

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. (2020). Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. (2022). Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Scialom, T., Chakrabarty, T., and Muresan, S. (2022). Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6107–6122.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In Erk, K. and Smith, N. A., editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. (2020). Megatron-lm: Training multi-billion parameter language models using model parallelism.

Shu, R., Bui, H. H., Narui, H., and Ermon, S. (2018). A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*.

Singh, M., Gustafson, L., Adcock, A., de Freitas Reis, V., Gedik, B., Kosaraju, R. P., Mahajan, D., Girshick, R., Dollár, P., and Van Der Maaten, L. (2022). Revisiting weakly supervised pre-training of visual perception models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 804–814.

Sitawarin, C., Bhagoji, A. N., Mosenia, A., Chiang, M., and Mittal, P. (2018). Darts: Deceiving autonomous cars with toxic signs. *arXiv preprint arXiv:1802.06430*.

Smith, J., Hsu, Y.-C., Balloch, J., Shen, Y., Jin, H., and Kira, Z. (2021). Always be dreaming: A new approach for data-free class-incremental learning. pages 9374–9384.

Soboleva, D., Al-Khateeb, F., Myers, R., Steeves, J. R., Hestness, J., and Dey, N. (2023). SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama.

Soldaini, L., Kinney, R., Bhagia, A., Schwenk, D., Atkinson, D., Authur, R., Bogin, B., Chandu, K., Dumas, J., Elazar, Y., Hofmann, V., Jha, A. H., Kumar, S., Lucy, L., Lyu, X., Lambert, N., Magnusson, I., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Ravichander, A., Richardson, K., Shen, Z., Strubell, E., Subramani, N., Tafjord, O., Walsh, P., Zettlemoyer, L., Smith, N. A., Hajishirzi, H., Beltagy, I., Groeneveld, D., Dodge, J., and Lo, K. (2024). Dolma: an open corpus of three trillion tokens for language model pretraining research. *CoRR*, abs/2402.00159.

Song, C., He, K., Wang, L., and Hopcroft, J. E. (2018). Improving the generalization of adversarial training with domain adaptation. *arXiv preprint arXiv:1810.00740*.

Su, J.-C., Tsai, Y.-H., Sohn, K., Liu, B., Maji, S., and Chandraker, M. (2020). Active adversarial domain adaptation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 739–748.

Sun, Y., Wang, S., Li, Y., Feng, S., Tian, H., Wu, H., and Wang, H. (2020). ERNIE 2.0: A continual pre-training framework for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8968–8975. AAAI Press.

Sutton, R. (2019). The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023a). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton-Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023b). Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Tramèr, F. and Boneh, D. (2019). Adversarial training and robustness for multiple perturbations. *arXiv preprint arXiv:1904.13000*.

Tseng, P. (1995). On linear convergence of iterative methods for the variational inequality problem. *Journal of Computational and Applied Mathematics*, 60(1-2):237–252.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2018). Robustness may be at odds with accuracy. In *International Conference on Learning Representations*.

Utrera, F., Kravitz, E., Erichson, N. B., Khanna, R., and Mahoney, M. W. (2021). Adversarially-trained deep nets transfer better: Illustration on image classification. In *International Conference on Learning Representations*.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

Vaswani, S., Bach, F., and Schmidt, M. (2019). Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019*, pages 1195–1204.

Veniat, T., Denoyer, L., and Ranzato, M. (2021). Efficient continual learning with modular networks and task-driven priors. In *International Conference on Learning Representations*.

Volpi, R., Namkoong, H., Sener, O., Duchi, J. C., Murino, V., and Savarese, S. (2018). Generalizing to unseen domains via adversarial data augmentation. In *Advances in neural information processing systems*, pages 5334–5344.

Wang, H., Weng, C., and Yuan, J. (2014). Multi-feature spectral clustering with minimax optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4106–4113.

Wang, J., Lan, C., Liu, C., Ouyang, Y., Zeng, W., and Qin, T. (2021). Generalizing to unseen domains: A survey on domain generalization. *arXiv preprint arXiv:2103.03097*.

Wang, M. and Deng, W. (2018). Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153.

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., and Gu, Q. (2019). On the convergence and robustness of adversarial training. In *International Conference on Machine Learning*, pages 6586–6595.

Wilson, G. and Cook, D. J. (2020). A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. (2018). Scaling provable adversarial defenses. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 8410–8419. Curran Associates, Inc.

Wu, C., Gan, Y., Ge, Y., Lu, Z., Wang, J., Feng, Y., Luo, P., and Shan, Y. (2024). Llama pro: Progressive llama with block expansion. *CoRR*, abs/2401.02415.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. (2018). Spatially transformed adversarial examples. In *International Conference on Learning Representations*.

Xie, Y., Aggarwal, K., and Ahmad, A. (2023). Efficient continual pre-training for building domain specific large language models.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. (2020). On layer normalization in the transformer architecture.

Yadav, P., Sun, Q., Ding, H., Li, X., Zhang, D., Tan, M., Bhatia, P., Ma, X., Nallapati, R., Ramanathan, M. K., Bansal, M., and Xiang, B. (2023a). Exploring continual learning for code generation models. In Rogers, A., Boyd-Graber, J. L., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 782–792. Association for Computational Linguistics.

Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. (2023b). Ties-merging: Resolving interference when merging models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yan, Z., Guo, Y., and Zhang, C. (2018). Deep defense: Training dnns with improved adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 419–428.

Yang, G., Hu, E. J., Babuschkin, I., Sidor, S., Farhi, D., Pachocki, J., Liu, X., Chen, W., and Gao, J. (2022). Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. In *NeurIPS 2021*.

Yang, X., Gao, J., Xue, W., and Alexandersson, E. (2024). Pllama: An open-source large language model for plant science. *CoRR*, abs/2401.01600.

You, K., Long, M., Wang, J., and Jordan, M. I. (2019). How does learning rate decay help modern neural networks?

Yu, C., Wang, J., Chen, Y., and Huang, M. (2019). Transfer learning with dynamic adversarial adaptation network. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 778–786. IEEE.

Zan, D., Chen, B., Yang, D., Lin, Z., Kim, M., Guan, B., Wang, Y., Chen, W., and Lou, J. (2022). CERT: continual pre-training on sketches for library-oriented code generation. In Raedt, L. D., editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2369–2375. ijcai.org.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. (2019). Hellaswag: Can a machine really finish your sentence?

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2022). Scaling vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 1204–1213. IEEE.

Zhang, F., editor (2005). *The Schur Complement and Its Applications*, volume 4 of *Numerical Methods and Algorithms*. Springer-Verlag, New York.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. (2019). Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482.

Zhang, J., Hong, M., and Zhang, S. (2022). On lower iteration complexity bounds for the convex concave saddle point problems. *Mathematical Programming*, 194(1):901–935.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Chapter A

n-player quadratic games

While our work focuses mostly on 2-player games, one of the main results, Prop. **??**, is independent of the number of players, and is proven for general n. Therefore, a short discussion of the form of quadratic n-player games is provided below.
For an n-player game, the general form of a quadratic is given by

$$l_i(\mathbf{w}) = \sum_{j=1}^{n}\sum_{k=1}^{n} \mathbf{w}_j^\top \boldsymbol{M}_{ijk}\mathbf{w}_k + \sum_{j=1}^{n} \mathbf{w}_j^\top \mathbf{b}_{ij} + c_i \,. \tag{62}$$

Because the dynamics depend only on the $\nabla_{\mathbf{w}_i} l_i(\mathbf{w})$, we will get equivalent dynamics by pruning the terms that do not depend on $\mathbf{w}_i$ and working directly with the simpler objectives

$$\begin{aligned} l_i(\mathbf{w}) &= \frac{1}{2}\mathbf{w}_i^\top \boldsymbol{M}_{ii}\mathbf{w}_i + \sum_{j\neq i}^{n} \mathbf{w}_i^\top \boldsymbol{M}_{iij}\mathbf{w}_j + \sum_{j\neq i}^{n} \mathbf{w}_j^\top \boldsymbol{M}_{iji}\mathbf{w}_i + \mathbf{w}_i^\top \mathbf{b}_{ii} \\ &= \frac{1}{2}\mathbf{w}_i^\top \boldsymbol{M}_{ii}\mathbf{w}_i + \sum_{j\neq i}^{n} \mathbf{w}_i^\top \boldsymbol{M}_{ij}\mathbf{w}_j + \mathbf{w}_i^\top \mathbf{b}_i \end{aligned} \tag{63}$$

where we have let $\boldsymbol{M}_{ij} \triangleq \boldsymbol{M}_{iij} + \boldsymbol{M}_{iji}^\top, \mathbf{b}_i \triangleq \mathbf{b}_{ii}\,, 1 \leq i,j \leq n$. Note that we may assume the $\boldsymbol{M}_{ii}$ to be symmetric, since in general $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} = \frac{1}{2}\boldsymbol{x}^\top(\boldsymbol{A} + \boldsymbol{A}^\top)\boldsymbol{x}$. Thus, we can write:

$$\nabla_{\mathbf{w}_i} l_i(\mathbf{w}) = \begin{pmatrix} \boldsymbol{M}_{i1} & \dots & \boldsymbol{M}_{in} \end{pmatrix} \mathbf{w} + \mathbf{b}_i \tag{64}$$

which yields the following equation for the vector field:

$$\boldsymbol{v}(\mathbf{w}) = \boldsymbol{A}\mathbf{w} + \mathbf{b}, \qquad \boldsymbol{A} \triangleq \begin{pmatrix} \boldsymbol{S}_1 & \dots & \boldsymbol{M}_{1n} \\ \vdots & & \vdots \\ \boldsymbol{M}_{n1} & \dots & \boldsymbol{S}_n \end{pmatrix}, \quad \mathbf{b} \triangleq \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{pmatrix} \tag{65}$$

where $\boldsymbol{A}$ is the Jacobian of $\boldsymbol{v}$ and where we let $\boldsymbol{S}_i \triangleq \boldsymbol{M}_{ii}$.

Chapter B

Supplementary Material For the First Article

B.1. n-player quadratic games

While our work focuses mostly on 2-player games, one of the main results, Prop. 2, is independent of the number of players, and is proven for general n. Therefore, a short discussion of the form of quadratic n-player games is provided below.

For an n-player game, the general form of a quadratic is given by

$$l_i(\mathbf{w}) = \sum_{j=1}^{n}\sum_{k=1}^{n} \mathbf{w}_j^\top \boldsymbol{M}_{ijk}\mathbf{w}_k + \sum_{j=1}^{n} \mathbf{w}_j^\top \mathbf{b}_{ij} + c_i \,. \tag{66}$$

Because the dynamics depend only on the $\nabla_{\mathbf{w}_i} l_i(\mathbf{w})$, we will get equivalent dynamics by pruning the terms that do not depend on $\mathbf{w}_i$ and working directly with the simpler objectives

$$\begin{aligned} l_i(\mathbf{w}) &= \frac{1}{2}\mathbf{w}_i^\top \boldsymbol{M}_{ii}\mathbf{w}_i + \sum_{j\neq i}^{n} \mathbf{w}_i^\top \boldsymbol{M}_{iij}\mathbf{w}_j + \sum_{j\neq i}^{n} \mathbf{w}_j^\top \boldsymbol{M}_{iji}\mathbf{w}_i + \mathbf{w}_i^\top \mathbf{b}_{ii} \\ &= \frac{1}{2}\mathbf{w}_i^\top \boldsymbol{M}_{ii}\mathbf{w}_i + \sum_{j\neq i}^{n} \mathbf{w}_i^\top \boldsymbol{M}_{ij}\mathbf{w}_j + \mathbf{w}_i^\top \mathbf{b}_i \end{aligned} \tag{67}$$

where we have let $\boldsymbol{M}_{ij} \triangleq \boldsymbol{M}_{iij} + \boldsymbol{M}_{iji}^\top$, $\mathbf{b}_i \triangleq \mathbf{b}_{ii}$, $1 \leq i,j \leq n$. Note that we may assume the $\boldsymbol{M}_{ii}$ to be symmetric, since in general $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} = \frac{1}{2}\boldsymbol{x}^\top(\boldsymbol{A} + \boldsymbol{A}^\top)\boldsymbol{x}$. Thus, we can write:

$$\nabla_{\mathbf{w}_i} l_i(\mathbf{w}) = \begin{pmatrix} \boldsymbol{M}_{i1} & \dots & \boldsymbol{M}_{in} \end{pmatrix} \mathbf{w} + \mathbf{b}_i \tag{68}$$

which yields the following equation for the vector field:

$$\boldsymbol{v}(\mathbf{w}) = \boldsymbol{A}\mathbf{w} + \mathbf{b}, \qquad \boldsymbol{A} \triangleq \begin{pmatrix} \boldsymbol{S}_1 & \dots & \boldsymbol{M}_{1n} \\ \vdots & & \vdots \\ \boldsymbol{M}_{n1} & \dots & \boldsymbol{S}_n \end{pmatrix}, \quad \mathbf{b} \triangleq \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{pmatrix} \tag{69}$$

where $\boldsymbol{A}$ is the Jacobian of $\boldsymbol{v}$ and where we let $\boldsymbol{S}_i \triangleq \boldsymbol{M}_{ii}$.

B.2. Proofs of Nesterov's bounds for games

The proofs in this section are based on min-max problems for a class of functions* $f : \ell_2 \times \ell_2 \to \mathbb{R}$ such that

$$f(\boldsymbol{x}, \boldsymbol{y}) = c\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{y} - d_1\boldsymbol{x}^\top \boldsymbol{e_1} + d_2\boldsymbol{y}^\top \boldsymbol{e_1} + \frac{\mu_1}{2}\|\boldsymbol{x}\|^2 - \frac{\mu_2}{2}\|\boldsymbol{y}\|^2 \tag{70}$$

where $\boldsymbol{e_1}$ is a vector with a 1 in the first entry and 0 elsewhere, $c, d_1, d_2 \in \mathbb{R}$, and $\mu_1, \mu_2 \in \mathbb{R}^{++}$, with $\boldsymbol{M}$ upper bidiagonal matrix such that

$$\boldsymbol{M} = \begin{bmatrix} a_0 & a_1 & 0 & 0 & \dots \\ 0 & a_0 & a_1 & 0 & \dots \\ 0 & 0 & a_0 & a_1 & \dots \\ \vdots & & & \ddots & \ddots \end{bmatrix} \tag{71}$$

where $a_0, a_1 \neq 0$.

As Nesterov (2004), we shall assume that $\boldsymbol{x}_0, \boldsymbol{y}_0$ are initialised at 0, as otherwise we may work with $\boldsymbol{x}-\boldsymbol{x}_0$ and $\boldsymbol{y}-\boldsymbol{y}_0$ in the counterexample and perform the change of variable $\boldsymbol{x} \leftarrow \boldsymbol{x}-\boldsymbol{x}_0,\ \boldsymbol{y} \leftarrow \boldsymbol{y}-\boldsymbol{y}_0$ (which would give us zero-initialisation) and switch back at the end of the analysis.

B.2.1. On the domino argument

More about the domino argument can be found in Nesterov (2004); here, we shall give the intuition as to why it works. Let us introduce the ingredients of the update rule under our assumptions.

$$\boldsymbol{A} = \begin{pmatrix} \mu_1 & \boldsymbol{M} \\ -\boldsymbol{M}^\top & \mu_2 \end{pmatrix} \qquad \boldsymbol{A}^2 = \begin{pmatrix} \mu_1^2 - \boldsymbol{M}\boldsymbol{M}^\top & (\mu_1+\mu_2)\boldsymbol{M} \\ -(\mu_1+\mu_2)\boldsymbol{M}^\top & \mu_2^2 - \boldsymbol{M}^\top\boldsymbol{M} \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} -d_1\boldsymbol{e}_1 \\ d_2\boldsymbol{e}_2 \end{pmatrix} \tag{72}$$

$$\boldsymbol{A}\mathbf{b} = \begin{pmatrix} -\mu_1 d_1 \boldsymbol{e}_1 + d_2 \boldsymbol{M}\boldsymbol{e}_1 \\ d_1 \boldsymbol{M}^\top \boldsymbol{e}_1 + \mu_2 d_2 \boldsymbol{e}_1 \end{pmatrix} \qquad \boldsymbol{M}\boldsymbol{e}_1 = \begin{pmatrix} a_0 & 0 & \dots \end{pmatrix}^\top \qquad \boldsymbol{M}^\top\boldsymbol{e}_1 = \begin{pmatrix} a_0 & a_1 & 0 & \dots \end{pmatrix}^\top \tag{73}$$

$$\boldsymbol{A}\mathbf{w} = \begin{pmatrix} \mu_1\boldsymbol{x} + \boldsymbol{M}\boldsymbol{y} \\ -\boldsymbol{M}^\top\boldsymbol{x} + \mu_2\boldsymbol{y} \end{pmatrix} \qquad \boldsymbol{A}^2\mathbf{w} = \begin{pmatrix} (\mu_1^2 - \boldsymbol{M}\boldsymbol{M}^\top)\boldsymbol{x} + (\mu_1+\mu_2)\boldsymbol{M}\boldsymbol{y} \\ -(\mu_1+\mu_2)\boldsymbol{M}^\top\boldsymbol{x} + (\mu_2^2 - \boldsymbol{M}^\top\boldsymbol{M})\boldsymbol{y} \end{pmatrix} \tag{74}$$

B.2.1.1. One-step linear span assumption. If the algorithm follows the one-step assumption (e.g. gradient descent), which we define as

$$\mathbf{w}_t \in \mathbf{w}_0 + \mathrm{Span}\Big(\mathbf{w}_0, ..., \mathbf{w}_{t-1}, \boldsymbol{A}\mathbf{w}_0, ..., \boldsymbol{A}\mathbf{w}_{t-1}, \mathbf{b}\Big) \tag{75}$$

*As evoked in the discussion of the improved bound, for these functions, a limitation is that $\mu_1 = L_1$ and $\mu_2 = L_2$, but we were not able to find counterexamples with $L_i \neq \mu_i$ for which the bound had simple enough closed form, even when choosing terms of the form $\boldsymbol{x}^\top \boldsymbol{M}_{\boldsymbol{x}}\boldsymbol{x}, \boldsymbol{y}^\top \boldsymbol{M}_{\boldsymbol{y}}\boldsymbol{y}$ with $\boldsymbol{M}_{\boldsymbol{x}}, \boldsymbol{M}_{\boldsymbol{y}}$ bidiagonal or tridiagonal.

note that the part of $\mathbf{b}$ contributing to the update rules of both $\boldsymbol{x}$ and $\boldsymbol{y}$ will have be a vector with a single non-zero entry as its first entry, i.e. $(*, 0, ...)$. Therefore,

$$\boldsymbol{x}_t \in \text{Span}\Big((*, 0, ...), \boldsymbol{x}_0, ..., \boldsymbol{x}_{t-1}, \boldsymbol{M}\boldsymbol{y}_0, ..., \boldsymbol{M}\boldsymbol{y}_{t-1}\Big) \tag{76}$$

$$\boldsymbol{y}_t \in \text{Span}\Big((*, 0, ...), \boldsymbol{y}_0, ..., \boldsymbol{y}_{t-1}, \boldsymbol{M}^\top\boldsymbol{x}_0, ..., \boldsymbol{M}^\top\boldsymbol{x}_{t-1}\Big) \tag{77}$$

Since $(\boldsymbol{x}_0, \boldsymbol{y}_0) = 0$ but $(\boldsymbol{x}^*, \boldsymbol{y}^*) \neq 0$, we want to see, at every iteration t, how many components of $\boldsymbol{x}_t, \boldsymbol{y}_t$ have been *initialised*, i.e. received information from (which components of) past iterations and therefore could have changed from their initial values of zero. The dependence of the components of $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ on past iterates, based on the one-step linear span assumption, is summarised below:

Comp. $i = 1$ of $\boldsymbol{x}_t$: comp. 1 from const. vector, i from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$, and $i, i+1$ from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$

Comp. $i = 1$ of $\boldsymbol{y}_t$: comp. 1 from const. vector, i from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$, and i from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$

Comp. $i \geq 2$ of $\boldsymbol{x}_t$: comp. i from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$, and $i, i+1$ from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$

Comp. $i \geq 2$ of $\boldsymbol{y}_t$: comp. i from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$, and $i-1, i$ from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$

Therefore, if $(\boldsymbol{x}_0, \boldsymbol{y}_0) = 0$, it is clear that the only terms in the update rule that may initialise any new component of $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ are the constant vectors. Thus, in $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ only the first component will be initialised if we are using simultaneous first-order black box methods satisfying the one-step linear span assumption, and additionally the second component of $\boldsymbol{y}_1$ if we extended the definition of the linear span assumption to use $\boldsymbol{x}_t$ instead of $\boldsymbol{x}_{t-1}$ when computing $\boldsymbol{y}_t$.

We then move on to $(\boldsymbol{x}_2, \boldsymbol{y}_2)$ and compute from the rules above which components, i.e. values of i, can be initialised given the initialisation of the past iterates. For simultaneous methods, we see that we still cannot initialise the second component of $\boldsymbol{x}_2$ since that would require the second component of $\boldsymbol{x}_0$ or $\boldsymbol{x}_1$, or the second or third components of either $\boldsymbol{y}_0$ or $\boldsymbol{y}_1$ to have been initialised. Nevertheless, given that the second component of $\boldsymbol{y}_2$ depends on the first component of $\boldsymbol{x}_0, \boldsymbol{x}_1$, we may initialise a second component in $\boldsymbol{y}_2$. However, a third component would require either the third component of $\boldsymbol{y}_0, \boldsymbol{y}_1$ or the second or third components of $\boldsymbol{x}_0, \boldsymbol{x}_1$ to be already initialised, which is not the case. Therefore, in simultaneous one-step methods, only 1 component of $\boldsymbol{x}_2$ and 2 components of $\boldsymbol{y}_2$ will be initialised at most.

This logic is applied in table 11, which indicates the number of components in both sets of parameters that have been updated from their initial value (e.g. that are nonzero if we initialise the parameters at 0) at each iteration.

Table 11. Number of components initialised in $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ at iteration t, for methods using $\mathbf{w}_i, \boldsymbol{A}\mathbf{w}_i, \mathbf{b}$

Iteration	Simultaneous		Alt. $\boldsymbol{x}_t$ instead of $\boldsymbol{x}_{t-1}$ for $\boldsymbol{y}_t$		Alt. $\boldsymbol{y}_t$ instead of $\boldsymbol{y}_{t-1}$ for $\boldsymbol{x}_t$	
t	# dim $\boldsymbol{x}_t$	# dim $\boldsymbol{y}_t$	# dim $\boldsymbol{x}_t$	# dim $\boldsymbol{y}_t$	# dim $\boldsymbol{x}_t$	# dim $\boldsymbol{y}_t$
0	0	0	0	0	0	0
1	1	1	1	2	1	1
2	1	2	2	3	2	2
3	2	2	3	4	3	3
4	2	3	4	5	4	4

A simple proof by induction can generalise that for both alternating or simultaneous updates, at most $t+1$ components of $\boldsymbol{x}_t, \boldsymbol{y}_t$ have been initialised. The consequence is that at iteration t we have $\boldsymbol{x}_t(i) = \boldsymbol{x}_0(i), \boldsymbol{y}_t(i) = \boldsymbol{y}_0(i)$ for $i > t+1$, where $(\boldsymbol{x}_0, \boldsymbol{y}_0) = 0$. Note that this still holds if we compute elements of the span with diagonal matrices as coefficients. This can be summarised as the following.

Lemma 1 (One-step linear span domino argument)**.** *Suppose* $(\boldsymbol{x}_0, \boldsymbol{y}_0) = 0$. *Then for algorithms satisfying the one-step linear span assumption (where elements of the span may be computed using diagonal matrices as coefficients), we have*

$$\begin{aligned} \boldsymbol{x}_t(i) &= 0 \\ \boldsymbol{y}_t(i) &= 0 \end{aligned} \qquad \textit{for } i > t+1 \tag{78}$$

B.2.1.2. Two-step linear span assumption. For an algorithm satisfying the two-step assumption such as extragradient (see eq. 43 for the update rule), i.e. if we have

$$\mathbf{w}_t \in \mathbf{w}_0 + \text{Span}\Big(\mathbf{w}_0, ..., \mathbf{w}_{t-1}, \boldsymbol{A}\mathbf{w}_0, ..., \boldsymbol{A}\mathbf{w}_{t-1}, \boldsymbol{A}^2\mathbf{w}_0, ..., \boldsymbol{A}^2\mathbf{w}_{t-1}, \mathbf{b}, \boldsymbol{A}\mathbf{b}\Big) \tag{79}$$

the part of $\mathbf{b}$ and $\boldsymbol{A}\mathbf{b}$ contributing to the update rule on $\boldsymbol{x}$ will be a vector of the form $(*, 0, ...)$ and the part contributing to the update rule on $\boldsymbol{y}$ will have the form $(*, *, 0, ...)$. Therefore,

$$\boldsymbol{x}_t \in \text{Span}\Big((*, 0, 0, ...), \boldsymbol{x}_0, ..., \boldsymbol{x}_{t-1}, \boldsymbol{M}\boldsymbol{y}_0, ..., \boldsymbol{M}\boldsymbol{y}_{t-1}, \boldsymbol{M}\boldsymbol{M}^\top\boldsymbol{x}_0, ..., \boldsymbol{M}\boldsymbol{M}^\top\boldsymbol{x}_{t-1}\Big) \tag{80}$$

$$\boldsymbol{y}_t \in \text{Span}\Big((*, *, 0, ...), \boldsymbol{y}_0, ..., \boldsymbol{y}_{t-1}, \boldsymbol{M}^\top\boldsymbol{x}_0, ..., \boldsymbol{M}^\top\boldsymbol{x}_{t-1}, \boldsymbol{M}^\top\boldsymbol{M}\boldsymbol{y}_0, ..., \boldsymbol{M}^\top\boldsymbol{M}\boldsymbol{y}_{t-1}\Big) \tag{81}$$

We can see from eq. 97 ($\boldsymbol{M}^\top\boldsymbol{M}$ yields the same matrix with a_0^2 instead of $a_0^2 + a_1^2$ in the first entry) that the i-th component of $\boldsymbol{M}\boldsymbol{M}^\top\boldsymbol{x}$ depends on the $i-1, i, i+1$-th components of $\boldsymbol{x}$ for $i \geq 2$. Since only the number of initialised components will interest us, we want to see, at every iteration t, how many components of $\boldsymbol{x}_t, \boldsymbol{y}_t$ received information from past iterations and therefore could have changed from their initial values of zero. The dependence of the components of $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ on past

iterates, based on the two-step linear span assumption, is summarised below:

Comp. $i = 1$ of $\boldsymbol{x}_t$: comp. 1 from const., $i, i+1$ from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$, and $i, i+1$ from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$

Comp. $i = 1$ of $\boldsymbol{y}_t$: comp. 1 from const., $i, i+1$ from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$, and i from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$

Comp. $i = 2$ of $\boldsymbol{x}_t$: comp. $i-1$,i,$i+1$ from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$, and $i, i+1$ from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$

Comp. $i = 2$ of $\boldsymbol{y}_t$: comp. 1 from const., $i-1$,i,$i+1$ from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$, and $i-1, i$ from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$

Comp. $i > 2$ of $\boldsymbol{x}_t$: comp. $i-1$,i,$i+1$ from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$, and $i, i+1$ from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$

Comp. $i > 2$ of $\boldsymbol{y}_t$: comp. $i-1$,i,$i+1$ from $\boldsymbol{y}_0$,...,$\boldsymbol{y}_{t-1}$, and $i-1, i$ from $\boldsymbol{x}_0$,...,$\boldsymbol{x}_{t-1}$

so for the first few iterations we get Table 12.

Table 12. Number of components initialised in $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ for methods using $\mathbf{w}_i$, $\boldsymbol{A}\mathbf{w}_i$, $\boldsymbol{A}^2\mathbf{w}_i$, $\mathbf{b}$, $\boldsymbol{A}\mathbf{b}$

Iteration	Simultaneous		Alt. $\boldsymbol{x}_t$ instead of $\boldsymbol{x}_{t-1}$ for $\boldsymbol{y}_t$		Alt. $\boldsymbol{y}_t$ instead of $\boldsymbol{y}_{t-1}$ for $\boldsymbol{x}_t$	
t	# dim $\boldsymbol{x}_t$	# dim $\boldsymbol{y}_t$	# dim $\boldsymbol{x}_t$	# dim $\boldsymbol{y}_t$	# dim $\boldsymbol{x}_t$	# dim $\boldsymbol{y}_t$
0	0	0	0	0	0	0
1	1	2	1	2	2	2
2	2	3	2	3	3	3
3	3	4	3	4	4	4
4	4	5	4	5	5	5

Hence, we can prove once again by induction that in any case at iteration t we have $\boldsymbol{x}_t(i) = \boldsymbol{x}_0(i), \boldsymbol{y}_t(i) = \boldsymbol{y}_0(i)$ for $i > t+1$ and $(\boldsymbol{x}_0, \boldsymbol{y}_0) = 0$ for methods also accessing $\boldsymbol{A}^2\mathbf{w}_i$, $\boldsymbol{A}\mathbf{b}$. Here again, this still holds if we multiply the entries in our span by diagonal matrices. We can once again summarise this as a lemma.

Lemma 2 (Two-step linear span domino argument)**.** *Suppose* $(\boldsymbol{x}_0, \boldsymbol{y}_0) = 0$*. Then for algorithms satisfying the two-step linear span assumption (where elements of the span may be computed using diagonal matrices as coefficients), we have*

$$\begin{aligned} \boldsymbol{x}_t(i) &= 0 \\ \boldsymbol{y}_t(i) &= 0 \end{aligned} \qquad \textit{for } i > t+1 \tag{82}$$

B.2.2. Proof of Prop. 1

We look for stationary points $(\boldsymbol{x}^*, \boldsymbol{y}^*)$:

$$\nabla_x f(\boldsymbol{x}^*, \boldsymbol{y}^*) = c\boldsymbol{M}\boldsymbol{y}^* - d_1\boldsymbol{e_1} + \mu_1\boldsymbol{x}^* = 0 \tag{83}$$

$$\nabla_y f(\boldsymbol{x}^*, \boldsymbol{y}^*) = c\boldsymbol{M}^\top\boldsymbol{x}^* + d_2\boldsymbol{e_1} - \mu_2\boldsymbol{y}^* = 0 \tag{84}$$

Therefore, denoting $x_i = \boldsymbol{x}^*(i), y_i = \boldsymbol{y}^*(i)$, the components of stationary points satisfy the recurrence:

$$x_1 : a_0 c y_1 + a_1 c y_2 - d_1 + \mu_1 x_1 = 0 \tag{85}$$

$$y_1 : a_0 c x_1 \qquad\qquad + d_2 - \mu_2 y_1 = 0 \tag{86}$$

and for $n \geq 2$:

$$x_n : a_0 c y_n + a_1 c y_{n+1} + \mu_1 x_n = 0 \tag{87}$$

$$y_n : a_1 c x_{n-1} + a_0 c x_n - \mu_2 y_n = 0 \tag{88}$$

We can rewrite the above as

$$x_n = -a_0 \frac{c}{\mu_1} y_n - a_1 \frac{c}{\mu_1} y_{n+1} \tag{89}$$

$$y_n = a_1 \frac{c}{\mu_2} x_{n-1} + a_0 \frac{c}{\mu_2} x_n \tag{90}$$

and using eq. 90 to substitute y_n in eq. 87 we get a recurrence on x only:

$$a_0 a_1 \frac{c^2}{\mu_2} x_{n-1} + a_0^2 \frac{c^2}{\mu_2} x_n + a_1^2 \frac{c^2}{\mu_2} x_n + a_0 a_1 \frac{c^2}{\mu_2} x_{n+1} + \mu_1 x_n = 0 \tag{91}$$

which can be rewritten as

$$a_0 a_1 x_{n+1} + \left(\frac{\mu_1 \mu_2}{c^2} + a_0^2 + a_1^2 \right) x_n + a_0 a_1 x_{n-1} = 0 \tag{92}$$

The roots of the characteristic polynomial of the above linear recurrence are given by

$$\chi_\pm = \frac{-\left(\frac{\mu_1 \mu_2}{c^2} + a_0^2 + a_1^2 \right) \pm \sqrt{\left(\frac{\mu_1 \mu_2}{c^2} + a_0^2 + a_1^2 \right)^2 - 4 a_0^2 a_1^2}}{2 a_0 a_1} \tag{93}$$

and the solution to the linear recurrence is given by $x_n = C_1 \chi_+^n + C_2 \chi_-^n$ (see Brassard and Bratley (1996) for a reference on solving linear recurrences). Note that

$$\begin{aligned} \chi_\pm + 1 &= \frac{-\left(\frac{\mu_1 \mu_2}{c^2} + a_0^2 + a_1^2 \right) \pm \sqrt{\left(\frac{\mu_1 \mu_2}{c^2} + a_0^2 + a_1^2 \right)^2 - 4 a_0^2 a_1^2} + 2 a_0 a_1}{2 a_0 a_1} \\ &= \frac{-\left(\frac{\mu_1 \mu_2}{c^2} + (a_0 - a_1)^2 \right) \pm \sqrt{\left(\frac{\mu_1 \mu_2}{c^2} + a_0^2 + a_1^2 \right)^2 - 4 a_0^2 a_1^2}}{2 a_0 a_1} \end{aligned} \tag{94}$$

Suppose $a_0 a_1 > 0$. As $\frac{\mu_1 \mu_2}{c^2} > 0$, we have $\chi_- + 1 < 0$ i.e. $|\chi_-| > 1$. Similarly, if we had $a_0 a_1 < 0$ instead, we would have $\chi_- - 1 > 0$ which also yields $|\chi_-| > 1$. Therefore, χ_- is not a solution as it will not yield a $\boldsymbol{x}$ in ℓ_2. However, note that $\chi_+ \chi_- = 1$ which implies that we always have $|\chi_+| < 1$. Hence, we are only concerned with $\chi \triangleq \chi_+$. Moreover, note that the square root always exist as we

can rewrite the content of the square root to show that it is always positive:

$$\begin{aligned}\chi &= \frac{-\left(\frac{\mu_1\mu_2}{c^2}+a_0^2+a_1^2\right)+\sqrt{\left(\frac{\mu_1\mu_2}{c^2}+a_0^2+a_1^2\right)^2-4a_0^2a_1^2}}{2a_0a_1} \\ &= \frac{-\left(\frac{\mu_1\mu_2}{c^2}+a_0^2+a_1^2\right)+\sqrt{\left(\frac{\mu_1\mu_2}{c^2}\right)^2+2\frac{\mu_1\mu_2}{c^2}\left(a_0^2+a_1^2\right)+\left(a_0^2+a_1^2\right)^2-4a_0^2a_1^2}}{2a_0a_1} \\ &= \frac{-\left(\frac{\mu_1\mu_2}{c^2}+a_0^2+a_1^2\right)+\sqrt{\left(\frac{\mu_1\mu_2}{c^2}\right)^2+2\frac{\mu_1\mu_2}{c^2}\left(a_0^2+a_1^2\right)+\left(a_0^2-a_1^2\right)^2}}{2a_0a_1}\end{aligned} \tag{95}$$

In order to simplify the results, we let $a_0 = -a_1 = 1$ and we get:

$$\chi = \left(\frac{\mu_1\mu_2}{2c^2}+1\right)-\sqrt{\left(\frac{\mu_1\mu_2}{2c^2}\right)^2+\frac{\mu_1\mu_2}{c^2}} \tag{96}$$

One may note that $L_{12} = c\sqrt{\rho(\boldsymbol{M}\boldsymbol{M}^\top)}$. As we have

$$\boldsymbol{M}\boldsymbol{M}^\top = \begin{bmatrix} a_0^2+a_1^2 & a_0a_1 & 0 & 0 & \dots \\ a_0a_1 & a_0^2+a_1^2 & a_0a_1 & 0 & \dots \\ 0 & a_0a_1 & a_0^2+a_1^2 & a_0a_1 & \dots \\ \vdots & & \ddots & \ddots & \ddots \end{bmatrix} = \begin{bmatrix} 2 & -1 & 0 & 0 & \dots \\ -1 & 2 & -1 & 0 & \dots \\ 0 & -1 & 2 & -1 & \dots \\ \vdots & & \ddots & \ddots & \ddots \end{bmatrix} \tag{97}$$

we note that $\boldsymbol{M}\boldsymbol{M}^\top$ is a tridiagonal Toeplitz matrix, for which the upper end of the spectrum is given by (see Theorem 7.20 of Douglas (2012))

$$\begin{aligned}\sup|\sigma(\boldsymbol{M}\boldsymbol{M}^\top)| &= \operatorname{ess}\sup_{\theta\in[0,2\pi)}\left(a_0a_1e^{-i\theta}+(a_0^2+a_1^2)+a_0a_1e^{i\theta}\right) \\ &= \operatorname{ess}\sup_{\theta\in[0,2\pi)}\left(2a_0a_1\cos\theta+(a_0^2+a_1^2)\right) = 4\end{aligned} \tag{98}$$

and therefore $L_{12} = 2c$. Defining the condition number as

$$\kappa = \frac{L_{12}}{\sqrt{\mu_1\mu_2}} \tag{99}$$

to retrieve the condition number from the upper bound literature, we get that

$$\begin{aligned}\chi &= \left(\frac{2}{\kappa^2}+1\right)-\sqrt{\frac{4}{\kappa^4}+\frac{4}{\kappa^2}} \\ &= \left(\frac{2}{\kappa^2}+1\right)-\frac{2}{\kappa^2}\sqrt{\kappa^2+1} \\ &= 1-2\frac{\sqrt{\kappa^2+1}-1}{(\kappa^2+1)-1} \\ &= 1-\frac{2}{\sqrt{\kappa^2+1}+1}\end{aligned} \tag{100}$$

Going back to the recurrence, and given that the recurrence on y_n can be shown to be the same as eq. 92, we get that if $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is a stationary point of f in $\ell_2 \times \ell_2$, then

$$\boldsymbol{x}^*(i) = x_i = c_1 \chi^i \tag{101}$$

$$\boldsymbol{y}^*(i) = y_i = c_2 \chi^i \tag{102}$$

where c_1, c_2 can be determined from the initial conditions given in eq. 85 and 86. Using the domino argument, which yields that $\forall i > t+1, \boldsymbol{x}_t(i) = 0$, we get that the distance to the optimum of $\boldsymbol{x}$ is given by

$$\begin{aligned}
\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2 = \sum_{i=1}^{t+1} (\boldsymbol{x}_t(i) - \boldsymbol{x}^*(i))^2 + \sum_{i=t+2}^{\infty} (\boldsymbol{x}_t(i) - \boldsymbol{x}^*(i))^2 &\geq \sum_{i=t+2}^{\infty} \left(\boldsymbol{x}^*(i)\right)^2 \\
&= c_1^2 \sum_{i=t+2}^{\infty} \chi^{2i} \\
&= c_1^2 \sum_{i=1}^{\infty} \chi^{2(i+t+1)} \\
&= \chi^{2(t+1)} \|\boldsymbol{x}^*\|^2
\end{aligned} \tag{103}$$

Similarly, we can show that $\|\boldsymbol{y}_t - \boldsymbol{y}^*\|^2 \geq \chi^{2(t+1)} \|\boldsymbol{y}^*\|^2$.
Changing back our variables to $\boldsymbol{x} \to \boldsymbol{x} - \boldsymbol{x}_0, \boldsymbol{y} \to \boldsymbol{y} - \boldsymbol{y}_0$ yields the bound for arbitrary initialisation.

B.2.3. Proof of Thm. 1

If one computes μ_{12} for the function used in the previous bound, it becomes clear that $\mu_{12} = 0$. As such, it is not surprising that the bound failed to hold vs the upper bound of negative momentum on bilinear games with $\mu_{12} > 0$: the previous bound failed to be general enough because the example has the worst possible value of μ_{12}. An important note, however, is that the previous bound may still hold in finite dimensions if we only used it to lower bound the rate of convergence of games with $\mu_{12} = 0$, but it can easily be checked that the rate in the improved bound with $\mu_{12} = 0$ reduces to the first bound.
In order to address this, we will pick values of a_0 and a_1 that allow the counterexample to handle any value of μ_{12}. The proof of the improved domino bound follows the same line of argumentation as the proof of the first bound. We resume from eq. 98, and set $c = 1$, and suppose that $a_1 < 0, a_0 > 0$ such that $|a_1| \leq a_0$. Theorem 7.20 of Douglas (2012) yields that:

$$\max \sigma(\boldsymbol{M}\boldsymbol{M}^\top) = a_0^2 + a_1^2 - 2a_0 a_1 = (a_0 - a_1)^2 \tag{104}$$

$$\min \sigma(\boldsymbol{M}\boldsymbol{M}^\top) = a_0^2 + a_1^2 + 2a_0 a_1 = (a_0 + a_1)^2 \tag{105}$$

Thus, we have $\mu_{12}^2 = (a_0 + a_1)^2$, $L_{12}^2 = (a_0 - a_1)^2$ and since we assumed $|a_1| \leq a_0$, we get that $\mu_{12} = a_0 + a_1, L_{12} = a_0 - a_1$ which allows us to choose a_0, a_1 to make μ_{12}, L_{12} appear in the

bound:

$$a_0 = \frac{L_{12} + \mu_{12}}{2} \tag{106}$$

$$a_1 = \frac{\mu_{12} - L_{12}}{2} \tag{107}$$

Noting further that $a_0^2 + a_1^2 = \frac{L_{12}^2 + \mu_{12}^2}{2}$, $a_0^2 - a_1^2 = \mu_{12} L_{12}$, we have that

$$\begin{aligned}
\chi &= \frac{-\left(\frac{\mu_1\mu_2}{c^2} + a_0^2 + a_1^2\right) + \sqrt{\left(\frac{\mu_1\mu_2}{c^2}\right)^2 + 2\frac{\mu_1\mu_2}{c^2}\left(a_0^2 + a_1^2\right) + \left(a_0^2 - a_1^2\right)^2}}{2a_0 a_1} \\
&= \frac{-\left(\mu_1\mu_2 + \frac{L_{12}^2 + \mu_{12}^2}{2}\right) + \sqrt{\left(\mu_1\mu_2\right)^2 + 2\mu_1\mu_2\left(\frac{L_{12}^2 + \mu_{12}^2}{2}\right) + \left(\mu_{12} L_{12}\right)^2}}{\frac{\mu_{12}^2 - L_{12}^2}{2}} \\
&= \frac{-\left(2\mu_1\mu_2 + L_{12}^2 + \mu_{12}^2\right) + 2\sqrt{\left(\mu_1\mu_2\right)^2 + \mu_1\mu_2\left(L_{12}^2 + \mu_{12}^2\right) + \left(\mu_{12} L_{12}\right)^2}}{\mu_{12}^2 - L_{12}^2 + \mu_1\mu_2 - \mu_1\mu_2}
\end{aligned} \tag{108}$$

Letting $d_\mu = \mu_1\mu_2 + \mu_{12}^2$, $d_L = \mu_1\mu_2 + L_{12}^2$,

$$\begin{aligned}
\chi &= \frac{-\left(d_\mu + d_L\right) + 2\sqrt{\mu_1\mu_2\left(\mu_1\mu_2 + L_{12}^2\right) + \mu_{12}^2\left(\mu_1\mu_2 + L_{12}^2\right)}}{d_\mu - d_L} \\
&= \frac{\left(d_\mu + d_L\right) - 2\sqrt{d_\mu d_L}}{d_L - d_\mu} \\
&= \frac{\left(\sqrt{d_L} - \sqrt{d_\mu}\right)^2}{\sqrt{d_L}^2 - \sqrt{d_\mu}^2} \\
&= \frac{\sqrt{d_L} - \sqrt{d_\mu}}{\sqrt{d_L} + \sqrt{d_\mu}} \\
&= 1 - \frac{2}{\sqrt{\frac{d_L}{d_\mu}} + 1}
\end{aligned} \tag{109}$$

Letting $\kappa = \frac{d_L}{d_\mu} = \frac{L_{12}^2 + \mu_1\mu_2}{\mu_{12}^2 + \mu_1\mu_2}$ and proceeding as in the proof of the previous bound with the new value of χ yields Thm. 1. Note that as promised, this rate reduces to that of the first bound if $\mu_{12} = 0$:

$$\begin{aligned}
1 - \frac{2}{\sqrt{\frac{d_L}{d_\mu}} + 1} &= 1 - \frac{2}{\sqrt{\frac{L_{12}^2 + \mu_1\mu_2}{\mu_1\mu_2}} + 1} \\
&= 1 - \frac{2}{\sqrt{\kappa_{old}^2 + 1} + 1}
\end{aligned} \tag{110}$$

$$\tag{111}$$

where κ_{old} is the condition number of eq. 99.

B.3. Proofs of p-SCLI-n

B.3.1. Proof of Prop. 2

In this section, we follow Arjevani et al. (2016) to derive results for the p-SCLI-n methods. First, we reproduce several definitions and theorems that are proven in Arjevani et al. (2016) and that apply directly to the generalisation. Here, $\boldsymbol{A}$ will denote the Jacobian of some quadratic game with $f_{\boldsymbol{A},\mathbf{b}} \in \mathcal{Q}^{d_1,\dots,d_n}$ such that 0 is not in the spectrum of $\boldsymbol{A}$.

Definition 4 (Characteristic polynomial of a p-SCLI-n)**.** *Let $\mathcal{A}$ be a p-SCLI-n optimisation algorithm with coefficient matrices C_i as defined in def. 2. Then for $\boldsymbol{X} \in \mathbb{R}^{d\times d}$, the characteristic polynomial of $\mathcal{A}$ is given by*

$$\mathcal{L}(\lambda,\boldsymbol{X}) \triangleq \boldsymbol{I}_d\lambda^p - \sum_{i=0}^{p-1} \mathbb{E}C_i(\boldsymbol{X})\lambda^i \tag{112}$$

and its root radius is

$$\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{X})) = \rho(\det \mathcal{L}(\lambda,\boldsymbol{X})) = \max\left\{|\lambda| \;\middle|\; \det \mathcal{L}(\lambda,\boldsymbol{X}) = 0\right\}$$

Theorem 5 (Consistency - characteristic polynomial (Based on Theorem 5 of Arjevani et al. (2016))). *A p-SCLI-n algorithm $\mathcal{A}$ with characteristic polynomial $\mathcal{L}(\lambda, \boldsymbol{X})$ and inversion matrix $N(\boldsymbol{X})$ is consistent with respect to $\boldsymbol{A}$ if and only if the following two conditions hold:*

$$1.\ \mathcal{L}(1,\boldsymbol{A}) = -\mathbb{E}N(\boldsymbol{A})\boldsymbol{A} \tag{113}$$

$$2.\ \rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A})) < 1 \tag{114}$$

We may rephrase theorem 13 of Arjevani et al. (2016) (and lower bound t^{m-1} by 1 since $m \in \mathbb{N}$) as the following to use the root radius of the characteristic polynomial to show linear rates:

Theorem 6 (Based on Theorem 13 of Arjevani et al. (2016)). *If $\boldsymbol{A}$ is the Jacobian of a quadratic game and $\mathcal{A}$ is a p-SCLI-n, there exists an initialisation point $\mathbf{w}_0 \in \mathbb{R}^d$ such that*

$$\max_{i=0,\dots,p-1} \left\|\mathbb{E}\mathbf{w}^{t+i} - \mathbb{E}\mathbf{w}^*\right\| \in \Omega(\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))^t) \tag{115}$$

In other words, this means that $\mathcal{A}$ cannot converge on $f_{\boldsymbol{A},\mathbf{b}}$ with linear rate faster than $\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))$, up to a constant. As Arjevani et al. (2016) argue, in both deterministic and stochastic settings, a lower bound on $\left\|\mathbb{E}\left[\mathbf{w}^t - \mathbf{w}^*\right]\right\|^2$ implies[†] a lower bound on $\mathbb{E}\left\|\mathbf{w}^t - \mathbf{w}^*\right\|^2$, since

$$\mathbb{E}\left[\left\|\mathbf{w}^t - \mathbf{w}^*\right\|^2\right] = \mathbb{E}\left[\left\|\mathbf{w}^t - \mathbb{E}\mathbf{w}^t\right\|^2\right] + \left\|\mathbb{E}\left[\mathbf{w}^t - \mathbf{w}^*\right]\right\|^2 \tag{116}$$

We can now focus on finding a lower bound on $\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))$.

[†]Note that since we only use in this paper a lower bound on the second term of the right hand-side of the equation to bound the left hand-side, one may derive in stochastic settings tighter lower bounds than the ones presented in this paper by factoring in the first term of the right hand-side. We leave this as future work.

Proposition 3. *Let $\mathcal{A}$ be a p-SCLI-n optimisation algorithm with inversion matrix $N(\boldsymbol{X})$ that is consistent with respect to $\boldsymbol{A}$. Then,*

$$\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A})) \geq \max_{j=1,\dots,d} |\sqrt[p]{|\sigma_j(-\mathbb{E}[N(\boldsymbol{A})]\boldsymbol{A})|} - 1| \tag{117}$$

where the $\sigma_j(-\mathbb{E}[N(\boldsymbol{A})]\boldsymbol{A})$ are elements of the spectrum (eigenvalues) of $-\mathbb{E}[N(\boldsymbol{A})]\boldsymbol{A}$.

B.3.1.1. Proof of Prop. 3. Our proof starts exactly as the one presented by Arjevani et al. (2016) for the $n = 1$ particular case, where the authors assume that $\boldsymbol{A}$ is symmetric with strictly positive spectrum. However, we will generalise the proof to cover non-symmetric matrices and matrices that may not have strictly positive spectrum, since the Jacobian of a quadratic n-player game generally does not have these properties.

Let $\mathcal{A}$ be a deterministic p-SCLI-n optimisation algorithm with characteristic polynomial $\mathcal{L}(\lambda,\boldsymbol{X})$ and inversion matrix $N(\boldsymbol{X})$, and $f_{A,\mathbf{b}}(\mathbf{w}) \in \mathcal{Q}^{d_1,\dots,d_n}$ represent a quadratic n-player game. Since $\mathcal{A}$ is p-SCLI-n, its (expected) coefficient matrices $\mathbb{E}C_i$ evaluated on $\boldsymbol{A}$ are simultaneously triangularisable, so $\exists \boldsymbol{Q} \in \mathbb{R}^{d\times d}$ such that for $i = 0, ..., p-1$, we have

$$\boldsymbol{T}_i \triangleq \boldsymbol{Q}^{-1}\mathbb{E}C_i(\boldsymbol{A})\boldsymbol{Q} \tag{118}$$

where $\boldsymbol{T}_i$ is triangular. Thus,

$$\begin{aligned}\det \mathcal{L}(\lambda,\boldsymbol{A}) &= \det\left(\boldsymbol{Q}^{-1}\mathcal{L}(\lambda,\boldsymbol{A})\boldsymbol{Q}\right)\\ &= \det\left(\boldsymbol{I}_d\lambda^p - \sum_{i=0}^{p-1}\boldsymbol{T}_i\lambda^i\right)\end{aligned} \tag{119}$$

Since $\boldsymbol{I}_d\lambda^p - \sum_{i=0}^{p-1}\boldsymbol{T}_i\lambda^i$ is a upper triangular matrix, its determinant is given by

$$\det \mathcal{L}(\lambda,\boldsymbol{A}) = \prod_{j=1}^{d}\ell_j(\lambda) \tag{120}$$

where

$$\ell_j(\lambda) = \lambda^p - \sum_{i=0}^{p-1}\sigma_j^i\lambda^i \tag{121}$$

and where $\sigma_1^i, \dots, \sigma_d^i$, $i = 0, \dots, p-1$ denote the elements on the diagonal of $\boldsymbol{T}_i$, which are just the eigenvalues of $\mathbb{E}C_i$ ordered according to $\boldsymbol{Q}$. Hence, the root radius of the characteristic polynomial of $\mathcal{A}$ is

$$\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A})) = \max\left\{|\lambda| \;\middle|\; \ell_j(\lambda) = 0 \text{ for some } j = 1,...,d\right\} \tag{122}$$

On the other hand, by consistency condition (113) we get that for all $j = 1,...,d$

$$\ell_j(1) = \sigma_j\left(\mathcal{L}(1,\boldsymbol{A})\right) = \sigma_j\left(-\mathbb{E}[N(\boldsymbol{A})]\boldsymbol{A}\right) \tag{123}$$

In the case of p-SCLI-1, the authors prove their Corollary 7 (i.e. our prop. 3 without taking the modulus of the eigenvalues) by using a lemma (see Lemma 6 in Arjevani et al. (2016)) that gives a lower bound on each $\rho(\ell_j(\lambda))$ by using the sign of $\ell_j(1) = \sigma_j\left(-\mathbb{E}[N(\boldsymbol{A})]\boldsymbol{A}\right)$. Lemma 6 of Arjevani et al. (2016) is proven using the following lemma, which we can in fact use to handle arbitrary eigenvalues (e.g. complex or negative).

Lemma 3 (Lemma 15 of Arjevani et al. (2016))**.** *Let $q_r^*(z) \triangleq \left(z - (1 - \sqrt[p]{r})\right)^p$ where r is some non-negative constant. Suppose $q(z)$ is a monic polynomial of degree p with complex coefficients. Then,*

$$\rho(q(z)) \leq |\sqrt[p]{|q(1)|} - 1| \iff q(z) = q^*_{|q(1)|}(z)$$

The proof of the lemma can be found in Arjevani et al. (2016). Here, we can use the lemma directly on each ℓ_j with $q = \ell_j$ and $r = |q(1)| = |\ell_j(1)| = |\sigma_j\left(-\mathbb{E}[N(\boldsymbol{A})]A\right)|$. Indeed, since $r \geq 0$,

- if $q(z) = q_r^*(z) = (z - (1 - \sqrt[p]{r}))^p$ then clearly $\rho(q(z)) = |1 - \sqrt[p]{r}|$
- if $q(z) \neq q_r^*(z)$, then we have $\rho(q(z)) > |\sqrt[p]{|q(1)|} - 1|$

Which implies that for any j we have $\rho(\ell_j(\lambda)) \geq |\sqrt[p]{|\ell_j(1)|} - 1| = |\sqrt[p]{|\sigma_j\left(-\mathbb{E}[N(\boldsymbol{A})]\boldsymbol{A}\right)|} - 1|$. Using this in eq. 122 yields

$$\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A})) \geq \max_{j=1,\dots,d} |\sqrt[p]{|\sigma_j(-\mathbb{E}[N(\boldsymbol{A})]\boldsymbol{A})|} - 1| \tag{124}$$

B.3.1.2. Deriving the optimal ρ for scalar inversion matrices. We are now ready to obtain the general lower bound. Consider $f_{\boldsymbol{A},\mathbf{b}} \in \mathcal{Q}^{d_1,\dots,d_n}$ with $0 \notin \sigma(\boldsymbol{A})$ and a consistent p-SCLI-n algorithm $\mathcal{A}$. Let $\mu = \min|\sigma(\boldsymbol{A})|$, $L = \max|\sigma(\boldsymbol{A})|$ where $\sigma(\boldsymbol{A})$ is the spectrum of $\boldsymbol{A}$. For a scalar inversion matrix i.e. $\mathbb{E}[N(\boldsymbol{A})] = \nu$ we have from eq. 124:

$$\begin{aligned}\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A})) &\geq \max_{j=1,\dots,d} |\sqrt[p]{|\sigma_j(-\mathbb{E}[N(\boldsymbol{A})]\boldsymbol{A})|} - 1| = \max_{j=1,\dots,d} |\sqrt[p]{|\nu\sigma_j(\boldsymbol{A})|} - 1| \\ &= \max\left\{|\sqrt[p]{|\nu|\mu} - 1|, |\sqrt[p]{|\nu|L} - 1|\right\}\end{aligned} \tag{125}$$

Note that consistency (eq. 114) constrains $\nu \in \left(\frac{-2^p}{L}, \frac{2^p}{L}\right) \setminus \{0\}$. We proceed as Arjevani et al. (2016) in the p-SCLI-1 case, and study the ranges of $|\nu|$ by using $\max(a,b) = \frac{a+b+|a-b|}{2}$ to obtain table 13.

Table 13. Lower bound for ρ by subranges of $|\nu|$ and minimiser $|\nu^*|$

	$\sqrt[p]{\lvert\nu\rvert\mu} - 1 < 0$			$\sqrt[p]{\lvert\nu\rvert\mu} - 1 \geq 0$		
	Range	Minimiser	Bound	Range	Minimiser	Bound
$\sqrt[p]{\lvert\nu\rvert L} - 1 \leq 0$	$(0,1/L]$	$1/L$	$1 - \sqrt[p]{\frac{\mu}{L}}$		N/A	
$\sqrt[p]{\lvert\nu\rvert L} - 1 > 0$	$(1/L,1/\mu)$	$\left(\frac{2}{\sqrt[p]{L}+\sqrt[p]{\mu}}\right)^p$	$\frac{\sqrt[p]{L/\mu}-1}{\sqrt[p]{L/\mu}+1}$	$[1/\mu, 2^p/L)$	$1/\mu$	$\sqrt[p]{\frac{L}{\mu}} - 1$

Note that case 3 requires $p > \log_2 L/\mu$. Hence,

$$\rho \geq \min\left\{1 - \sqrt[p]{\frac{\mu}{L}}, \frac{\sqrt[p]{L/\mu} - 1}{\sqrt[p]{L/\mu} + 1}, \sqrt[p]{\frac{L}{\mu}} - 1\right\} = \frac{\sqrt[p]{L/\mu} - 1}{\sqrt[p]{L/\mu} + 1} \tag{126}$$

where $\mu = \min|\sigma(\boldsymbol{A})|$, $L = \max|\sigma(\boldsymbol{A})|$.

B.3.2. Finding a suitably hard example for 2-player with $d_1 = d_2$

We now only need to find a hard counterexample. We present the argument for $d_1 = d_2 = 2$, which can easily be generalised for arbitrary d. Consider the matrix

$$\boldsymbol{A} = \begin{pmatrix} \mu_1 & 0 & \mu_{12} & 0 \\ 0 & L_1 & 0 & L_{12} \\ -\mu_{12} & 0 & \mu_2 & 0 \\ 0 & -L_{12} & 0 & L_2 \end{pmatrix} \tag{127}$$

corresponding to the Jacobian of a quadratic game in $\mathcal{Q}^{d_1,d_2}$.
First we compute the characteristic polynomial of A, using the formula for the determinant of a block matrix (see Zhang (2005, Section 0.3) for instance):

$$\det(XI - A) = \det\begin{pmatrix} X - \mu_1 & 0 & -\mu_{12} & 0 \\ 0 & X - L_1 & 0 & -L_{12} \\ \mu_{12} & 0 & X - \mu_2 & 0 \\ 0 & L_{12} & 0 & X - L_2 \end{pmatrix} \tag{128}$$

$$= \det\left(\begin{pmatrix} (X - \mu_1)(X - \mu_2) & 0 \\ 0 & (X - L_1)(X - L_2) \end{pmatrix} + \begin{pmatrix} \mu_{12}^2 & 0 \\ 0 & L_{12}^2 \end{pmatrix}\right) \tag{129}$$

$$= (X^2 - (\mu_1 + \mu_2)X + \mu_1\mu_2 + \mu_{12}^2)(X^2 - (L_1 + L_2)X + L_1L_2 + L_{12}^2) \tag{130}$$

The discriminants of these two quadratic equations are, respectively:

$$\Delta_\mu = (\mu_1 + \mu_2)^2 - 4(\mu_1\mu_2 + \mu_{12}^2) = (\mu_1 - \mu_2)^2 - 4\mu_{12}^2 \tag{131}$$

$$\Delta_L = (L_1 + L_2)^2 - 4(L_1L_2 + L_{12}^2) = (L_1 - L_2)^2 - 4L_{12}^2 \tag{132}$$

which yields the following eigenvalues:

$$\begin{aligned} \lambda_{\mu\pm} &= \frac{\mu_1 + \mu_2}{2} \pm \sqrt{\left(\frac{\mu_1 - \mu_2}{2}\right)^2 - \mu_{12}^2} \\ \lambda_{L\pm} &= \frac{L_1 + L_2}{2} \pm \sqrt{\left(\frac{L_1 - L_2}{2}\right)^2 - L_{12}^2} \end{aligned} \tag{133}$$

We distinguish four cases, which are presented in the following table:

Table 14. Lower bounds on the condition number

	$\Delta_\mu < 0$	$\Delta_\mu \geq 0$
$\Delta_L < 0$	$\kappa = \sqrt{\frac{L_1 L_2 + L_{12}^2}{\mu_1 \mu_2 + \mu_{12}^2}}$	$\kappa \geq 2\frac{\sqrt{L_1 L_2 + L_{12}^2}}{\mu_1 + \mu_2 - \sqrt{\Delta_\mu}}$
$\Delta_L \geq 0$	$\kappa \geq \frac{1}{2}\frac{L_1 + L_2 + \sqrt{\Delta_L}}{\sqrt{\mu_1 \mu_2 + \mu_{12}^2}}$	$\kappa \geq \frac{L_1 + L_2 + \sqrt{\Delta_L}}{\mu_1 + \mu_2 - \sqrt{\Delta_\mu}}$

where we used that $\kappa = \frac{\max|\sigma(\boldsymbol{A})|}{\min|\sigma(\boldsymbol{A})|}$.
We now discuss these four cases:

- If $\Delta_\mu < 0$ and $\Delta_L < 0$, we have that

$$\begin{aligned} |\lambda_{\mu\pm}| &= \left| \frac{\mu_1 + \mu_2}{2} \pm i\sqrt{\mu_{12}^2 - \left(\frac{\mu_1 - \mu_2}{2}\right)^2} \right| \\ &= \sqrt{\mu_1 \mu_2 + \mu_{12}^2} \end{aligned} \tag{134}$$

  Similarly we get

$$|\lambda_{L\pm}| = \sqrt{L_1 L_2 + L_{12}^2} \tag{135}$$

  Clearly then $\min|\sigma(\boldsymbol{A})| = |\lambda_{\mu\pm}|$ and $\max|\sigma(\boldsymbol{A})| = |\lambda_{L\pm}|$, which yields $\kappa = \sqrt{\frac{L_1 L_2 + L_{12}^2}{\mu_1 \mu_2 + \mu_{12}^2}}$.
- If $\Delta_\mu \geq 0$ and $\Delta_L \geq 0$, λ_{L+}, λ_{L-}, $\lambda_{\mu+}$ and $\lambda_{\mu-}$ are all real. We have that,

$$\lambda_{\mu-} \geq \min|\sigma(\boldsymbol{A})|, \quad \text{and} \quad \lambda_{L+} \leq \max|\sigma(\boldsymbol{A})|, \tag{136}$$

  which yields the result.
- If $\Delta_\mu < 0$ and $\Delta_L \geq 0$, it holds that,

$$|\lambda_{\mu\pm}| = \min|\sigma(\boldsymbol{A})|, \quad \text{and} \quad \lambda_{L+} \leq \max|\sigma(\boldsymbol{A})|, \tag{137}$$

  from which we obtain the result.
- Similarly, if $\Delta_\mu \geq 0$ and $\Delta_L < 0$, it holds that,

$$\lambda_{\mu-} \geq \min|\sigma(\boldsymbol{A})|, \quad \text{and} \quad |\lambda_{L\pm}| = \max|\sigma(\boldsymbol{A})|. \tag{138}$$

One could wonder whether our lower bounds on κ when at least one of the discriminant is non-negative are actually equalities. We provide an example showing that it is not the case when $\Delta_L \geq 0$ and $\Delta\mu \geq 0$. A similar one can be found when $\Delta_L < 0$ and $\Delta\mu \geq 0$.
Take $\mu_{12} = 0$ and $L_{12} = \frac{|L_1 - L_2|}{2}$. Then $\Delta_L \geq 0$ and $\Delta\mu \geq 0$. Then,

$$\begin{aligned} \lambda_{\mu+} &= \frac{\mu_1 + \mu_2}{2} + \sqrt{\left(\frac{\mu_1 - \mu_2}{2}\right)^2 - \mu_{12}^2} = \max(\mu_1, \mu_2) \\ \lambda_{L\pm} &= \frac{L_1 + L_2}{2} \pm \sqrt{\left(\frac{L_1 - L_2}{2}\right)^2 - L_{12}^2} = \frac{L_1 + L_2}{2}. \end{aligned} \tag{139}$$

Choose $\mu_1 = L_1$, $\mu_2 = L_2$ and $L_1 \neq L_2$. Then $\lambda_{\mu+} > \lambda_{L\pm}$. However we have $\lambda_{\mu-} = \min|\sigma(A)|$ and so in this case $\kappa = \lambda_{\mu+}/\lambda_{\mu-}$.

B.4. Lower bounds on the iteration complexity

The lower bounds on the distance of the iterates to a minimiser also yield lower bounds on the number of iterations required to reach a maximum error ϵ on the iterates (iteration complexity).

p-SCLI-n case: For the p-SCLI-n bound, suppose $\max_{i=0,\dots,p-1}\left\|\mathbb{E}\mathbf{w}^{t+i} - \mathbb{E}\mathbf{w}^*\right\| < \epsilon$. Then from Theorem 6, we know that for some strictly positive real C,

$$\begin{aligned} C\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))^t \leq \epsilon &\implies \rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))^t \leq \frac{\epsilon}{C} \\ &\implies t\log\left(\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))\right) \leq \log\left(\frac{\epsilon}{C}\right) \\ &\implies t\log\left(\frac{1}{\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))}\right) \geq \log\left(\frac{C}{\epsilon}\right) \end{aligned} \tag{140}$$

Noting that $\log(x) \leq x-1$ for $x > 0$, we get that $\log\left(\frac{1}{\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))}\right) \leq \frac{1-\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))}{\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))}$ and hence,

$$t\frac{1-\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))}{\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))} \geq t\log\left(\frac{1}{\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))}\right) \geq \log\left(\frac{C}{\epsilon}\right) \tag{141}$$

Therefore, we get that

$$t \geq \frac{\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))}{1-\rho_\lambda(\mathcal{L}(\lambda,\boldsymbol{A}))}\log\left(\frac{C}{\epsilon}\right) \tag{142}$$

where one may use Prop. 2 to get

$$t \geq \left(\frac{\sqrt[p]{\kappa}-1}{2}\right)\log\left(\frac{C}{\epsilon}\right) \tag{143}$$

where κ is given as in Prop. B.3.1. One may also use in the 2-player case the κ given in Table 1.

2-player domino bound: For the two player bound stemming from the domino argument, it is equally straightforward to establish from Thm. 1 that given an upper bound ϵ on $\|(\boldsymbol{x}_t,\boldsymbol{y}_t) - (\boldsymbol{x}^*,\boldsymbol{y}^*)\|$, then using that $1-\frac{2}{\kappa+1} \geq \exp\left(-\frac{2}{\kappa-1}\right)$,

$$\exp\left(-\frac{2}{\kappa-1}\right)^{t+1}\cdot\|(\boldsymbol{x}_0,\boldsymbol{y}_0) - (\boldsymbol{x}^*,\boldsymbol{y}^*)\| \leq \epsilon \implies t \geq \frac{\kappa-1}{2}\log\left(\frac{\|(\boldsymbol{x}_0,\boldsymbol{y}_0) - (\boldsymbol{x}^*,\boldsymbol{y}^*)\|}{\epsilon}\right) - 1 \tag{144}$$

where κ is defined as in Thm. 1.

Chapter C

Supplementary Material For the Second Article

The code can be found at https://github.com/AIproj/Towards-Out-of-Distribution-Adversarial-Robustness.

C.1. More on methodology

C.1.1. Motivation

Benchmark design: we use a similar benchmark as (Tramèr and Boneh, 2019) and (Maini et al., 2020), with the addition of AutoAttack. Furthermore, to empirically support our claims that REx improves worst case robustness, we additionally consider ensembles of seen, unseen, and all attacks. We consider worst-case ensembles because for real-world robustness, an attacker may try several attacks until they succeed. The evaluation against an ensemble of seen attacks is to support our first claim, i.e. that REx improves multi-perturbation robustness, and the consideration of unseen attacks supports the second claim: that REx generalises to attacks that were not seen during training.
Why weak attacks: weak attacks capture a signal that would be missed otherwise. Indeed, as argued in the discussion of our results, REx does improve significantly robustness against strong attacks and even unbounded attacks (which can't be ignored in the general context of robustness until out-of-distribution detectors are perfectly able to capture shifts beyond the typical balls); however, REx often lowers the accuracy slightly on the weakest attacks. In other words, weak attacks allow us to capture a trade-off in using REx, which is important information for readers. Not having those evaluations would suggest that REx is an improvement in every setting.
Unbounded attacks: most work in the literature focuses on bounded attacks. This is because bounded attacks constrain the strength of attacks, and have long been needed to obtain non-trivial robustness results. In this work, we consider several of the most commonly used bounded attacks, and show the improvement yielded by REx on the ones affecting the accuracy of the model the most. However, at deployment, an attacker might use some unbounded attacks, especially if they

are not perceptible. We encourage the reader to decide from Fig. 26 and 27 whether they consider the corruption to be perceptible by a human, in spite of being out of the $\epsilon = 0.5$ L_2 ball. In the absence of algorithms able to detect perturbations larger than the usual choices of ϵ, we argue that non-visually perceptible adversarial attacks, regardless of being bounded or not, are of particular concern. Therefore, we additionally consider an unbounded attack (CW) to show that REx provides benefits even in that less-studied case. Note that we find that REx also improves robustness when bounding CW by rejecting examples out of the L_2 ball of radius $\epsilon = 0.5$ on CIFAR10.

Keeping "overtuned" attacks: in our tables of results, we refer to some attacks are "overtuned". What is meant is that after choosing an alternative tuning of those attacks, all models mostly failed against them. For example, for $P^{\bullet}_{\infty}$ on MNIST, this is done by choosing an $\epsilon = 0.4$ instead of the commonly used value of 0.3. Since we already provide a significant number of evaluations throughout this work, we decided to report those overtuned attacks to highlight two points. First, the models still have weaknesses, and as seen in Fig. 26 and 27, adversarial examples produced by the overtuned attacks are not necessarily perceptible, so a truly robust model should defend against them. Second, as argued in the discussion, these attacks, along with the AutoAttack evaluation, highlight that REx does not rely on gradient obfuscation (Athalye et al., 2018).

Ablation on hyperparameter optimisation: we perform an ablation on hyperparameter optimisation on CIFAR10 in Sec. C.2.2. This is done in order to highlight several points. First, that without hyperparameter optimisation, REx-based defenses may outperform hyperparameter-optimised baselines. Second, as argued in the discussion of Sec. 6.4.4, in the specific case of the MSD baseline, while the use of REx on that baseline still yields improvements over the baseline, it is slightly less pronounced than without hyperparameter optimisation.

Max baseline: we choose not to include the Max baseline from (Tramèr and Boneh, 2019) since Tables 3 and 4 of (Maini et al., 2020) have evaluations on a very similar benchmark. Note that Maini et al. (2020) find on CIFAR10 that Avg performs significantly better than Max. Moreover, on both MNIST and CIFAR10, Maini et al. (2020) find that MSD performs better than both Avg and Max anyway, so improving on MSD with REx implies that MSD+REx would outperform Max on both datasets.

C.1.2. Attack tunings

Using Advertorch's and Croce's implementation of AutoAttack's* parameter names, we report the attacks' tuning here.

C.1.2.1. MNIST (subsection 6.4.2).

(1) P_1: $\epsilon = 10$, $n_{iter} = 40$, $\epsilon_{iter} = 0.5$

(2) P_2: $\epsilon = 2$, $n_{iter} = 40$, $\epsilon_{iter} = 0.1$

*https://github.com/fra31/auto-attack

(3) P_∞: $\epsilon = 0.3$, $n_{iter} = 40$, $\epsilon_{iter} = 0.01$
(4) DF_∞: $\epsilon = 0.11$, $n_{iter} = 30$
(5) CW_2: *max_iterations* $= 20$, *learning_rate* $= 0.1$, *binary_search_steps* $= 5$
(6) $P^\bullet_\infty$: $\epsilon = 0.4$, $n_{iter} = 40$, $\epsilon_{iter} = 0.033$
(7) $DF^\bullet_\infty$: $\epsilon = 0.4$, $n_{iter} = 50$
(8) $CW^\bullet_2$: *max_iterations* $= 30$, *learning_rate* $= 0.12$, *binary_search_steps* $= 7$
(9) *AutoAttack*$_\infty$: $\epsilon = 0.3$, *norm* $=$ "Linf"

The MSD attack uses the same tuning as the individual P_p attacks.

C.1.2.2. CIFAR10 without hyperparameter optimisation (subsection C.2.2).

(1) P_1: $\epsilon = 10$, $n_{iter} = 40$, $\epsilon_{iter} = \frac{2}{255}$
(2) P_2: $\epsilon = 0.5$, $n_{iter} = 40$, $\epsilon_{iter} = \frac{2}{255}$
(3) P_∞: $\epsilon = \frac{8}{255}$, $n_{iter} = 40$, $\epsilon_{iter} = \frac{2}{255}$
(4) DF_∞: $\epsilon = 0.011$, $n_{iter} = 30$
(5) CW_2: *max_iterations* $= 20$, *learning_rate* $= 0.01$, *binary_search_steps* $= 5$
(6) $P^\bullet_\infty$: $\epsilon = \frac{12}{255}$, $n_{iter} = 70$, $\epsilon_{iter} = \frac{2}{255}$
(7) $DF^\bullet_\infty$: $\epsilon = \frac{8}{255}$, $n_{iter} = 50$
(8) $CW^\bullet_2$: *max_iterations* $= 30$, *learning_rate* $= 0.012$, *binary_search_steps* $= 7$
(9) *AutoAttack*$_\infty$: $\epsilon = \frac{8}{255}$, *norm* $=$ "Linf"

MSD uses the same tuning as the individual P_p attacks.

C.1.2.3. CIFAR10 with hyperparameter optimisation (subsection 6.4.4). We use the same tuning as above for testing, with the addition of an *AutoAttack*$_2$ adversary with $\epsilon = 0.5$ and *norm* ="L2". However, for training, based on (Rice et al., 2020), we set

(1) P_1: $\epsilon = 10$, $n_{iter} = 10$, $\epsilon_{iter} = \frac{20}{255}$
(2) P_2: $\epsilon = 0.5$, $n_{iter} = 10$, $\epsilon_{iter} = \frac{15}{255}$
(3) P_∞: $\epsilon = \frac{8}{255}$, $n_{iter} = 10$, $\epsilon_{iter} = \frac{2}{255}$

and do the same for MSD.

C.1.3. More on how the attack discriminator is trained

In order to train the attack discriminator of subsections 6.4.1 and C.2.1, we load the "IMAGENET1K_SWAG_LINEAR_V1" weights (Singh et al., 2022) of a ViT b16 available on Torchvision with

```
torchvision.models.vit_b_16(weights="IMAGENET1K_SWAG_LINEAR_V1")
```

We freeze all but the last (linear) layer of the ImageNet-pretrained ViT, which we reset and adapt to the proper number of output classes, and which is therefore the only layer that we train. We use a ViT because it has significantly higher performance than a ResNet18, which we tried initially and found to have much difficulty converging to good classifiers both on the validation and test sets, even after 150 epochs. In contrast, the ViT almost reaches its final values in 2 epochs.

We proceed by using the attacks with their tuning from subsubsection C.1.2.3 on CIFAR10, using the hyperparameter-optimised model adversarially trained against P_∞. Since the task is to show that a discriminator can classify different attacks, we do not need to perturb all CIFAR10 classes and limit ourselves to a single at a time. The confusion matrices provided in this work are based on perturbing all CIFAR10 samples of class "0" (airplanes). After saving all the images of airplanes perturbed with every attack, we load them as a dataset for attack classification with the pretrained ViT. Our code allows one generate or use this dataset, selecting the classes and attacks used.
As preprocessing, we transform the images into perturbations by subtracting the unperturbed original image from the adversarial images, i.e. $A(x^0) - x^0$ where x^0 is the original unperturbed image and A is an attack. Furthermore, we opt to standardise at an instance level (i.e. per image) instead of using batch or dataset statistics. A seed of 0 is used in every case.
We use Adam (Kingma and Ba, 2014) as an optimiser with its default Pytorch settings in torch1.13[†] (lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0).
We also report the epoch at which we early stopped training for the analyses:

(1) Fig. 8 (discriminating between all L_p attacks considered in this work): epoch 40.
(2) Fig. 20 (discriminating between P_∞ and $DF^\bullet_\infty$): epoch 16.
(3) Fig. 21 (discriminating between $P_1, P_2, P_\infty, DF_\infty, CW_2$): epoch 56.

As a reminder, the same base dataset is used for all 3 models, retaining only the samples corresponding to attacks selected for the analysis.
Disclaimer: we made no particular attempt to improve further the attack discriminator. It is probable that more engineering, e.g. transfer learning or preprocessing heuristics, might lead to even better results. However, we believe our current models achieve the intended goal of showing that all these attacks induce different distributions even for the same p-norm and $epsilon$, as evidenced by the fact that for any attack, we can show on the test set that $p(A_{true}) > p(A \neq A_{true})$ where A_{true} is the true attack used to generate the sample and $p(A)$ is the probability predicted by the model that a perturbation corresponds to attack A. See 6.4.1 and C.2.1.

C.1.4. More on how the models are trained

Note that when activating REx, we always reset the optimiser to avoid using accumulated momentum from the baseline. When tuning hyperparameters, we find that activating the REx penalty after learning rate decays generally is a worse strategy than activating it before when the baseline's accuracy decreases after a decay. All models are trained using a single NVIDIA A100 for MNIST and 2 NVIDIA A100s for CIFAR10.

C.1.4.1. No hyperparameter optimisation. First, we pretrain the architecture on the clean dataset. Then, the baseline is trained on the appropriate seen domains. On MNIST, convergence does not

[†]Every other experiment uses torch1.8.1, the ViT analysis being a late addition to this work and ViT requiring more recent Pytorch versions.

happen in many baselines' case until thousands of epochs. Therefore, we choose to stop training when progress on the seen domains slows, as in Fig. 9 where we stopped training for example at epoch 1125 for both the Avg and the Avg+REx models. For CIFAR10, as Fig. 22 indicate, the accuracies peak in significantly less epochs, so we early stop when the Ensemble (seen) accuracy on the seen domains peaks. Note that we do this manually by looking at the seen domains' validation curves (see Sec. C.3 for more details on early stopping). REx is triggered on a baseline before the baseline's early stopping epoch, when progress on the seen domains slows.

An important precision about Fig. 9, 22 is that unseen attack performance is only evaluated every 5 epochs, hence the jagged aspect of the curves. We do this because of the huge computational cost of running all 9 attacks on each sample every epoch.

For a full description of early stopping and when we activated the REx penalty on baselines:

MNIST

- P_1 model: early stopped at epoch 95
- P_2 model: early stopped at epoch 75
- P_∞ model: early stopped at epoch 1125
- Avg model: early stopped at epoch 1125
- Avg+REx model: REx penalty activated at epoch 726, early stopped at epoch 1125
- Avg_{PGDs} model: early stopped at epoch 1105
- Avg+REx_{PGDs} model: REx penalty activated at epoch 551, early stopped at epoch 1105
- MSD model: early stopped at epoch 655
- MSD+REx model: REx penalty activated at epoch 101, early stopped at epoch 655

CIFAR10

- P_1 model: early stopped at epoch 69
- P_2 model: early stopped at epoch 59
- P_∞ model: early stopped at epoch 45
- Avg model: early stopped at epoch 50
- Avg+REx model: REx penalty activated at epoch 301, early stopped at epoch 330
- Avg_{PGDs} model: early stopped at epoch 95
- Avg+REx_{PGDs} model: REx penalty activated at epoch 301, early stopped at epoch 370
- MSD model: early stopped at epoch 40
- MSD+REx model: REx penalty activated at epoch 26, early stopped at epoch 70

C.1.4.2. With hyperparameter optimisation. The experiments are run with a ResNet18 architecture on CIFAR10. We follow the results of Rice et al. (2020) and Pang et al. (2020), using a piecewise learning rate schedule decay and a weight decay value of $5 \cdot 10^{-4}$. In all cases we start with a learning rate of 0.1, decayed to 0.01 at the corresponding milestone. In preliminary tuning experiments, we observe the Avg_{PGDs} to perform very poorly relative to other baselines, and chose to drop that baseline.

- P_∞ model: milestone at epoch 100, early stopped at epoch 103
- Avg model: milestone at epoch 100, early stopped at epoch 140
- Avg+REx model: REx penalty activated at epoch 50 (before lr decay), milestone at epoch 100, early stopped at epoch 110
- MSD model: milestone at epoch 50, early stopped at epoch 51
- MSD+REx model: REx penalty activated at epoch 50 (before lr decay), milestone at epoch 97, early stopped at epoch 99

We attempt several learning rate schedule milestones for both MSD and MSD+REx, which has a higher impact than for other models. This process can be automated since the worst-case seen accuracy always peaks within a few epochs of a milestone.

C.1.5. Other implementation details

We use the implementation of `https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py` for ResNet18. For StAdv and RecolorAdv, we use the code of Laidlaw and Feizi (2019), available at `https://github.com/cassidylaidlaw/ReColorAdv`.

REx's β parameter is generally set to 10, except for MSD+REx on MNIST in subsection 6.4.2 where it is set to 4. These numbers initially come from setting β to a value of the same order of magnitude as $\frac{L_{Avg}}{\text{Var}}$'s value at the epoch REx is activated, in the early iterations of our experiments. This is done to encourage the optimisation dynamics to neglect neither term of the REx loss. We found that $\beta = 10$ worked generally well, even in many settings where empirically $\frac{L_{Avg}}{\text{Var}} \simeq 30$. See Sec. C.4 for a discussion of how the choice of β affects performance.

C.2. Additional results

C.2.1. More about the attack discriminator

We invite the reader to consult Apdx C.1.3 for details about the methodology of the results presented here and in subsection 6.4.1.

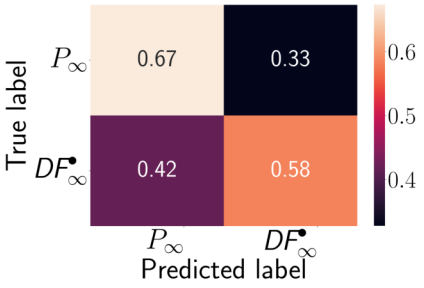


Figure 20. Confusion matrix of ViT predicting whether perturbations stem from P_∞ or $DF^\bullet_\infty$ (normalised by row).

Here, we give more results about the attack discriminator. To claim unequivocally that all attacks considered are distributionally shifted with respect to one another, we require $p(A_{true}) > p(A \neq A_{true})$ where A_{true} is the true attack used to generate the sample and $p(A)$ is the probability predicted by the model that a perturbation corresponds to attack A. [‡] This is true for all attacks as seen in Fig. 8, except $DF^\bullet_\infty$ being confused for P_∞, and CW_2 being confused for $CW^\bullet_2$ or AutoAttack$_\infty$.

$DF^\bullet_\infty$ and P_∞ share the same $\epsilon = 8/255$ and the model on which the attacks were generated (the P_∞ adversarially trained model from Table 3) performs roughly equally on both (46.3% accuracy on $DF^\bullet_\infty$ vs 47.3% on P_∞). Therefore, it would be particularly important for our claim to be able to measure some difference between those distributions, even if there may be some overlap. Note that despite sharing a similar performance and ϵ, AutoAttack$_\infty$ does not induce the same confusion.

In order to tackle this, we train a binary ViT classifier to tell apart $DF^\bullet_\infty$ and P_∞. We can see in Fig. 20 that the model is now able to properly predict each class with higher probability than the other, with average probability 63%. The confusion (off-diagonal entries) are indicative that as we might suspect, there is *some* overlap between the distributions of perturbations generated by the two attacks.

Concerning the CW_2 confusion results, this is an artefact of the implementation of CW_2 that we use. Indeed, the implementation of the attack from advertorch early stops by default if the iterations get stuck in a local minimum. This leads to some samples having near 0 perturbation and not being adversarial. As argued in the main text, this is no longer an issue as soon as CW_2 is tuned so that this does not happen, e.g. with the $CW^\bullet_2$ tuning.

C.2.2. CIFAR10 - no hyperparameter optimisation

The results on CIFAR10 without performing hyperparameter optimisation on each model are summarised in Table 15. We observe again that REx is an improvement over the ensemble of seen attacks compared to the baselines it was used on. As on MNIST, this happens by improving the performance on the strongest of the seen attacks and sacrificing a little performance on the top

[‡] While this is out of the scope of this work, it is possible to relate the divergence between the distributions to the performance of the discriminator, as done by Bashivan et al. (2021) using Kifer et al. (2004)'s $\mathcal{H}\Delta\mathcal{H}$ divergence theory.

Table 15. Accuracy on CIFAR10 for different domains. Ensembles omit $CW_2^\bullet$ due to overtuning.

	Defenses									
	None	Adversarial training			Avg	Avg+REx	Avg_{PGDs}	Avg+REx_{PGDs}	MSD	MSD+REx
No attack	87.6	92.1	87.1	77.4	80.9	75.1	82.8	79.0	76.3	75.1
P_1	80.3	87.6	85.0	75.7	78.7	72.0	80.4	77.0	74.4	72.7
P_2	19.9	47.8	70.9	66.6	69.8	65.3	71.1	68.0	65.7	65.9
P_∞	0.0	0.0	9.7	39.5	34.4	**44.2**	32.3	**41.2**	37.9	**41.9**
DF_∞	4.1	19.2	60.2	64.6	64.9	62.2	66.8	64.5	62.7	63.6
CW_2	0.0	0.0	1.3	11.2	9.8	**21.6**	8.7	**16.5**	10.7	**17.5**
$P_\infty^\bullet$	0.0	0.0	1.0	20.1	16.3	**24.1**	13.2	**22.1**	19.3	**23.8**
$DF_\infty^\bullet$	0.0	0.0	9.5	38.5	35.6	**40.5**	33.0	**39.1**	36.6	**40.4**
AutoAttack_∞	0.0	0.0	8.1	37.2	33.7	**38.8**	31.2	**37.6**	36.0	**39.0**
Ensemble (seen)	-	-	-	-	9.8	**21.2**	32.3	**41.2**	37.9	**41.8**
Ensemble (unseen by all models)	0.0	0.0	1.0	20.1	16.3	**24.0**	13.2	**22.0**	19.3	**23.6**
Ensemble (unseen by this model)	0.0	0.0	0.9	10.7	16.3	**24.0**	7.7	**14.2**	10.3	**16.1**
Ensemble (all)	0.0	0.0	0.9	10.7	9.4	**17.9**	7.7	**14.2**	10.3	**16.1**
$CW_2^\bullet$	0.0	0.0	0.1	1.1	1.6	**2.6**	1.1	**2.7**	1.0	**2.7**

performing attack(s). Moreover, REx consistently yields a significant improvement in robustness when evaluated against the ensemble of unseen attacks, too. The only individual domains where REx never yields significant improvements are the clean (unperturbed) data, P_1 and P_2, whether

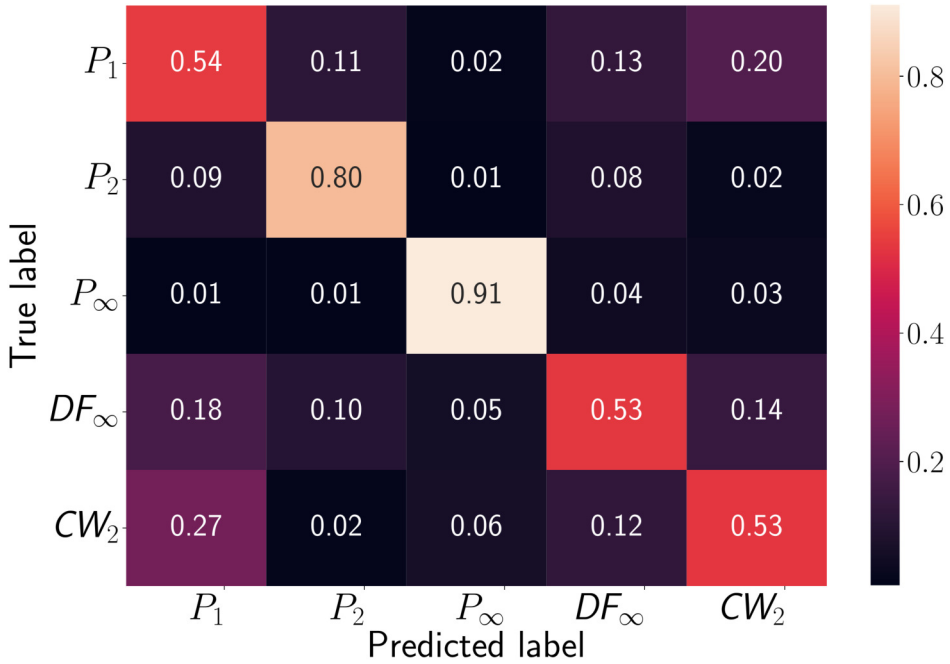
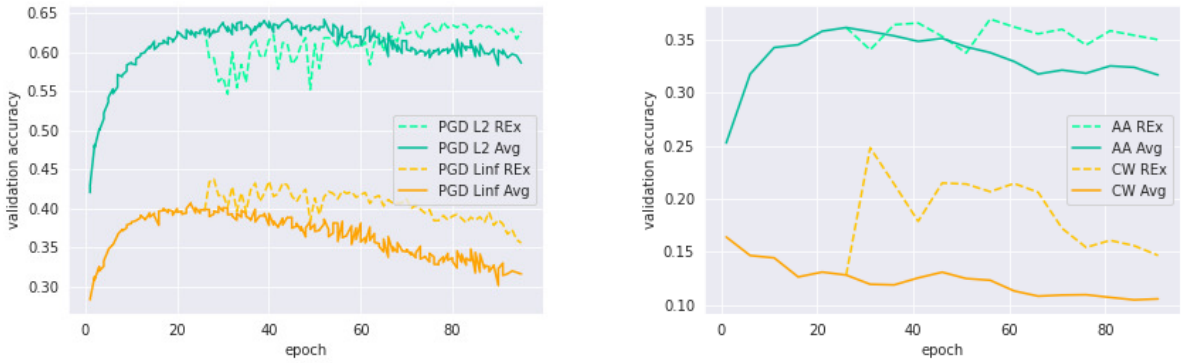


Figure 21. Confusion matrix of ViT predicting whether perturbations stem from 5 different types of attacks (normalised by row). Test accuracy is 66.3%.

they were seen during training or not. Given the relatively good performance of the baselines and REx on those domains, this is in line with REx's tendency to sacrifice a little accuracy on the best performing domains to improve significantly the performance on the worst performing domains.
While adversarial training on either P_1 or P_2 fails to yield robustness to unseen attacks, we observe that these two defenses are the only ones for which the clean accuracy does not decrease significantly.
We note that unlike on MNIST, MSD is significantly more competitive with the other baselines, and its performance is relatively similar to the one reported by Maini et al. (2020) (likely due to using the same architecture on CIFAR10).
As with tuned models (subsec. 6.4.4), the model trained on P_∞ performs better than the Avg, Avg_{PGDs} and MSD models on the set of attacks unseen by all models. Conversely, training on ensembles of attacks also hurts performance on P_∞, unless we apply REx. In other words, only REx is able to improve both P_∞ *and* worst-case performance over an ensemble of attacks. Moreover, without hyperparameter tuning, REx appears to lose more performance on the best performing domains. This is particularly notable in the case of P_2 attacks, where for example REx improves the P_2 accuracy by $+0.8\%$ with hyperparameter tuning, vs -4.5% without with the Avg model training on $\{P_\infty, DF_\infty, CW_2\}$. In contrast, the gap in performance in performance between MSD and MSD+REx is even larger than without individual hyperparameter optimisation, showing that REx helps bridge gaps in performance due to suboptimal tuning.



(a) CIFAR10 seen attacks

(b) CIFAR10 unseen attacks

Figure 22. Validation accuracy of MSD on CIFAR10 (bottom) with and without REx (dashed line), against seen attacks (left) and unseen attacks (right) (AA=AutoAttack).

As with MNIST and the CIFAR10 hyperparameter-optimised models, early stopping is important, and moreso than with MNIST, the performance of REx performance quickly peaks, as seen in Fig. 22.

Key observations 4 (no hyperparameter tuning):

- REx improves the performance of all baselines on CIFAR10 with a ResNet18, from 10.7% with the best baseline to 17.9% accuracy against an ensemble of attacks, by sacrificing a little robustness against the weakest individual attacks.
- Multi-perturbation defenses only achieve higher P_∞ and worst-case performance than P_∞ adversarial training when using REx.
- REx helps bridge gaps in performance due to suboptimal hyperparameter tunings.

C.2.3. Perceptual Adversarial Training and additional experiments

Perceptual Adversarial Training: due to the similarity of our motivation with that of Perceptual Adversarial Training (PAT) (Laidlaw et al., 2021) (i.e. be robust against imperceptible attacks), we evaluate their model on our benchmark. We simply run the code made publicly available by the authors in order to train with PAT a model, and load it in our evaluation pipeline. With a ResNet18 on CIFAR10, we find that they achieve considerably worse robustness, with Ensemble (all) worst-case robustness of 3.8%. The model is noticeably weak against L_∞ attacks, achieving 6.9% accuracy on AutoAttack and 10.0% on P_∞. While this may be surprising, this can be explained by noting that Laidlaw et al. (2021) use a ResNet50 on CIFAR10, while we used their code with the "–arch resnet18" argument to train a model immediately comparable to the main results of our work.
Robustness against different tunings of AutoAttack: we consider an additional tuning of AutoAttack L_∞ on CIFAR10 with the hyperparameter-optimised ResNet18, this time with $\epsilon = 12/255$. We find that the MSD model achieves 25.0% accuracy while MSD+REx achieves 26.8%. The Avg model from Table 3 achieves 15.6% accuracy while its REx counterpart reaches 24.2% in 30 less epochs of training (see Sec. C.1.4.2 for number of epochs for each model). The P_∞ model yields 25.9%, which is higher than the other baselines, but still worse than MSD+REx. This further supports the claim that REx improves robustness against unseen attacks and tunings of attacks.

C.2.4. Additional results on CIFAR10-C

In order to test the investigate the robustness of the different tuned baselines against non-adversarial perturbations, we evaluate them on the corruptions of CIFAR10-C (Hendrycks and Dietterich, 2019). Table 16 shows how P_∞ is the overall best performing model. The Avg and MSD baselines perform surprisingly poorly on CIFAR10-C, and REx leads to very sigificant improvements on both baselines that bring the performance closer to that of the model trained on P_∞. In particular, MSD+REx comes very close (within 2% accuracy) to the P_∞ baseline's performance, only outperforming P_∞ marginally on defocus blurs, elastic transforms, gaussian blurs, motion blurs, and zoom blurs, and having equal performance on fog. This is in contrast with the intuition that because REx had lower (in-distribution) clean accuracy than most other baselines on CIFAR10, this would also be true for out-of-distribution non-adversarial data.

Table 16. Accuracies of tuned CIFAR10 models on CIFAR10-C corruptions.

	Defenses					
	None	P_∞	Avg	Avg+REx	MSD	MSD+REx
brightness	31.7	61.5	29.1	54.5	51.7	60.2
contrast	35.4	43.7	33.4	40.6	39.8	42.7
defocus blur	22.9	52.6	25.8	49.1	44.9	52.8
elastic transform	21.6	50.1	24.5	46.4	42.1	50.3
fog	32.5	45.8	32.7	44.2	41.7	45.8
frost	28.4	63.1	33.6	56.8	54.5	61.0
gaussian blur	22.2	51.4	25.7	48.1	44.5	52.0
gaussian noise	13.4	53.3	25.1	48.7	35.4	50.9
glass blur	18.0	49.7	24.2	46.1	39.8	48.3
impulse noise	14.7	47.7	24.8	45.6	33.0	46.0
jpeg compression	18.6	54.7	25.4	50.5	44.8	53.7
motion blur	23.2	49.6	24.9	46.3	42.9	50.4
pixelate	20.9	54.8	25.3	50.2	44.6	53.2
saturate	20.8	49.1	20.8	43.6	37.6	47.6
shot noise	13.6	52.3	24.8	48.4	34.5	50.3
snow	22.3	57.9	27.6	52.4	47.9	55.8
spatter	18.4	52.6	24.6	48.4	42.4	50.5
speckle noise	13.2	50.9	24.5	47.5	32.3	49.0
zoom blur	22.3	52.3	26.1	48.1	45.2	52.7
average	21.8	52.3	26.5	48.2	42.1	51.2

C.2.5. Transferability of REx models

Several works have highlighted how adversarial robustness can be of interest for transfer learning (Salman et al., 2020; Utrera et al., 2021). We freeze the parameters of the hyperparameter-optimised CIFAR10 models, only resetting the last linear layer (and adapting it to the appropriate number of output classes) and allowing it to train on the CIFAR100 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011) training datasets, then evaluating them on their test sets. We train for 30 epochs with a learning rate of 0.1, decaying by factors 0.1 at epochs 10 and 20. We repeat this 3 times per model, using a different seed ("0", "1" and "2") to report average accuracies on the test set and standard deviations.

Disclaimer: we did not attempt any particular optimisation of these results with state of the art techniques, and merely provide these results for completeness. We also highlight that work on transfer learning with robust models often focuses on larger models (e.g. ResNet50 for Utrera et al. (2021)), which can significantly affect the results. Finally, we realise that transferring from CIFAR10 to CIFAR100 (instead of the other way around) may be overly ambitious, with works such as (Utrera et al., 2021) focusing instead on CIFAR100 to CIFAR10.

Table 17. Accuracies of tuned CIFAR10 models finetuned on CIFAR100, averaged over 3 finetunings with different seeds per defense.

	Defenses					
	None	P_∞	Avg	Avg+REx	MSD	MSD+REx
Accuracy	28.9±0.1	29.3±0.1	25.9±0.2	27.7±0.2	28.4±0.1	27.7±0.1

On CIFAR100, we can see on Table 17 that the P_∞ model transfers best, marginally better than the non-robust model. All other baseline perform worse than the non-robust model, and REx only appear to improve the Avg baseline, while decreasing by 0.7% the accuracy when used on the MSD baseline before the finetuning. The performance of the non-robust model is perhaps not surprising, given the similarity between the CIFAR10 and CIFAR100 classes, which may explain the relative performances of the baselines being correlated with those observed on CIFAR10 on unperturbed data.

Table 18. Accuracies of tuned CIFAR10 models finetuned on SVHN, averaged over 3 finetunings with different seeds per defense.

	Defenses					
	None	P_∞	Avg	Avg+REx	MSD	MSD+REx
Accuracy	42.4±0.2	49.2±0.3	44.8±0.1	47.1±0.1	49.1±0.2	48.0±0.1

In order to investigate this hypothesis, we also attempt a similar finetuning on the SVHN dataset. The (cropped) SVHN dataset consists in pictures of house numbers, where the task is to read the digit at the center of the image. Therefore, this task corresponds to a shift substantially different than CIFAR100. As suspected, we observe in Table 18 than all of the robust models now have significantly better performance than the non-robust model. The P_∞ and MSD models both transfer the best, with the MSD+REx model losing again roughly 1% of performance compared to MSD, but still ahead of the non-robust model or the Avg and Avg+REx models.

C.3. When to early stop, and how learning rate milestones affect performance

This section shows results about learning rate schedule milestones and early stopping for the CIFAR10 models with hyperparameter optimisation.

C.3.1. MSD model

In this subsection, we illustrate two points using Fig. 23: how early stopping is performed on the MSD and MSD+REx models, and how the choice of learning rate decay milestone affects performance. Regarding the former, we early stop based on the peak of the validation Ensemble (seen) accuracy. We motivated that worst-case performance is a more appropriate notion of robustness, and unseen attacks should not be used to make model-selection decisions as they are used to simulate "future", novel attacks that were not known when designing the defenses. Concerning early stopping, we note that the peak performance on Ensemble (seen) accuracy is reached:

- At epoch 101 for the MSD model with milestone at epoch 100
- At epoch 51 for the MSD model with milestone at epoch 50
- At epoch 105 for the MSD+REx model with milestone at epoch 100
- At epoch 99 for the MSD+REx model with milestone at epoch 97
- At epoch 54 or 56 for the MSD+REx model with milestone at epoch 50.

Regarding how milestone choice affects performance, this illustrates how we searched for hyperparameters. We attempt learning rate decays milestones at various epochs, which we then evaluate only for a few epochs, as the performance decays fast anyway as predicted by Rice et al. (2020). The curves in Fig. 23 represent the best learning rate scheduler milestones found for MSD and MSD+REx. We retain the model with best validation Ensemble (seen) accuracy for our final results presented in Sec. 6.4. As performance of the best checkpoints of MSD with milestones 50 and 100, and respectively MSD+REx with milestones 97 and 100, are very close, in both cases we evaluated the final models on the test set and kept the best in each case (MSD with milestone 50 and MSD+REx with milestone 97), observing only minor differences between each defense's pair of choices.

C.3.2. Avg model

As argued in our introduction, it is somewhat surprising that REx successfully improved MSD due to MSD being a single-domain baseline. REx was originally designed to be used with baselines where multiple domains appear in the loss, and in particular ERM over multiple domains. Fig. 24 illustrates how REx clearly benefits the Avg baseline more, with very little tuning effort required to achieve results above all baselines as reported in subsection 6.4.4.

C.4. Effect of varying REx's β coefficient

In Fig. 25 we investigate the effect of varying β in REx+MSD. Note that the hyperparameters of both MSD and MSD+REx are suboptimally tuned in this figure, which only aims to illustrate the effect of varying β. To generate those figures, both baselines use weight decay, MSD's learning rate milestone is set at epoch 100. MSD+REx loads an MSD checkpoint at epoch 105 with a learning rate set to 0.1 which is decayed at epoch 155.

We observe that while there is some robustness to the choice of β for some domains, the difference is especially large on CW_2 and $P_\infty^\bullet$, where a larger value benefits the model. β also has a fairly large impact on the clean, P_1 and P_2 accuracies. This is explained by the fact that low values of β imply that the variance term will have lower impact and the model will value high performing seen domains (clean, P_1, P_2) more when updating weights than if larger values of β were used. In contrast, large values of β emphasise the variance regularisation which benefits accuracy against stronger attacks more.

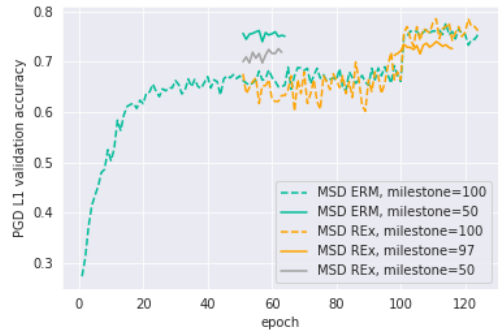
C.5. Miscellaneous results

- In preliminary experiments without hyperparameter tuning, REx did not benefit a model pretrained solely on P_∞.
- In preliminary experiments without hyperparameter tuning, REx (and the Avg baselines) incurred significant losses in robustness when attempting to train on only one type of perturbation per sample in the batch (in the sense that each sample from a minibatch would only contribute to a single random domain among the seen domains, instead of all seen domains for each sample).
- We attempted to just add the variance penalty term to the MSD baseline while still training on the MSD attack. More explicitly, the ERM term consisted in the loss on the MSD attack, and the variance term separately computed P_1, P_2, P_∞ perturbations. This leads to iterations that are twice as expensive, for no observed benefit. Therefore, we instead prefered to define MSD+REx as performing REx+Avg$_{PGDs}$ on a model pretrained with MSD.
- On CIFAR10, training on $\{P_\infty, DF_\infty, CW_2\}$ is about 8 times more computationally expensive than training on *PGDs* or MSD with 10 iterations per P_p attack (factoring in that in all cases, we validate on all domains every 5 epochs). Since the former leads to a significant advantage in robustness over the ensembles of attacks evaluated here, there is a strong trade-off between computational cost and adversarial robustness when training on those attacks.

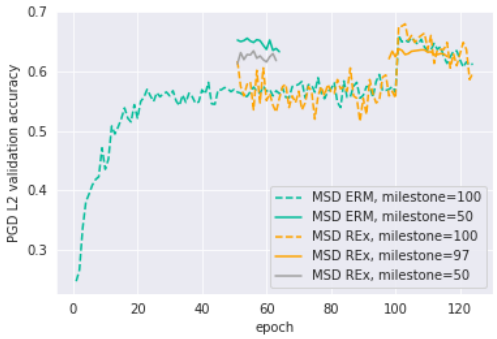
In figures 26 and 27, we show the domains generated for the Avg and Avg+REx models from different attacks, along with the unperturbed data. Above each image is the class predicted by the model, and in parentheses the true class. The classes match the following numbers:

- airplane : 0
- automobile : 1
- bird : 2
- cat : 3
- deer : 4
- dog : 5
- frog : 6
- horse : 7
- ship : 8
- truck : 9

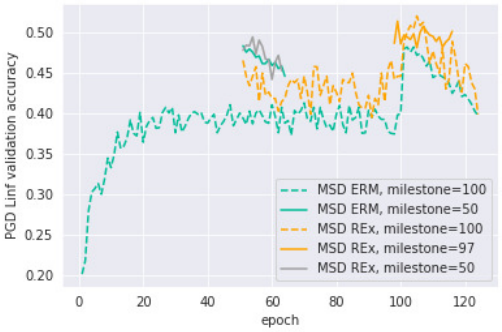
This highlights how in general, these adversarial examples are not difficult to classify for humans. **This however also illustrates how *CW*$_2$ perturbations, in spite of being unbounded, tend to be much less perceptible than those stemming from most commonly used bounded attacks, such as P_∞ or *AA*$_\infty$, at tunings where they have the highest attack success rate by far among attacks considered.**



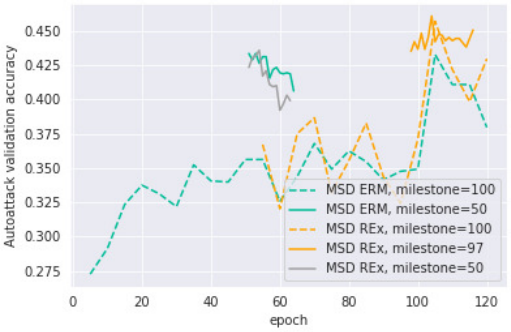
(a) P_1 accuracy



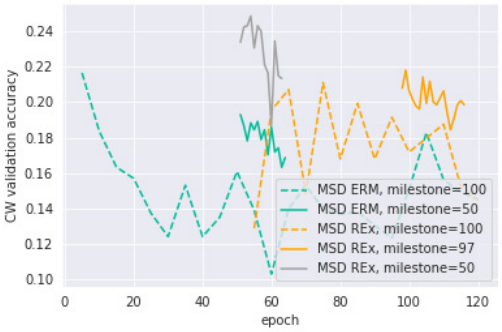
(b) P_2 accuracy



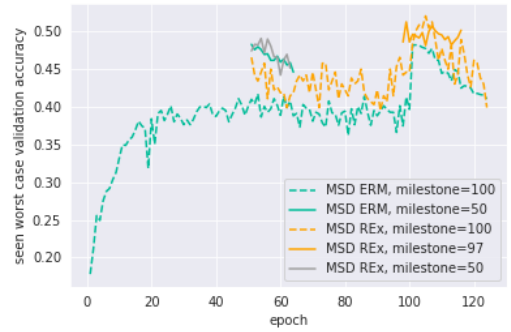
(c) P_∞ accuracy



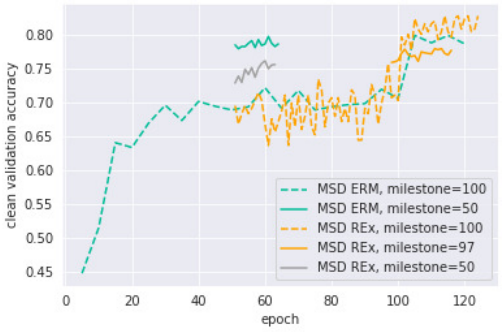
(d) AutoAttack$_\infty$ accuracy



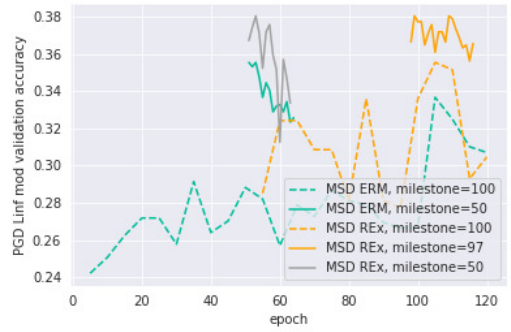
(e) CW$_2$ accuracy



(f) Ensemble (seen) accuracy

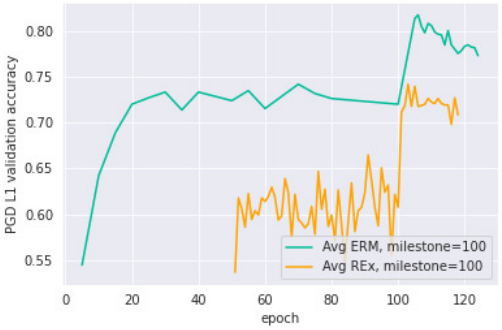


(g) No attack (clean) accuracy

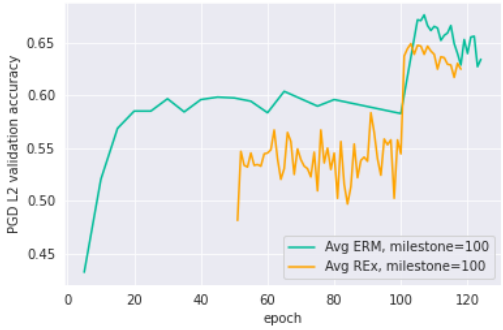


(h) $P_\infty^\bullet$ accuracy

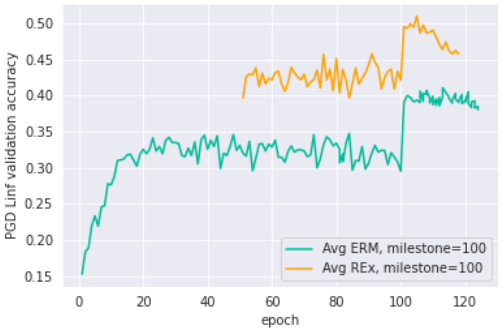
Figure 23. Validation accuracy of MSD and MSD+REx on CIFAR10 on various attacks with different milestones for the learning rate decay. Early stopping is performed for each model at the peak of the ensemble (seen) accuracy. The MSD model with a milestone at epoch 50 and the MSD+REx model with a milestone at epoch 97 are the final models retained in subsection 6.4.4.



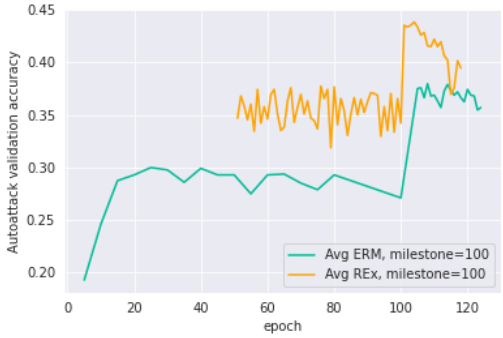
(a) P_1 accuracy



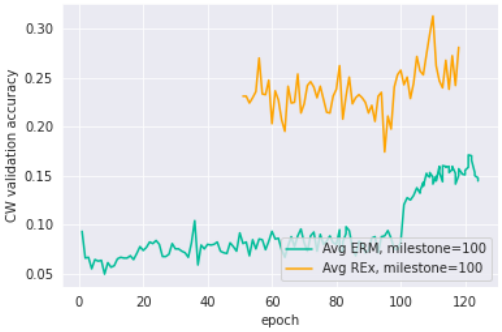
(b) P_2 accuracy



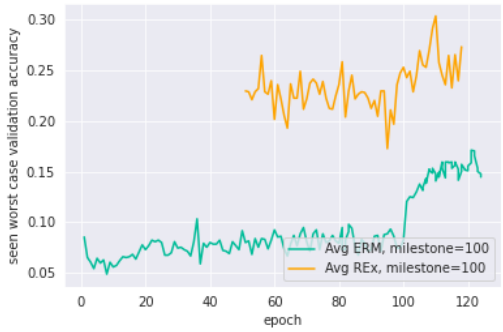
(c) P_∞ accuracy



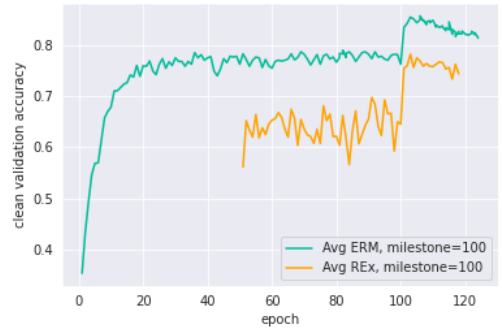
(d) AutoAttack$_\infty$ accuracy



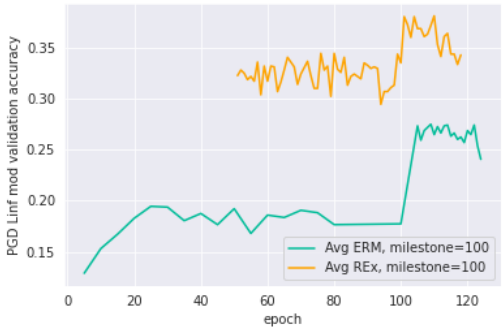
(e) CW$_2$ accuracy



(f) Ensemble (seen) accuracy



(g) No attack (clean) accuracy



(h) $P_\infty^\bullet$ accuracy

Figure 24. Validation accuracy of Avg and Avg+REx on CIFAR10 on various attacks with learning rate decay milestone at epoch 100. This illustrates how much easier it is to get improvements with REx on baselines based on ERM over multiple domains.

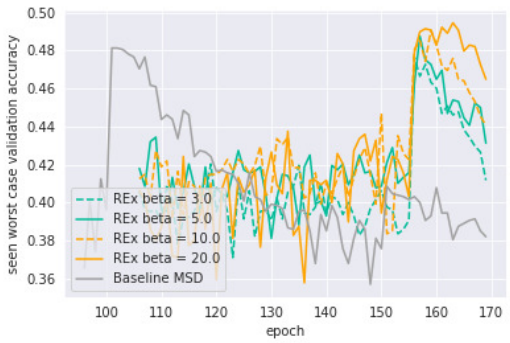
(a) P_1 accuracy

(b) P_2 accuracy

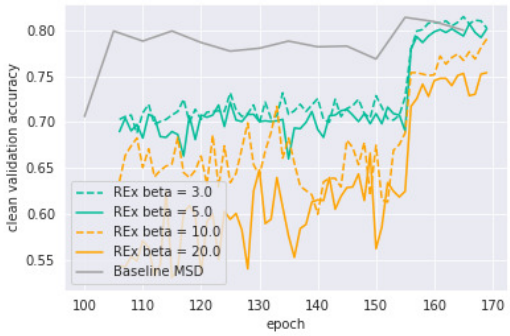
(c) P_∞ accuracy

(d) AutoAttack$_\infty$ accuracy

(e) CW$_2$ accuracy



(f) Ensemble (seen) accuracy

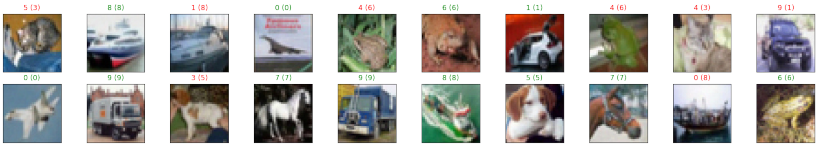


(g) No attack (clean) accuracy

(h) $P_\infty^\bullet$ accuracy

Figure 25. Validation accuracy of MSD and MSD+REx on CIFAR10 on various attacks with different values of β. Note that MSD has a learning rate decay at epoch 100, and MSD+REx at epoch 155.

(a) P_1 adversarial examples



(b) P_2 adversarial examples

(c) P_∞ adversarial examples

(d) AutoAttack$_\infty$ adversarial examples

(e) CW$_2$ adversarial examples

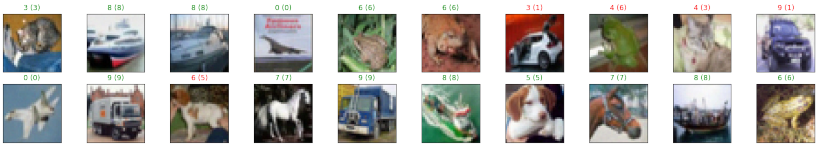
(f) CW$_2^\bullet$ examples

(g) No attack (clean) adversarial examples

(h) $P_\infty^\bullet$ adversarial examples

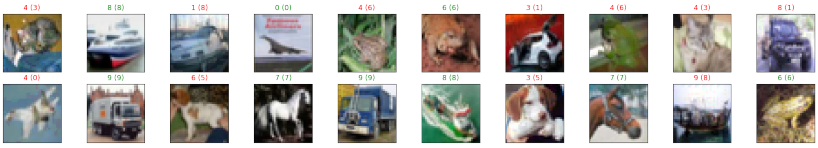
Figure 26. Adversarial examples generated from the hyperparameter-optimised Avg model, for each attack. The predicted class and in parentheses, the true class, are displayed above each image.

(a) P_1 adversarial examples



(b) P_2 adversarial examples

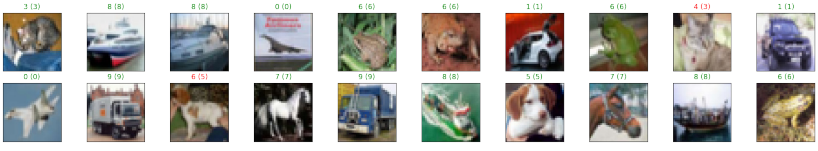
(c) P_∞ adversarial examples



(d) AutoAttack$_\infty$ adversarial examples

(e) CW$_2$ adversarial examples

(f) CW$_2^\bullet$ adversarial examples



(g) No attack (clean) adversarial examples

(h) $P_\infty^\bullet$ adversarial examples

Figure 27. Adversarial examples generated from the hyperparameter-optimised Avg+REx model, for each attack. The predicted class and in parentheses, the true class, are displayed above each image.

Chapter D

Supplementary Material For the Third Article

D.1. Extended results

In the following subsections, we first present some new results in the *Domain incremental continual pre-training* setting, comparing replay for different dataset sizes, and providing a qualitative analysis of our German language models. We also provide aggregated evaluation and final loss tables for all models in the paper.

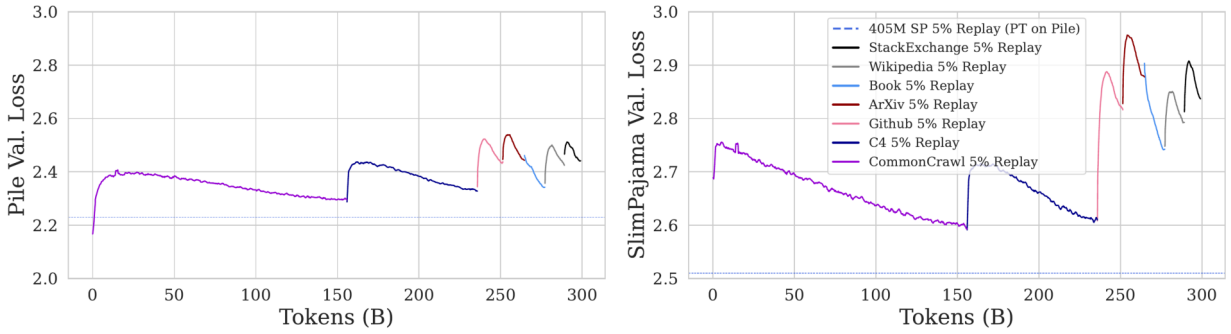
D.1.1. Domain Incremental continual pre-training

We consider a **domain incremental learning** setting, where we train on a sequence of N future datasets $\{\mathcal{D}_0,\mathcal{D}_1,\ldots,\mathcal{D}_{N-1}\}$ each of which comes from a distinct domain. This setting is particularly challenging due to the distribution shift experience at the transition between each domain. Concretely, we consider dataset $\mathcal{D}_0$ to be pre-training on the Pile (Gao et al., 2020) and $\mathcal{D}_1$ to be pre-training on SlimPajama. However, instead of consuming the data in $\mathcal{D}_1$ using the sampling percentages from table 6, we process the dataset one domain at a time starting from the largest domain to the smallest. This simulates a situation where new data is received from different domains at different times and we wish to update our models on the sequence while being robust to all distribution shifts.

Adapting replay to Domain Incremental continual pre-training. In settings that span more than two tasks, we use a form of reservoir sampling (Buzzega et al., 2020), where the replay buffer is updated at discrete intervals. We refer to this technique as *discrete reservoir sampling*. Specifically, given a sequence of N datasets $\mathcal{D}_0,\mathcal{D}_1,\ldots,\mathcal{D}_{N-1}$ of sizes $s_0,s_1,\ldots,s_{N-1}$ that are trained on in sequence, we update the replay buffer $\mathcal{R}$ at each dataset transition. Let $\mathcal{R}_i$ correspond to the state of the replay buffer as the start of training on the i-th dataset. For replay ratio $0 \leq \alpha \leq 1$, at any given $i > 0$, $\mathcal{R}_i$ will contain data from all $\mathcal{D}_j$ for $j < i$ in proportions

$$p_{i,j} := \frac{s_j \cdot (1-\alpha)^{\gamma_j} + \sum_{k=j+1}^{i-1} p_{k,j} \cdot s_k \cdot \alpha}{\sum_{k=0}^{i-1} s_k} \quad \text{with} \quad \forall j,\, p_{j+1,j} = 1, \tag{145}$$

where i is the index of the dataset on which we are currently pretraining, and $\gamma_j = 0$ if $j = 0$, and 1 otherwise. The latter is because when pretraining on $\mathcal{D}_i$, we only see $s_i \cdot (1 - \alpha)$ tokens of $\mathcal{D}_i$ because we use compute-equivalent replay, except for the first dataset $\mathcal{D}_0$ where replay is not used, and where we hence see all s_0 tokens of $\mathcal{D}_0$.



(a) Pile Validation Loss

(b) SlimPajama Validation Loss

Figure 28. Ingesting data sequentially by domain. We explore an alternative approach to consuming the available data. Similar to task-incremental or class-incremental learning, we train on one domain at a time in a sequential fashion. We use LR re-warming and LR re-decaying for each domain as well as *discrete reservoir sampling*.

Figure 28 reports results for a domain incremental approach to consuming our 300B token split of SlimPajama, where each domain is processed in decreasing order of size. We use learning rate re-warming, LR re-decaying, and replay with discrete reservoir sampling. The total data processed is the same as the model trained with LR re-warming and re-decaying on the 300B tokens of SlimPajama (blue dotted line). However, the order in which the data is processed and the learning rate schedule differs from the baseline.

We hypothesize that difficulties arise due to relatively strong distribution shifts and the many learning rate re-warmings. The number of consecutive distribution shifts with varying amounts of data from different domains makes it difficult to obtain strong final performance across all of them. In particular, it seems especially difficult to efficiently adapt to the new data distribution. We hypothesize part of these difficulties are optimization-related. Less frequent warming up and decaying of the learning rate may be beneficial in this setting, and the infinite learning rate schedules might help, but future research is needed to answer this question. At any rate, we believe this setting is unnecessarily difficult compared to consuming the data using a mixture (as we did in the main paper) and is therefore of limited practical use.

D.1.2. Model Merging v.s. Continual Pre-training

Some may be curious about the performance of model merging as a continual pre-training approach, and how it compares to continuing to pre-train with the methods used in this paper. In preliminary experiments with TIES (Yadav et al., 2023b) using Mergekit v0.0.4 (Goddard et al., 2024), we consider merging models trained separately on each dataset (e.g., merging 300B SP into 300B Pile

and 200B German into 300B Pile) using 300B Pile as the base model. We also merge continually pre-trained checkpoints into 300B Pile. English and German LM evaluation results can be found in Tables 19 and 20.

We find that, in general, continual pre-training outperforms merging models trained separately on each dataset and requires the same amount of compute. Moreover, we find that merging checkpoints of a continually pre-trained model does not increase performance, likely due to replay being a more efficient strategy. However, we note that these experiments are by no means comprehensive; other techniques may yield better performance.

Table 19. English LM Eval Performance of Models Merged with TIES v.s. Continual Pre-training. Normalized accuracy is reported for HellaSwag and exact match (EM) is reported for NaturalQuestions and TriviaQA. All other tasks report unnormalized accuracy. MMLU and TriviaQA are evaluated 5-shot, while all other tasks are zero-shot. In (x% R), the R should be read as Replay.

Model	Method	Base Model	Task Model(s)	TopK%	Hella-Swag	ARC-c	ARC-e	BoolQ	MathQA	MMLU	OpenBookQA	PIQA	WinoGrande	TruthfulQA MC1	TruthfulQA MC2	NaturalQuestions (EM)	TriviaQA (EM)	AVG
405M	TIES	300B Pile	200B Ger.	25	25.91	20.31	24.87	40.83	18.46	26.06	15.20	54.03	50.28	24.60	50.48	0.00	0.01	27.00
				75	27.64	18.00	30.43	42.57	21.21	25.45	15.40	55.11	51.85	24.72	45.73	0.00	0.30	27.57
				50	25.97	19.62	27.36	37.98	19.20	26.41	15.60	53.75	51.14	21.91	47.01	0.00	0.00	26.61
			300B Pile → 200B Ger. (25% R)	25	30.62	19.11	35.61	62.11	21.57	25.37	14.80	59.19	50.75	24.60	44.81	0.08	0.51	29.93
				50	34.52	20.90	42.63	61.77	23.48	25.02	17.00	61.37	50.59	26.56	45.78	0.25	2.65	31.73
				75	35.78	20.99	45.62	61.99	23.02	25.25	18.60	62.89	51.78	25.83	43.84	0.42	5.78	32.44
	Baseline	300B Pile → 200B Ger. (25% R)		–	36.05	21.59	47.56	60.49	23.82	25.00	17.20	63.49	51.14	25.70	43.84	0.36	5.97	32.48
	TIES	300B Pile	300B SP	25	25.14	21.67	25.88	37.83	19.87	24.65	16.00	52.12	50.67	24.36	48.78	0.03	0.02	26.69
				50	26.31	19.97	28.70	37.83	20.97	24.56	12.40	54.52	53.51	23.50	51.25	0.06	0.02	27.20
				75	41.35	19.80	48.11	60.06	22.21	25.10	16.00	65.94	48.93	22.03	40.57	0.80	6.36	32.10
			300B Pile → 300B SP (5% R)	25	32.40	19.97	35.02	51.74	21.41	24.10	14.20	58.54	52.17	24.97	46.02	0.17	0.44	29.32
				50	38.98	21.33	45.71	44.62	23.12	24.11	16.60	63.55	50.99	25.46	44.00	1.16	3.98	31.05
				75	44.80	23.21	53.79	58.84	23.69	24.84	19.40	68.12	52.96	23.01	39.24	2.52	13.02	34.42
	Baseline	300B Pile → 300B SP (5% R)		–	46.55	23.55	55.01	57.92	24.22	25.94	20.60	69.37	54.22	23.38	38.35	1.99	15.70	35.14
10B	TIES	300B Pile	300B SP	25	25.45	22.18	25.63	52.45	18.66	26.93	16.20	53.86	48.15	23.75	48.82	0.00	0.01	27.85
				50	31.85	21.16	35.56	62.17	19.46	24.08	16.60	60.50	51.14	24.24	45.39	0.08	0.01	30.17
				75	66.93	33.02	66.75	66.02	24.59	27.03	25.00	74.21	62.83	21.91	34.71	6.90	35.34	41.94
	Baseline	300B Pile → 300B SP (5% R)		–	73.24	39.42	74.24	70.80	26.83	28.79	30.60	78.02	68.67	23.01	35.02	13.32	57.86	47.68

TfQA: Truthful QA, WG:WinoGrande, NQ: Natural Questions, OBQA: OpenBook QA, TrQA:TriviaQA

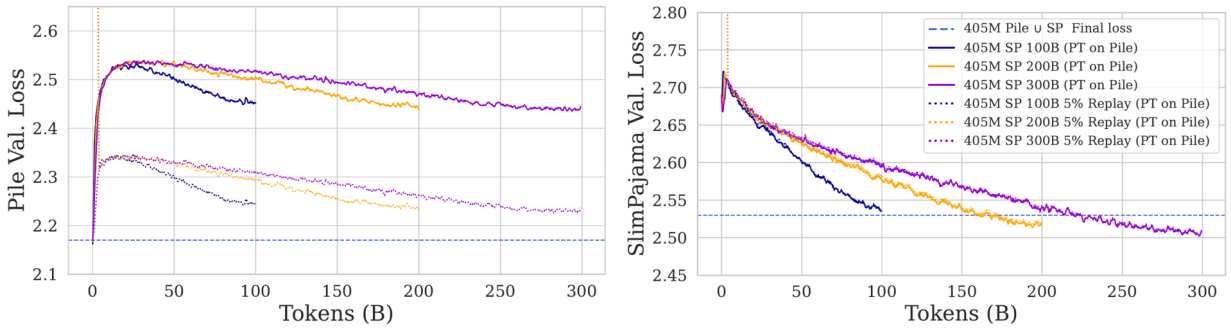
Table 20. German LM Eval Performance of Models Merged with TIES v.s. Continual Pre-training. Normalized accuracy is reported for HellaSwag and exact match (EM) is reported for NaturalQuestions and TriviaQA. All other tasks report unnormalized accuracy. MMLU and TriviaQA are evaluated 5-shot, while all other tasks are zero-shot.

Model Size	Merging Method	Base Model	Task Model(s)	TopK%	MMLU DE	ARC-C DE	Hella-Swag DE	TruthfulQA DE	AVG
405M	TIES	300B Pile	200B Ger.	25	22.86	20.65	24.78	21.05	22.33
				75	24.38	19.54	28.67	25.83	24.60
				50	23.47	21.08	25.64	23.50	23.42
			300B Pile → 200B Ger. (25% Replay)	25	23.89	19.45	26.76	25.70	23.95
				50	25.12	18.86	29.45	26.32	24.94
				75	23.91	18.94	30.81	24.60	24.57
	Baseline	300B Pile → 200B Ger. (25% Replay)		–	23.78	19.20	31.04	25.58	24.90

TfQA: Truthful QA, WG:WinoGrande, NQ: Natural Questions, OBQA: OpenBook QA, TrQA:TriviaQA

D.1.3. Replay for Different Dataset Sizes

To ablate the insights gathered from the experiments in Sec. 8.6.2 we vary the size of the continued pre-training on SlimPajama with 100B, 200B, and 300B tokens. The respective linear warmup schedules and cosine decays are fitted to the altered length of training. As expected, we observe a consistent decrease in loss at the end of training for SlimPajama with an increased number of training tokens (see Fig. 29). We further observe similar trends concerning the effect of replay on forgetting and adapting to novel data. In conclusion, the simple combination of LR re-warming, LR re-decaying, and replay is effective at different dataset sizes.



(a) Pile Validation Loss

(b) SlimPajama Validation Loss

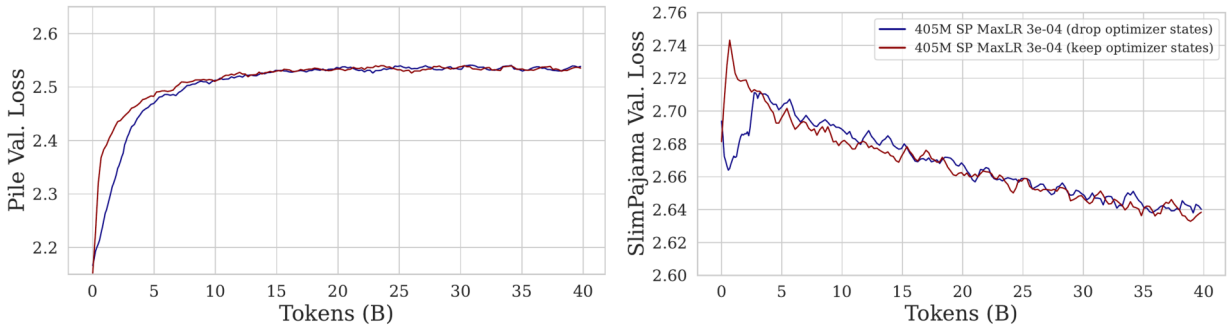
Figure 29. Replay v.s. no replay when re-warming the learning rate and continually pre-training on different amounts of data. We train 405M parameter models with and without 5% replay on 100B, 200B, and 300B tokens of SlimPajama data. Each model starts from a checkpoint pre-trained on 300B tokens of Pile and re-warms the learning rate using a linear warmup and cosine decay schedule fitted to its dataset size.

D.1.4. The effect of resetting optimizer states at dataset transitions

In all of our experiments, we drop optimizer states between dataset transitions to show that our techniques can be used to continually pre-training open-source LLMs, which rarely provide optimizer states. In this section, we investigate whether keeping the optimizer states from pre-training can resolve instabilities seen during continual pre-training. Figure 30 reports validation loss curves during the continual pre-training of a 405M parameter decoder-only transformer on 40B tokens of SlimPajama data. Each model starts from a checkpoint pre-trained on 300B tokens of Pile and re-warms and re-decays the learning rate using a linear warmup and cosine decay schedule fit to a dataset size of 300B tokens ($\eta_{max} = 3 \cdot 10^{-4}$, $\eta_{min} = 3 \cdot 10^{-5}$). We observe that, initially, both models follow slightly different trajectories. However, after 40B tokens of training both models reach similar loss values with differences attributable to randomness. We therefore conclude that, in this setting, keeping optimizer states does not affect continual pre-training compared to discarding them. We note that this is to be expected, as the relatively large momentum coefficient values used for LLM pre-training (0.9 and 0.95 for β_1 and β_2) cause the contribution of the starting optimizer states to quickly decay to 0. For instance, after 1000 steps of optimization, the moment estimates from pre-training will contribute to $< 0.0044\%$ of the current moment estimates.

D.1.5. Qualitative evaluation of German models

In this section, we provide a brief qualitative evaluation of the models trained on German Common crawl (Sec. 8.6.3). We select five German prompts that contain various peculiarities of the German language (see Tab. 21). We then generate a fixed token-length response for each of the models trained or continually pre-trained on German Common Crawl. As a baseline, we also evaluate the same model trained only on the Pile.



(a) Pile Validation Loss

(b) SlimPajama Validation Loss

Figure 30. Keeping v.s. dropping optimizer states when transitioning from Pile to Slim Pajama. We train a 405M parameter decoder-only transformer on 40B tokens of SlimPajama data. Each model starts from a checkpoint pre-trained on 300B tokens of Pile and re-warms and re-decays the learning rate using a linear warmup and cosine decay schedule fitted to a dataset size of 300B tokens ($\eta_{max} = 3 \cdot 10^{-4}$,$\eta_{min} = 3 \cdot 10^{-5}$). We observe that, initially, both models follow slightly different trajectories. However, after 40B tokens of training both models reach similar loss values with differences attributable to randomness.

We provide the responses and a manual translation of the response in Table 22. While the relatively small 405M parameter models generally do not produce meaningful sentences, the model trained on German Common Crawl generally produces grammatically correct sentences. This includes correct upper and lower casing, the generation of compound words, and the use of Umlaut characters. In all cases, responses tend to be repetitive after a few words. The incompleteness and fractional words of some outputs can be attributed to the number of tokens generated being very short. Longer sequences tend to provide syntactically correct sentences but also tend to repeat themselves. The limited number of samples and the generally low quality of the generated text does not allow us to make robust statements of the qualitative differences between the individual models trained on German Common Crawl. However, the aforementioned problems of repetitiveness as well as grammatical errors seem to be significantly stronger on the Pile baseline, which generally fails to even respect the context given in the prompt. This is expected as the amount of German text is significantly smaller on the Pile.

Thus we conclude that there is an observable effect on the generative quality of the German language from the models that used the data of German Common Crawl during training.

Table 21. German phrases used for qualitative evaluation alongside their translation (for readability) and justification.

Ref.	Prompt	Translation	Reasoning
P_1	Eine typisch deutsche Mahlzeit ist	A typical German meal is	Conversational sentence.
P_2	Döner ist besser als Poutine weil	Doener is superior to poutine because	This prompt contains an Umlaut.
P_3	Das Pferd isst keinen Gurkensalat.	The horse does not eat cucumber salad	Short sentence with compound word
P_4	Handy ist ein deutsches Wort	Cellphone is a german word	Prompt with an anglicism (Handy = Cellphone)
P_5	Der Eiersollbruchstellenverursacher ist	The egg breaker is	Long compound word

Table 22. Generated text from various prompt-model combinations. All text was generated with a fixed token length and thus is sometimes incomplete. A manual translation of the text is also provided to give the non-German speaker an idea of the generative quality. The translation is as literal as possible to allow for adequate assessment.

Prompt Ref. (translation)	Model	Fixed Length Response	Translation
P_1: (A typical German meal is	405M 500B Pile ∪ SP	ein Apfelkuchen. Wie viel kostet es, einen Apfelk	an apple pie. How much does an [Apple pie incomplete] cost
	405M 200B German	ein Gericht, das in der Regel . aus mehreren Gängen besteht	a dish, that usually consists of multiple courses.
	405M 300B Pile → 200B German (25% Replay)	das Bier. Wie wäre es mit einem leckeren Eis?	Beer. Would you like some delicious ice cream?
	405M 300B Pile → 200B German	der Kartoffelsalat. Kartoffelsalat	the potato salad potato salad
	405M 300B Pile	die Kinderwunsch Die Kinderwunsch ist eine der wicht	the desire to have kids The desire to have kids is the most [important incomplete]
P_2: Doener is superior to poutine because	405M 500B Pile ∪ SP	viel besser schmeckt. Wenn du dich für die Küche	tastes much better. If you are (missing verb) in the kitchen
	405M 200B German	es so lecker ist. Antworten Schreibe einen Kommentar An	it is so tasty. Answer Write a comment to
	405M 300B Pile → 200B German (25% Replay)	er viel schneller zubereitet ist, Wie viel kostet ein D	it is much faster to cook. How much does a D [incomplete] cost
	405M 300B Pile → 200B German	es mehr Fleisch enthält. Wenn du dich für einen Döner ents	it contains more meat. If you opt [incomplete] for a Doener
	405M 300B Pile	er eine komplette Kombination von Kombinationen von Kombinationen von K	he a complete combination of combinations. of combinations of combinations of K [incomplete]
P_3: (The horse does not eat cucumber salad)	405M 500B Pile ∪ SP	Wenn du dich für die Kategorie habe, dann ist das Pferd ke	If you are interested in the category then
	405M 200B German	Wenn ich das richtig verstanden habe, dann ist das Pferd ke	If I understood this correctly, then the horse is horse is [incomplete]
	405M 300B Pile → 200B German (25% Replay)	Wie kann ich meinem Pferd helfen? Wenn du dein	How can I help my horse? If your
	405M 300B Pile → 200B German	Wenn du das Pferd nicht magst, dann kanns du es auch mit einem	If you don't like the horse, then you can also [missing verbt] with a
	405M 300B Pile	Die Pferde sind nicht nur die Pferde, sondern auch die Pfer	The horses are not only the horses, but also the horses (last word incomplete)
P_4: Cellphone is a german word	405M 500B Pile ∪ SP	das aus dem Lateinischen stammt. Wie funktioniert das mit dem Handy?	that originates from latin. How does this work with the cellphone?
	405M 200B German	das in der deutschen Sprache für das Handy steht. Es ist ein Wort,	that refers to the cellphone in German. It is a word, [incomplete]
	405M 300B Pile → 200B German (25% Replay)	das für das Telefon steht. Es ist ein Telefon, das mit einem Handy	that translates to telephone. It is a telephone
	405M 300B Pile → 200B German	das aus dem Englischen übersetzt "Handy" bedeutet.	which translates from English as "cellphone".
	405M 300B Pile	das in der Tat nicht nur für die deutsche Sprache ist, sondern auch für die deut	which is indeed not only for the German language, but also for the (last word incomplete)
P_5:(The egg breaker is)	405M 500B Pile ∪ SP	der Eiersollbruchstellenverursacher. Eiersollbruchstellenverurs	the egg breaker. Egg [breaker, incomplete]
	405M 200B German	ein kleiner, aber feiner Unterschied. Ein Eiersollbru	a small, but important difference, An egg [breaker, incomplete]
	405M 300B Pile → 200B German (25% Replay)	der Mensch. Wenn wir uns die Eier in den Eierstock legen, dann ist das ein ganz normaler Vorgang.	the man. When we place the eggs for us in the ovary, then this is a natural process.
	405M 300B Pile → 200B German	der Eierkocher. Eierkocher Eierkocher sind	the egg stove. Egg stove Egg stoves are
	405M 300B Pile	der Einsatz von Einsatzkraftwerken für die Einsatzfahrze	the use of emergency power plants for the emergency vehicles emergency [vehicle, incomplete]

D.1.6. Aggregated LM Evaluation Results

We evaluate our models on the following English and German LM evaluation tasks:

(1) **HellaSwag** (Zellers et al., 2019) and **HellaSwag DE**: An English commonsense reasoning benchmark composed of multiple-choice questions that are deliberately designed to confuse language models. HellaSwag DE is a German translation of the HellaSwag benchmark.

(2) **AI2 Reasoning Challenge (ARC)** (Clark et al., 2018): An English commonsense reasoning benchmark composed of science examination questions in multiple-choice format. The $7{,}787$ total questions have been divided into an easy subset with $5{,}197$ questions and a hard subset with $2{,}590$ questions. ARC-c DE is a German translation of the challenge subset of questions.

(3) **BoolQ** (Clark et al., 2019): An English reading comprehension benchmark composed of $15{,}942$ yes/no question-answering samples. Each example is split into a question, relevant paragraph, and the solution.

(4) **MathQA** (Amini et al., 2019): An English math word problem benchmark composed of multiple-choice questions across various areas of mathematics.

(5) **MMLU** (Hendrycks et al., 2021) and **MMLU-DE**: An English benchmark designed to evaluate both zero-shot and few-shot scenarios, in order to evaluate both the general knowledge and on-the-fly problem solving of the model under test. MMLU covers a broad range of subjects. MMLU-DE is a German translation of the MMLU question set, which was translated by the OpenAI GPT 3.5 API.

(6) **OpenBookQA (OBQA)** (Mihaylov et al., 2018): An English question-answering benchmark modeled after real-world open-book exams for assessing human understanding of a particular subject. Questions about elementary science are paired with scientific facts and common knowledge, which the model is intended to use in multi-hop reasoning.

(7) **PIQA** (Bisk et al., 2019): An English question-answering benchmark designed to test the physical commonsense reasoning abilities of the model. Most questions focus on applying uncommon solutions to everyday situations, which requires understanding of the physical world.

(8) **WinoGrande** (Sakaguchi et al., 2019): An English natural language understanding benchmark that involves determining when two or more expressions in a text refer to the same entity. The benchmark includes a diverse set of sentences and a new evaluation metric that rewards models for making human-like predictions.

(9) **TruthfulQA** and **TruthfulQA DE** (Lin et al., 2022): An English question-answering benchmark designed to evaluate the truthfulness of generated answers to questions. The questions are designed to contain common human misunderstandings that lead to incorrect answers. TruthfulQA DE is a German translation of the TruthfulQA benchmark.

(10) **Natural Questions** (Kwiatkowski et al., 2019): Is an English question-answering benchmark consisting of search queries submitted to the Google search engine.

(11) **TriviaQA** (Joshi et al., 2017): Is an English question-answering benchmark comprised of question-answer pairs provided by trivia enthusiasts. Its main focus is to determine a model's general world knowledge.

Table 23. All Zero-shot and Few-shot results on popular LM benchmarks. Normalized accuracy is reported for HellaSwag and exact match (EM) is reported for NaturalQuestions and TriviaQA. All other tasks report unnormalized accuracy. MMLU and TriviaQA are evaluated 5-shot, while all other tasks are zero-shot. We observe **on average**, as expected, that 10B param. models outperform their 405M counterparts and that the English-only 405M models outperform their German-trained counterparts.

Model Size	Training Tokens	HellaSwag	ARC-c	ARC-e	BoolQ	MathQA	MMLU	OBQA	PIQA	WG	TfQA1	TfQA2	NQ	TrQA	AVG
10B	300B Pile	68.46	34.81	69.49	68.20	27.34	27.28	27.20	76.82	62.51	20.44	33.68	6.65	41.92	43.45
	300B SP	70.38	36.77	71.93	68.04	24.76	27.42	28.20	76.99	65.04	22.40	33.99	11.25	52.63	45.37
	300B Pile → 300B SP	73.66	37.37	73.02	73.18	26.43	29.94	30.20	78.51	66.30	23.26	35.04	12.99	57.94	47.53
	300B Pile → 300B SP (5% Replay)	73.24	39.42	74.24	70.80	26.83	28.79	30.60	78.02	68.67	23.01	35.02	13.32	57.86	47.68
	600B Pile ∪ SP	73.39	39.25	73.57	72.05	26.83	37.78	27.80	77.58	67.32	23.13	36.16	12.41	56.73	48.00
405M	300B Pile	40.95	22.01	51.77	59.24	24.12	26.18	19.80	66.59	53.83	24.85	42.11	0.91	8.97	33.95
	300B SP	44.22	21.76	54.08	59.63	22.71	26.18	19.60	68.23	49.80	22.64	38.63	1.69	14.18	34.11
	300B Pile → 300B SP	46.22	22.70	54.04	57.43	24.22	25.28	21.20	69.26	54.46	23.13	38.91	2.02	15.23	34.93
	300B Pile → 300B SP (0.5% Replay)	46.26	21.42	55.18	60.18	24.09	25.90	19.80	68.72	55.64	23.38	39.13	1.86	15.58	35.16
	300B Pile → 300B SP (1% Replay)	46.09	23.55	54.88	57.95	23.42	26.02	20.60	68.66	54.78	22.89	37.83	1.91	15.29	34.91
	300B Pile → 300B SP (5% Replay)	46.55	23.55	55.01	57.92	24.22	25.94	20.60	69.37	54.22	23.38	38.35	1.99	15.70	35.14
	300B Pile → 300B SP (10% Replay)	46.15	21.93	54.42	58.44	23.62	25.53	21.40	68.77	54.54	23.75	37.87	2.38	15.00	34.91
	300B Pile → 300B SP (50% Replay)	44.77	23.55	53.75	60.49	25.03	26.04	20.00	68.72	53.91	23.50	39.69	1.14	13.67	34.94
	600B Pile ∪ SP	45.06	23.55	52.99	55.57	23.12	26.65	18.20	69.37	52.72	23.50	38.81	1.72	14.63	34.30
	200B Ger.	27.47	17.15	29.80	45.11	21.11	25.27	13.40	56.75	52.17	26.07	46.02	0.03	0.31	27.74
	300B Pile → 200B Ger.	30.02	19.20	32.74	61.80	20.40	23.34	13.40	58.05	49.96	24.48	44.01	0.19	1.93	29.20
	300B Pile → 200B Ger. (1% Replay)	30.84	18.09	36.49	57.80	20.60	24.69	14.60	59.52	49.49	24.72	45.74	0.11	2.81	29.65
	300B Pile → 200B Ger. (5% Replay)	32.88	21.25	41.12	60.06	22.58	24.94	15.80	62.35	51.30	25.46	45.36	0.11	4.62	31.37
	300B Pile → 200B Ger. (10% Replay)	34.10	20.65	43.73	52.60	22.35	25.41	18.40	63.06	50.04	25.34	44.33	0.19	4.59	31.14
	300B Pile → 200B Ger. (25% Replay)	36.05	21.59	47.56	60.49	23.82	25.00	17.20	63.49	51.14	25.70	43.84	0.36	5.97	32.48
	300B Pile → 200B Ger. (50% Replay)	38.38	20.65	49.54	60.09	24.12	26.45	18.60	65.78	50.51	25.95	42.47	0.69	7.87	33.16
	600B Pile ∪ Ger.	36.99	19.37	47.69	59.27	23.99	25.62	17.40	64.91	52.96	23.87	42.10	0.47	6.92	32.43
	100B×3 SP InvSqrt	43.20	20.31	51.35	60.70	21.91	25.38	18.20	68.34	52.01	19.71	35.91	1.94	14.35	33.33
	100B×3 SP CosineInf	43.22	22.61	51.05	59.57	22.78	26.48	18.60	68.17	53.51	23.01	36.90	1.83	14.29	34.00
	100B×3 SP Cosine	43.38	20.05	50.46	60.06	22.75	25.23	18.00	67.57	52.17	23.13	39.74	1.44	13.40	33.65

TfQA: Truthful QA, WG:WinoGrande, NQ: Natural Questions, OBQA: OpenBook QA, TrQA:TriviaQA

Table 24. All Zero-shot and Few-shot results on popular German LM benchmarks. Normalized accuracy is reported for HellaSwag. All other tasks report unnormalized accuracy. MMLU is evaluated 5-shot, while all other tasks are zero-shot. Unexpectedly, we observe **on average**, that English language models match the performance of their German counterparts. Further inspection shows that performance is close to random chance on MMLU and ARC-C for all models, while German models perform better on HellaSwag and English models perform better on Truthful QA. This has the effect of canceling out the perceived improvements in the overall score.

Training Tokens	MMLU DE	ARC-C DE	HellaSwag DE	TruthfulQA MC1 DE	AVG
300B Pile	24.92	18.77	27.09	26.93	24.43
300B SP	23.28	17.49	27.03	27.91	23.93
300B Pile → 300B SP	25.34	17.32	27.43	26.81	24.22
300B Pile → 300B SP (0.5% Replay)	25.95	17.83	27.35	24.72	23.96
300B Pile → 300B SP (1% Replay)	25.44	19.03	27.62	25.34	24.36
300B Pile → 300B SP (5% Replay)	24.82	16.89	27.09	25.95	23.69
300B Pile → 300B SP (10% Replay)	25.24	19.11	27.39	26.68	24.61
300B Pile → 300B SP (50% Replay)	25.11	18.86	27.51	27.17	24.66
600B Pile ∪ SP	24.43	18.26	27.36	26.44	24.12
200B Ger.	23.74	18.26	29.53	26.32	24.46
300B Pile → 200B Ger.	24.29	19.62	31.23	24.85	25.00
300B Pile → 200B Ger. (1% Replay)	23.62	19.62	31.21	24.72	24.80
300B Pile → 200B Ger. (5% Replay)	24.82	18.94	31.03	26.68	25.37
300B Pile → 200B Ger. (10% Replay)	23.51	20.05	31.21	25.58	25.09
300B Pile → 200B Ger. (25% Replay)	23.78	19.20	31.04	25.58	24.90
300B Pile → 200B Ger. (50% Replay)	23.80	20.48	30.91	24.60	24.95
500B Pile ∪ Ger.	24.53	18.43	30.45	25.70	24.78

D.1.7. Aggregated average final accuracy

Table 25. Final loss of models reported in sections 8.6.1, 8.6.2, 8.6.3, 8.6.4, and 8.7.4. The loss is averaged over the last 100 iterations of training sampled at intervals of 10 iterations. The standard error is ≤ 0.001 for all models so we don't report the specific value. We note, however, that this averaging does not correspond to Monte Carlo sampling over different random seems and is merely meant to reduce noise. We observe that models using more replay achieve a better adaptation-forgetting trade-off (AVG Loss). Interestingly, the model using 50% replay archives nearly identical loss values while seeing 150B fewer tokens on SlimPajama. All evaluation

				Validation Loss		
Model Size	Training Tokens	Max LR	Schedule	$\mathcal{D}_0$ (Pile)	$\mathcal{D}_0$ (SP/German)	AVG
405M	300B Pile → 300B SP	Constant	Cosine	2.42	2.55	2.48
		1.5×10^{-4}	Cosine	2.43	2.51	2.47
		3×10^{-4}	Cosine	2.44	2.50	2.47
		6×10^{-4}	Cosine	2.48	2.50	2.49
	300B Pile → 200B Ger.	Constant	Cosine	3.22	1.21	2.22
		1.5×10^{-4}	Cosine	3.47	1.13	2.30
		3×10^{-4}	Cosine	3.56	1.11	2.34
		6×10^{-4}	Cosine	3.63	1.11	2.37
	300B Pile	3×10^{-4}	Cosine	2.17	2.70	2.44
	300B SP	3×10^{-4}	Cosine	2.51	2.53	2.52
	200B Ger.	3×10^{-4}	Cosine	3.97	1.17	2.57
	500B Pile ∪ 200B Ger.	3×10^{-4}	Cosine	2.26	1.25	1.75
	300B Pile ∪ 300B SP	3×10^{-4}	Cosine	2.17	2.53	2.35
	300B Pile → 300B SP	3×10^{-4}	Cosine	2.44	2.50	2.47
	300B Pile → 300B SP (0.5% Replay)	3×10^{-4}	Cosine	2.27	2.50	2.39
	300B Pile → 300B SP (1% Replay)	3×10^{-4}	Cosine	2.26	2.50	2.38
	300B Pile → 300B SP (5% Replay)	3×10^{-4}	Cosine	2.23	2.51	2.37
	300B Pile → 300B SP (10% Replay)	3×10^{-4}	Cosine	2.21	2.51	2.36
	300B Pile → 300B SP (50% Replay)	3×10^{-4}	Cosine	2.16	2.54	2.35
	300B Pile → 200B Ger.	3×10^{-4}	Cosine	3.56	1.11	2.34
	300B Pile → 200B Ger (1% Replay)	3×10^{-4}	Cosine	2.83	1.12	1.97
	300B Pile → 200B Ger (5% Replay)	3×10^{-4}	Cosine	2.57	1.12	1.85
	300B Pile → 200B Ger (10% Replay)	3×10^{-4}	Cosine	2.46	1.13	1.80
	300B Pile → 200B Ger (25% Replay)	3×10^{-4}	Cosine	2.33	1.16	1.75
	300B Pile → 200B Ger (50% Replay)	3×10^{-4}	Cosine	2.24	1.22	1.73
10B	300B Pile → 300B SP	1.2×10^{-4}	Cosine	1.98	2.00	1.99
	300B SP	1.2×10^{-4}	Cosine	2.08	2.05	2.07
	300B Pile	1.2×10^{-4}	Cosine	1.75	2.24	1.99
	600B Pile ∪ SP	1.2×10^{-4}	Cosine	1.72	2.02	1.87
	300B Pile → 300B SP (5% Replay)	1.2×10^{-4}	Cosine	1.79	2.00	1.89
405M	300B SP	3×10^{-4}	Cosine Inf	2.51	2.53	2.52
			InvSqrt Inf	2.51	2.54	2.53
			Cosine	2.51	2.53	2.52
	100B×3 SP	3×10^{-4}	Cosine Repeats	2.53	2.55	2.54
			Cosine Inf	2.58	2.61	2.59
			Cosine	2.59	2.61	2.60

D.2. Model hyperparameters

Description	Value
10B Transformer Model	
Parameters	9,642,249,216
Non-Embedding Parameters	9,408,678,912
Num layers	36
Hidden size	4608
Num attention heads	36
405M Transformer Model	
parameters	405,334,016
Non-Embedding Parameters	353,822,720
Num layers	24
Hidden size	1024
Num attention heads	16
Common	
Optimizer	AdamW
β_1,β_2	0.9,0.95
Batch size	1104
Sequence length	2048
Hidden activation	GeLU
Weight decay	0.1
Gradient clipping	1.0
Decay	Cosine
Positional embedding	Rotary
GPT-J-Residual	True
Weight tying	False
Vocab Size	50432
Rotary PCT	0.25

Table 26. Hyperparameters of our 405M and 10B parameter transformer LLMs.

Description	Value
10B Model- Cosine Schedule	
Max learning rate (η_{max})	$1.2 \cdot 10^{-4}$
Min learning rate (η_{min})	$1.2 \cdot 10^{-5}$
Warmup percent (T_{warmup})	1
405M Model - Cosine Schedule	
Max learning rate (η_{max})	$3 \cdot 10^{-4}$
Min learning rate (η_{min})	$3 \cdot 10^{-5}$
Warmup percent (T_{warmup})	1
405M Model - Infinite LR Schedule Common	
Max learning rate (η_{max})	$3 \cdot 10^{-4}$
Min learning rate (η_{min})	$3 \cdot 10^{-5}$
Constant learning rate (η_{const})	$1.65 \cdot 10^{-4}$
Warmup percent (T_{warmup})	1
Cooldown iters percent (T_{cd})	60
Constant iters percent (T_{ann})	25
Inverse Square root cooldown schedule	
Timescale (α)	10

Table 27. Hyperparameters of LR schedules. Unless otherwise specified in the text, we use these values.