Université de Montréal

Deep Learning on Signals: Discretization Invariance, Lossless Compression and Nonuniform Compression

par

Léa Demeule

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

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Informatique

July 28, 2023

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Deep Learning on Signals: Discretization Invariance, Lossless Compression and Nonuniform Compression

présenté par

Léa Demeule

a été évalué par un jury composé des personnes suivantes :

Liam Paull

(président-rapporteur)

Glen Berseth

(directeur de recherche)

Christopher Pal

(membre du jury)

Résumé

Une grande variété d'information se prête bien à être interprétée comme *signal*; à peu près toute *quantité fluctuant continuellement dans l'espace* se trouve inclue. La vie quotidienne abonde d'exemples; les images peuvent être vues comme une variation de couleur à travers l'espace bidimensionnel; le son, la pression à travers le temps; les environnements physiques, la matière à travers l'espace tridimensionnel.

Les calculs sur ce type d'information requièrent nécessairement une transformation de la forme *continue* vers la forme *discrète*, ce qui est accompli par le processus de *discrétisation*, où seules quelques valeurs du *signal continu* sous-jacent sont observées et compilées en un *signal discret*.

Sous certaines conditions, à l'aide seulement d'un nombre fini de valeurs observées, le *signal discret* capture la totalité de l'information comprise dans le *signal continu*, et permet de le reconstruire parfaitement. Les divers systèmes de senseurs permettant d'acquérir des signaux effectuent tous ce processus jusqu'à un certain niveau de fidélité, qu'il s'agisse d'une caméra, d'un enregistreur audio, ou d'un système de capture tridimensionnelle.

Le processus de discrétisation n'est pas unique par contre. Pour un seul signal continu, il existe une infinité de signaux discrets qui lui sont équivalents, et entre lesquels les différences sont contingentes. Ces différences correspondent étroitement aux différences entre systèmes de senseurs, qui ont chacun leur niveau de fidélité et leurs particularités techniques.

Les réseaux de neurones profonds sont fréquemment spécialisés pour le type de données spécifiques sur lesquels ils opèrent. Cette spécialisation se traduit souvent par des *biais inductifs* qui supportent des *symétries* intrinsèques au type de donnée. Quand le comportement d'une architecture neuronale reste inchangé par une certaine opération, l'architecture est dite *invariante* sous cette opération. Quand le comportement est affecté d'une manière identique, l'architecture est dite *équivariante* sous cette opération.

Nous explorons en détail l'idée que les architectures neuronales puissent être formulées de façon plus générale si nous abstrayions les spécificités contingentes des *signaux discrets*, qui dépendent généralement de particularités de systèmes de senseurs, et considérions plutôt l'unique *signal continu* représenté, qui est la réelle information d'importance. Cette idée

correspond au biais inductif de *l'invariance à la discrétisation*, qui reconnaît que les signaux ont une forme de *symétrie à la discrétisation.*

Nous formulons une architecture très générale qui respecte ce biais inductif. Du fait même, l'architecture gagne la capacité d'être évaluée sur des discrétisations de taille arbitraire avec une grande robustesse, à l'entraînement et à l'inférence. Cela permet d'accéder à de plus grands corpus de données pour l'entraînement, qui peuvent être formés à partir de discrétisations hétérogènes. Cela permet aussi de déployer l'architecture dans un plus grand nombre de contextes où des systèmes de senseurs produisent des discrétisations variées.

Nous formulons aussi cette architecture de façon à se généraliser à n'importe quel nombre de dimensions, ce qui la rend idéale pour une grande variété de signaux.

Nous notons aussi que son coût d'évaluation diminue avec la taille de la discrétisation, ce qui est peu commun d'architectures conçues pour les signaux, qui ont généralement une discrétisation fixe.

Nous remarquons qu'il existe un lien entre l'invariance à la discrétisation, et la distinction séparant *l'équivariance à la translation discrète* et *l'équivariance à la translation continue.* Ces deux propriétés reflètent la même symétrie à la translation, mais l'une est plus diluée que l'autre. Nous notons que la plus grande part de la littérature entourant les architectures motivées par *l'algèbre générale* omettent cette distinction, ce qui affaiblit la force des biais inductifs implémentés.

Nous incorporons aussi dans notre méthode la capacité d'implémenter d'autres *invariances* and *equivariances* plus générales à l'aide de couches formulées à partir de l'opérateur de dérivée partielle. La symétrie à la translation, la rotation, la réflexion, et la mise à l'échelle peuvent être adoptées, et l'expressivité et l'efficacité en paramètres de la couche résultante sont excellentes.

Nous introduisons aussi un nouveau *bloc résiduel Laplacien*, qui permet de compresser l'architecture sans perte en fonction de la densité de la discrétisation. À mesure que le *nombre d'échantillons de la discrétisation* réduit, le *nombre de couches requises pour l'évaluation* diminue aussi. Le coût de calcul de l'architecture diminue ainsi à mesure que certaines de ses couches sont retirées, mais elle se comporte de façon virtuellement identique; c'est ainsi une forme de compression sans perte qui est appliquée. La validité de cette compression sans perte est prouvée théoriquement, et démontrée empiriquement. Cette capacité est absente de la littérature antérieure, au meilleur de notre savoir.

Nous greffons à ce mécanisme une forme de *décrochage Laplacien*, qui applique effectivement une augmentation spectrale aux données pendant l'entraînement. Cela mène à une grande augmentation de la robustesse de l'architecture à des dégradations de qualité de la

discrétisation, sans toutefois compromettre sa capacité à performer optimalement sur des discrétisations de haute qualité. Nous n'observons pas cette capacité dans les méthodes comparées.

Nous introduisons aussi un algorithme d'initialisation des poids qui ne dépend pas de dérivations analytiques, ce qui permet un prototypage rapide de couches plus exotiques.

Nous introduisons finalement une méthode qui généralise notre architecture de l'application à des *signaux échantillonnés uniformément* vers des *signaux échantillonnés non uniformément*. Les garanties théoriques que nous fournissons sur son efficacité d'échantillonnage sont positives, mais la complexité ajoutée par la méthode limite malheureusement sa viabilité.

Mots-clés: intelligence artificielle, apprentissage profond, traitement de signal, traitement de signal multidimensionnel, traitement de signal non uniforme, invariance à la discrétisation, équivariance à la translation, équivariance à la rotation, équivariance à la réflexion, équivariance à la mise à l'échelle, opérateur de dérivée partielle, opérateur neural, normalizing flows

Abstract

Signals are a useful representation for many types of information that consist of *continuously changing quantities*. Examples from everyday life are abundant: images are fluctuations of colour over two-dimensional space; sounds are fluctuations of air pressure over time; physical environments are fluctuations of material qualities over three-dimensional space.

Computation over this information requires that we reduce its *continuous* form to some *discrete* form. This is done through the process of *discretization*, where only a few values of the underlying *continuous signal* are observed and compiled into a *discrete signal*.

This process incurs no loss of information and is reversible under some conditions. Sensor systems, such as cameras, sound recorders, and laser scanners all effectively perform discretization when they capture signals, and they preserve them up to a certain degree.

This process is not unique, however. Given a single continuous signal, there are countless discrete signals that correspond to it, and the specific choice of discrete signal is generally contingent. Sensor systems all have different technical characteristics that lead to different discretizations.

Deep neural network architectures are often tailored to respect the fundamental properties of the specific data type they operate on. Their behaviour often implements *inductive biases* that respect some fundamental *symmetry* of the data. When behaviour is unchanged by some operation, the architecture is *invariant* under it. When behaviour transparently reproduces some operation, the architecture is *equivariant* under it.

We explore in great detail the idea that neural network architectures can be formulated in a more general way if we abstract away the contingent details of the *discrete signal*, which generally depend on the implementation details of a sensor system, and only consider the underlying *continuous signal*, which is the true information of interest. This is the intuitive idea behind *discretization invariance*.

We formulate a very general architecture that implements this inductive bias. This allows handling discretizations of various sizes with much greater robustness, both during training and inference. We find that training can leverage more data by allowing heterogeneous discretizations, and that inference can apply to discretizations produced by a broader range of sensor systems. The architecture is agnostic to dimensionality, which makes it widely applicable to different types of signals. The architecture also lowers its computational cost proportionally to the sample count, which is unusual and highly desirable.

We find that discretization invariance is also key to the distinction between *discrete shift equivariance* and *continuous shift equivariance*. We underline the fact that the majority of previous work on architecture design motivated by abstract algebra fails to consider this distinction. This nuance impacts the robustness of convolutional neural network architectures to translations on signals, weakening their inductive biases if unaddressed.

We also incorporate the ability to implement more general *invariances* and *equivariances* by formulating steerable layers based on the partial derivative operator, and a set of other compatible architectural blocks. The framework we propose supports shift, rotation, reflection, and scale. We find that this results in excellent expressivity and parameter efficiency.

We further improve computational efficiency with a novel *Laplacian residual* structure that allows lossless compression of the whole network depending on the sample density of the discretization. As the *number of samples* reduces, the *number of layers* required for evaluation also reduces. Pruning these layers reduces computational cost and has virtually no effect on the behaviour of the architecture. This is proven theoretically and demonstrated empirically. This capability is absent from any prior work to our knowledge.

We also incorporate a novel form of *Laplacian dropout* within this structure, which performs a spectral augmentation to the data during training. This leads to greatly improved robustness to changes in spectral volume, meaning the architecture has a much greater tolerance to low-quality discretizations without compromising its performance on high-quality discretization. We do not observe this phenomenon in competing methods.

We also provide a simple data-driven weight initialization scheme that allows quickly prototyping exotic layer types without analytically deriving weight initialization.

We finally provide a method that generalizes our architecture from *uniformly* sampled signals to *nonuniformly* sampled signals. While the best-case theoretical guarantees it provides for sample efficiency are excellent, we find it is not viable in practice because of the complications it brings to the discretization of the architecture.

Keywords: artificial intelligence, deep learning, signal processing, multidimensional signal processing, nonuniform signal processing, discretization invariance, shift equivariance, rotation equivariance, reflection equivariance, scale equivariance, partial derivative operator, neural operator, normalizing flows

Contents

List of tables

List of figures

List of abbreviations

a	Real scalar.
$\mathbf{v}$	Real vector.
s	Signal represented in the *spatial domain*.
$\widehat{s}$	Signal represented in the *spectral domain*.
ϕ	Filter kernel represented in the *spatial domain*.
$\widehat{\phi}$	Filter kernel represented in the *spectral domain*.
$\mathbf{x} \in \mathbf{X}$	Point in the d-dimensional Euclidean *observed spatial* domain.
$\mathbf{z} \in \mathbf{Z}$	Point in the d-dimensional Euclidean *latent spatial* domain.
$\boldsymbol{\omega} \in \boldsymbol{\Omega}$	Point in the d-dimensional Euclidean *observed spectral* domain.
$\boldsymbol{\xi} \in \boldsymbol{\zeta}$	Point in the d-dimensional Euclidean *latent spectral* domain.

$*$	Convolution operator.
$\mathfrak{F}$	Fourier transform.
$\mathfrak{F}^{-1}$	Inverse Fourier transform.
$\mathfrak{T}$	Shift operator.
$\mathfrak{M}$	Mirror operator.
$\mathfrak{D}$	Partial derivative operator.
$\mathfrak{V}$	Volume operator.
$\mathfrak{d}$	Density operator.
$\mathfrak{S}$	Sampling operator.
$\mathfrak{R}$	Reapeating operator.
$\mathfrak{I}$	Interpolation operator.
$\mathbb{1}\{\,\cdot\,\}$	Indicator function.
$\delta(\,\cdot\,)$	Dirac delta.

Remerciements

Merci à Glen Berseth pour la liberté, la richesse de savoir, la confiance et le respect que tu m'as offert dans ce merveilleux jeu d'exploration. Ta contribution a un impact immesurable sur ma capacité à poursuivre cette passion que j'ai à créer, questionner, jouer avec des idées, ma capacité à faire quelque chose de valable pour autrui, et ma capacité à en faire quelque chose de balancé en soi. Elle a ouvert tant de portes, tant de discussions, tant d'introspection, tant d'espace pour évoluer. J'en suis extrêmement reconnaissante.

Merci à Mahtab Sandhu pour tous les efforts et les idées que tu as lancés avec vigueur dans ce projet qui aurait été impossible sans toi.

Merci au Mila et à l'Université de Montréal pour le financement que vous avez octroyé à ma recherche, et pour l'accueil chaleureux que j'ai reçu de votre communauté académique.

Merci à Pauline Palma pour toute ta bienveillance, ta douceur, ta vivacité d'esprit. Tu m'as soutenue dans certains des moments les plus marquants de ma vie, tout court, et dans cette aventure folle en contrées académiques. Je t'adore.

Merci à Isabelle Collin, André Beaudoin, Andrée Parent, Guy Collin, François Demeule, Éric Demeule, Roger Demeule, Pierre Collin, Christine Allen, Étienne Collin et Renaud Collin pour votre support inconditionnel et pour votre ouverture d'esprit. Je me compte si chanceuse d'être si bien entourée. Je vous aime tous.

Merci à Louis Parent pour ton support, ton humour, ta complicité. J'ai une affection profonde pour l'amitié que nous avons formée.

Merci à Jack Richter-Powell pour toutes les conversations qui se tissent sans arrêt entre nous, qui sont tantôt follement joyeuses et légertes, tantôt follement à propos et techniquement précises.

Introduction

Many forms of information that consist of *continuously changing quantities* can be conceptualized as *signals* (section 2.2). Examples from everyday life are abundant: images are fluctuations of colour over two-dimensional space; sounds are fluctuations of air pressure over time; physical environments are fluctuations of material qualities over three-dimensional space.

Computation over this information requires that we reduce its *continuous* form to some *discrete* form. This is done through the process of *discretization* (subsection 2.2.12), where only a few values of the underlying *continuous signal* are observed and compiled into a *discrete signal*.

A very important result in the theory of signals is that discretization can always be formulated in a way that captures all information present in a signal under some mild assumptions (subsection 2.2.13). Sensors that observe real physical signals and translate them to data that can be interpreted by computers all effectively perform discretization.

A consequence of this important result is that there are countless ways of formulating discretization for the same underlying signal. Sensors concretely embody the idea that there exists a great variety in discretization: images can be captured by countless different optical systems and cameras; sounds can be recorded with distinct microphones, amplifiers and capture devices; physical environments can be acquired by technologies that have entirely different modes of operation, such as multi-view reconstruction or laser ranging.

Computation on this data can be formulated in a more general way if we abstract away the contingent details of the *discrete signal*, which depend on the implementation details of the sensor system, and only consider the underlying *continuous signal*, which is the true information of interest. This is the intuitive idea behind *discretization invariance* (subsection 2.2.18).

Deep learning (section 2.1) architectures are typically tailored to the specific data type they operate on. Their behaviour often implements *inductive biases* that respect some fundamental *symmetry* of the data (subsection 2.1.2). When behaviour is unchanged by some operation, the architecture is *invariant* under it. When behaviour transparently reproduces some operation, the architecture is *equivariant* under it.

The most typical example of this is the shift invariance that all convolutional neural network architectures implement in classification tasks. This inductive bias enforces the idea that meaningful information in signals is independent of absolute position. If the task is to recognize animals in images, a cat is a cat, whether it is on the left or right of an image.

We find that *discretization invariance* is extremely useful to approaches tailored to signals. We highlight that this idea has not been thoroughly exploited by previous work.

We formulate a very general architecture that implements *discretization invariance* (chapter 3). This allows handling discretizations of various sizes with much greater robustness, both during training and inference. We find that training can leverage more data by allowing heterogeneous discretizations, and that inference can apply to discretizations produced by a broader range of sensor systems (subsection 4.1.1). The architecture is agnostic to dimensionality, which makes it widely applicable to different types of signals. The architecture also lowers its computational cost proportionally to the sample count, which is unusual and highly desirable.

We underline the fact that the majority of previous work on architecture design motivated by abstract algebra fails to consider the distinction between *continuous shift equivariance* and *discrete shift equivariance*, which weakens the strength of the inductive bias they guarantee.

The basic idea behind this can be shown pragmatically with images. When a camera captures an image, its exact position in space and its exact sensor alignment define what part of the image map to what exact pixel. Minute displacements can shift part of the image between pixels. If an architecture only has *discrete shift equivariance*, the original image and the minutely displaced image are treated as fundamentally different. If an architecture has *continuous shift equivariance*, the two images are treated identically, which is more desirable. Because there are *finite* displacements that result in *discrete* pixel shifts, but *infinite* displacements that result in *continuous* pixel shifts, this scenario is extremely common.

We provide this inductive bias in its full strength in our work (section 3.1).

We also incorporate the ability to implement more general *invariances* and *equivariances* by formulating steerable layers (section 3.2) based on the partial derivative operator, and a set of other compatible architectural blocks. The framework we propose supports shift, rotation, reflection, and scale. We find that this results in excellent expressivity and parameter efficiency (section 4.1).

We further improve computational efficiency with a novel *Laplacian residual* structure (section 3.6) that allows lossless compression of the whole network depending on the sample density of the discretization. As the *number of samples* reduces, the *number of layers* required for evaluation also reduces. Pruning these layers reduces computational cost and has virtually no effect on the behaviour of the architecture. This is proven theoretically and demonstrated empirically (subsection 4.1.3). This capability is absent from any prior work to our knowledge.

We also incorporate a novel form of *Laplacian dropout* (section 3.7) within this structure, which performs a spectral augmentation to the data during training. This leads to greatly improved robustness to spectral degradation, meaning the architecture has a much greater tolerance to low-quality discretizations without compromising its performance on high-quality discretization. We do not observe this phenomenon in competing methods (subsection 4.1.2).

We provide a simple data-driven weight initialization scheme (section 3.8) that allows quickly prototyping exotic layer types without analytically deriving weight initialization. Its formulation avoids the issue of vanishing or exploding gradients within the search process and converges very quickly even with initial weight initializations that are incorrect by orders of magnitude.

We finally provide a method that generalizes our architecture from *uniformly* sampled signals to *nonuniformly* sampled signals (section 3.9). We realize this by adapting sampling patterns through normalizing flows (section 2.3), and adapting the discretization of the operators that constitute our layers to the new local coordinate frame this creates. This overcomes the extremely low sample efficiency of other architectures based on lattice sampling patterns, and instead provides optimal sample efficiency in ideal conditions. This also allows a form of lossy architecture compression that is analogous to compressed sensing techniques. We find this is effective in simple situations (subsection 4.2.1), but less useful in practical scenarios (subsection 4.2.4).

Chapter 1

Related work

We provide an overview of neural network architectures (section 1.1, section 1.2) and weight initialization strategies (section 1.3) that are most relevant to our method.

1.1. Architectures motivated by classical computer vision

Most convolutional network architectures have been formulated from a pragmatic engineering standpoint, with the aim of solving computer vision tasks. They do not address, or only incompletely address the issues of signal discretization. We compare against these methods in section 4.1 and use them as a starting point for our method, so it is useful to overview them quickly.

1.1.1. Standard convolutional neural networks

Standard convolutional neural networks provide a basic implementation of the principle that underlies all convolutional neural networks. Their layers are built on a simple discretization of the convolution operator (subsection 2.2.7, subsection 2.2.20).

A large body of work has led to the formulation of standard convolutional networks that have great computational complexity, that thrive at exploiting parametrization sparsity, that can solve impressively complex tasks by leveraging depth, and that have well-behaved training dynamics (section 1.3) (Fukushima, 1980; LeCun et al., 2002; Glorot and Bengio, 2010; He et al., 2015; Ioffe and Szegedy, 2015; He et al., 2016a,b; Krizhevsky et al., 2017).

Standard convolutional neural networks do not address the challenges of dealing with various discretizations. They are rooted in a single, native discretization, and evaluation of these architectures under other discretizations is either impossible or results in severe performance degradation. We return to this fact when comparing these methods with ours in section 4.1.

1.1.2. Residual convolutional neural networks

Residual convolutional neural networks are often included within standard convolutional neural networks, as they are used ubiquitously (He et al., 2016a,b; Krizhevsky et al., 2017). They expand on the basic formulation of convolutional neural networks by introducing residual connections, which allow information to directly jump from an early layer to a later layer through an additive connection. This makes it trivial for networks to express simple mappings which are close to the identity mapping. This also makes it easier for information to propagate efficiently within deep networks. Both of these effects have a very positive impact on training dynamics.

Residual convolutional neural networks are closely related to the *Laplacian residual* blocks we propose in section 3.6, which also possess additive skip connections but also include a form of filtering.

1.1.3. Multiscale convolutional neural networks

Multiscale convolutional neural networks (Elizar et al., 2022) expand on the framework of standard convolutional networks by introducing dilations, expansion, contractions, and parallel paths among layers that can encourage variety in feature scale, and that provide some robustness to disturbances in the discretization process. They bear some resemblance to the scale equivariant neural networks discussed in subsection 1.2.3.

Multiscale convolutional neural networks do not explicitly deal with the issue of discretization, however, and generally cannot be rediscretized arbitrarily.

1.1.4. Spectrally disentangled convolutional neural networks

Spectrally disentangled convolutional neural networks establish a form of spectral structure or partitioning between layers. Some filtering schemes have been explored for spectrally disentangled residual connections (Singh et al., 2021), and Laplacian pyramids (subsection 2.2.25) have been used in super-resolution methods successfully (Lai et al., 2017).

Spectrally disentangled convolutional neural networks also do not address the challenge of discretization, but partition the signal spectrum similarly to the *Laplacian residual* blocks we propose in section 3.6.

1.1.5. Dynamic convolutional neural networks

Dynamic convolutional neural networks (Han et al., 2021) designate architectures that perform dynamic adaptation of the network structure. This sometimes takes the form of a scaling or resampling operation on the signal they process.

Recasens et al. (2018) is specifically relevant to our method. It allows bending the input signal to magnify salient areas, like the compression method proposed in subsection 3.9.11. However, the operations performed on the discretized signal in its bent coordinate frame ($\mathbf{Z}$) are not brought back to its original coordinate frame ($\mathbf{X}$), unlike we do in subsection 3.9.3 (Equation 3.9.17).

1.2. Architectures motivated by abstract algebra

The design decisions that underlie many neural architectures can be understood through the lens of abstract algebra, which allows describing how functions respect or break symmetries of certain mathematical objects in terms of *invariances* and *equivariances* (Bronstein et al., 2017; Kondor and Trivedi, 2018; Cohen et al., 2019; Finzi et al., 2020; Lang and Weiler, 2020; Bronstein et al., 2021). We return to this in more depth in subsection 2.1.2, but provide a quick overview for now.

The information involved in a machine learning task often possesses some form of fundamental symmetry that we expect an effective solution to respect. By enforcing these symmetries in the design of the architecture, we guarantee these symmetries hold, which can make solutions easier to learn and more robust.

The most typical example in the context of deep learning are convolutional neural networks, which respect the translation symmetry of signals, meaning translations to their input result in no change at all if the output is some form of vector (*invariance*, Equation 2.1.1), or in identical translations to their output if the output is a signal (*equivariance*, Equation 2.1.2). This relationship enforces the idea that the meaningful content of the signal does not depend on its absolute position.

This framework allows describing precisely a wide range of useful symmetries. The most relevant architectures that explicitly make use of this framework are covered here.

1.2.1. Shift equivariant neural networks

We first cover methods that explicitly leverage translation, also called shift, the most common symmetry on signals.

Methods that guarantee this property almost always formulate their layers through some form of convolution operator.

Methods sometimes specifically build on the partial derivative operator (Ruthotto and Haber, 2020; Shen et al., 2020; Jenner and Weiler, 2021), as our method does with its steerable layers (section 3.2). We later show in subsection 2.2.21 that the application of the partial derivative operator on a restricted class of signals can be conceptualized as a convolution operator.

We note that shift equivariance can hold in nuanced ways. We formalize the concept of this symmetry later in subsection 2.2.3, and show in subsection 2.2.22 that a distinction exists between *continuous* shift equivariance and *discrete* shift equivariance. They represent different strengths of the same symmetry and come with different requirements, which we develop in subsection 2.2.23. We argue in section 3.1 that continuous shift equivariance is very desirable for neural network architectures, and implement this throughout our method.

The analysis we present on this topic is common in the field of signal processing, but we have not seen it in the context of architecture design motivated by abstract algebraic concepts. The vast majority of prior work around shift equivariant neural networks tend to not consider the distinction between *continuous* shift equivariance and *discrete* shift equivariance, and only satisfy the conditions necessary for the latter (Ruthotto and Haber, 2020; Shen et al., 2020; Jenner and Weiler, 2021; Worrall et al., 2017; Weiler et al., 2018b,a; Esteves et al., 2018; Cohen and Welling, 2016; Weiler and Cesa, 2019; Xu et al., 2014; Marcos et al., 2018; Worrall and Welling, 2019; Sosnovik et al., 2019; Wang et al., 2019). This includes every publication we present on the topic of section 1.2, with one exception, which we review in the next paragraph. This body of work argues that pointwise nonlinearities have *discrete* shift equivariance because they have permutation invariance, which is true, but is insufficient to guarantee they have *continuous* shift equivariance. All classical convolutional neural network architectures (section 1.1) also effectively follow this pattern.

The outlier to this trend subscribes to the perspective of neural operators, which considers neural architectures that map from *continuous functions* to other *continuous functions*, which are discretized in some way that maintains the information present in the underlying functions. We return to this in subsection 1.2.4. Fanaskov and Oseledets (2022) notably implements continuous shift equivariance through phase invariant activation functions in the spectral domain. Karras et al. (2021) can also be conceptualized in this framework, as it implements continuous shift equivariance through correct antialiasing in the spatial domain, but it is instead motivated by a signal processing perspective.

1.2.2. Rotation and reflection equivariant neural networks

We discuss architectures that guarantee equivariance to rotations and reflections on signals, which is also commonly desirable, and which our method can implement through the steerable layers we introduce in section 3.2.

Some architectures implement rotation and reflection invariance instead by interpreting the signal as a vector field, or by observing its partial derivatives, and by then constructing a weighted sum of operators that are rotation and reflection invariant on these mathematical objects (Shen et al., 2020; Jenner and Weiler, 2021). The Laplacian or the divergence are common examples. This is also how our method can implement this equivariance through its steerable layers (section 3.2).

Some architectures implement rotation equivariance by decomposing convolutional kernels in terms of weighted sums of spherical harmonic kernels (Worrall et al., 2017; Weiler et al., 2018b,a; Esteves et al., 2018). Each spherical harmonic kernel corresponds to an oscillating wave that possesses both phase and frequency information. By designing the convolutional kernel such that the phase information of spherical harmonic kernels is lost, information about the absolute rotation of the signal is masked, rendering the whole operation equivariant to rotation.

Some architectures perform other decompositions of kernels to guarantee these invariances from a group theoretic perspective (Cohen and Welling, 2016; Weiler and Cesa, 2019).

1.2.3. Scale equivariant neural networks

We discuss architectures that guarantee equivariance to scale on signals, which is relevant to our method as the Laplacian residual blocks we formulate in section 3.6 lend themselves very well to the construction of scale equivariant networks.

The vast majority of methods that provide scale equivariance effectively evaluate scaled copies of a single network in parallel, and combine their results to form a single output that preserves scale equivariance for the specific copies chosen (Xu et al., 2014; Marcos et al., 2018; Worrall and Welling, 2019; Sosnovik et al., 2019; Wang et al., 2019). This is either done explicitly through the process just described, or implicitly by formulating a weight-sharing scheme that implements the same general idea.

1.2.4. Discretization invariant neural networks

We finally discuss architectures that enforce discretization invariance. Note that we provide an intuitive explanation of discretization invariance earlier in chapter , and define discretization invariance formally later in subsection 2.2.18. Our architecture implements full discretization invariance in all of its components. This is a central aspect of our work.

The most relevant line of work related to discretization invariance is that of neural operators (Kovachki et al., 2021), which study neural architectures that take *continuous functions* as inputs, and produce *continuous functions* as outputs. This is done through some form of representation that corresponds to discretization.

Architectures based on Fourier neural operators have been applied to image-based tasks, with some claiming full discretization invariance (Li et al., 2020; Kabri et al., 2023; Poli et al., 2022). This is not exactly true because of flawed assumptions on shift invariance outlined earlier in subsection 1.2.1, and analyzed in detail later in subsection 2.2.22 and subsection 2.2.23.

Architectures based on subtly different spectral neural operators have been applied to similar tasks, and in fact, possess full discretization invariance (Fanaskov and Oseledets, 2022). Their principle of operation was quickly discussed in subsection 1.2.1.

1.3. Initialization strategies for Deep Networks

A common trait of all neural architectures is that learning requires some choice of initial parameters. With deep neural networks, this choice of weight initialization becomes very important (LeCun et al., 2002; Glorot and Bengio, 2010; He et al., 2015; Mishkin and Matas, 2015). The learning process operates through gradients that propagate through each successive layer of the network, so the tendency of each layer to slightly increase or decrease the magnitude of gradients compounds exponentially over the many layers. This leads to the phenomenon of exploding and vanishing gradients, which collapses the learning process. Weight initialization strategies that control the magnitude of gradients at the start of the learning process are therefore necessary. We quickly review the two most common types.

1.3.1. Analytically driven

Weight initialization strategies are nearly always based on a statistical analysis of the gradient properties of weight initializations. They are crafted such that from one layer of the architecture to another, the variance of the gradients stays around one, rather than exploding and vanishing progressively LeCun et al. (2002); Glorot and Bengio (2010); He et al. (2015). Ideal weight initialization depends on the properties of each layer, so the exploration of novel layers generally requires some analytic derivations.

1.3.2. Data driven

Weight initialization strategies sometimes rely on learning methods instead of analytic derivations to find weight initializations that provide good statistical properties (Bengio et al., 2006; Krähenbühl et al., 2015; Mishkin and Matas, 2015). Instead of formulating analytic derivations by hand, a very simple learning program adjusts the weight initialization until it is satisfactory.

Chapter 2

Background

2.1. Deep Learning

Our work belongs to the field of deep learning, which broadly includes algorithms that leverage data to automatically learn solutions to tasks, and that base these solutions on deep neural network architectures (LeCun et al., 2015; Goodfellow et al., 2016). Each of these elements is tightly woven together to allow automatic learning.

Tasks define the general form that the solution must take and usually come with a sort of solution quality metric. Tasks also come with data that implicitly contains the desirable behaviour we wish to impart on the solution, and that allows evaluating the solution quality metric in a way that echoes this desirable behaviour.

Architectures parametrize solutions in a way that allows computing how each individual parameter can be incrementally changed to improve the quality metric. Architectures also introduce a form of bias to solutions by restricting the space of possible solutions to ones that conform to known, fundamental properties of the data.

In the two short subsections that follow, we provide an overview of typical task settings for deep learning, and we develop the notion of inductive biases in architecture design through the abstract algebraic concepts of *invariance* and *equivariance*.

2.1.1. Task settings

Tasks broadly refer to a problem setting for deep learning (Goodfellow et al., 2016). They define what a solution should do, the form of information it consumes, and the form of information it produces. The interest of machine learning lies in being able to find solutions to tasks that cannot be easily specified through formal or mathematical statements, but rather through examples. Tasks are specified by datasets, which provide example information to the solution that enables learning. Tasks can very broadly be partitioned into two categories in terms of the way they provide example information: *supervised learning* tasks, and *unsupervised learning* tasks.

Tasks that fit the supervised learning paradigm *explicitly* specify a distribution of pairings between input domain and output domain $X_{\text{in}} \times X_{\text{out}}$ which the solution is expected to learn. The dataset consists of pairings that allow direct evaluation of some quality metric on the solution, which enables learning. Typical examples of this type of task are classification and regression.

Tasks that fit the unsupervised learning paradigm *implicitly* specify a useful property or a desirable trait present in a distribution on an input domain X_{in} that the solution is expected to reproduce or extract. The dataset consists only of points in the input domain. The definition of the quality metric and its evaluation on the solution is much more open-ended. Typical examples of this type of task include distribution modeling and compression.

2.1.2. Architecture design, inductive biases, and abstract algebra

Architectures can be conceptualized as solutions to tasks that are implemented as functions $f_\theta : X_{\text{in}} \to X_{\text{out}}$, which map from some input domain X_{in} to some output domain X_{out}, whose behaviour is parameterized by θ (Goodfellow et al., 2016). Architecture design consists in establishing exactly how θ defines the behaviour of the function f_θ.

Deep learning architectures construct this function using a large set of interconnected neural layers, which each perform simple mathematical operations whose behaviour is controlled by some part of θ. Given sufficiently many simple neural layers consisting of linear interconnections and nonlinear functions, it is possible to approximate arbitrary functions to any degree of precision (Hornik et al., 1989).

While this extreme expressivity makes them very powerful, being able to formulate excessively complex solutions is often counterproductive to the learning process. Therefore, it is common to restrict the class of functions that can be expressed by an architecture to functions that echo the fundamental properties of the data. This is known as introducing *inductive biases.*

Abstract algebra can be useful in specifying these inductive biases. Abstract algebra studies how structures can be formed on mathematical objects, and how the properties of these structures are preserved or lost under certain operations. For instance, a vector space is a structure that involves mathematical objects that are vectors, and that possesses two operations, vector addition and scalar multiplication. These operations induce interesting properties that *are* the structure of vector spaces: associativity of vector addition, commutativity of vector addition, the existence of an identity and inverse element of vector addition, and so on. Abstract algebra studies how these properties can exist together as a structure, independently of the specific mathematical object they are implemented on. This is useful because the same structures are commonly shared between many distinct mathematical objects.

Abstract algebra also studies how certain mathematical objects possess *symmetries*, which have the property of making the object identical to the same object transformed by specific operations g. For example, a pattern that repeats itself endlessly is indistinguishable from the same pattern shifted by exactly one element; it has a symmetry that can be specified in terms of the shift operation.

Abstract algebra gives us precise language to express how other operations f respect the symmetry of a mathematical object. If f applies independently of the application of g, f is *invariant* under g (Equation 2.1.1). If f commutes with the application of g, f is *equivariant* under g (Equation 2.1.2).

$$f \circ g = f \qquad \iff \qquad f \text{ is invariant under } g \tag{2.1.1}$$

$$f \circ g = g \circ f \qquad \iff \qquad f \text{ is equivariant under } g \tag{2.1.2}$$

Coming back to a more pragmatic example, when formulating architectures for signals, it is useful to consider that the meaningful content of signals has a form of *shift symmetry*, since the absolute position of information a signal generally does not change its meaning. If the task we are trying to solve is animal classification on images, a cat is a cat, no matter where it is in the image. Crafting f_θ with an *inductive bias* that forces all solutions reachable by θ to be *shift invariant* or *shift equivariant* is very desirable, as it frees us from establishing by example that the countless possible translations of every image do not fundamentally alter the relevant properties of the image. This both facilitates the learning process and renders learnt solutions more general.

The implementation of inductive biases in deep neural architectures has been key to the success of deep learning. The typical taxonomy of neural architectures more or less corresponds to a categorization based on the inductive biases that each architecture implements. For instance, the most widely shared characteristic of convolutional neural networks is their shift symmetry.

More recently, the study of exotic symmetries and data types through the lens of abstract algebra has allowed the development of novel architectures that tackle challenging problems, and has also allowed a better interpretation of previously formulated architectures (Bronstein et al., 2017; Kondor and Trivedi, 2018; Cohen et al., 2019; Finzi et al., 2020; Lang and Weiler, 2020; Bronstein et al., 2021).

2.2. Signals

Our method heavily builds on the mathematics of functional analysis and signal processing. This subsection provides an introduction to signals for readers unfamiliar with their principles, and a discussion of the advanced topics that are required to correctly formalize our method. The intuition it builds only requires calculus, linear algebra, and knowledge of complex numbers, and should allow grasping the contributions presented in this thesis.

2.2.1. From finite to infinite linear algebra

The central intuition that underpins all theory of signals is that linear algebra allows change of basis not only for vectors of finite dimensionality, but for functions, which behave as vectors of infinite dimensionality. To illustrate this clearly, we outline the parallel between the change of basis in these two different regimes, which involves a short discussion of the inner product, and of orthonormal bases.

For a *real vector space of finite dimensionality* d, the inner product between vectors $\mathbf{v}$ and $\mathbf{v}'$ is defined as a *sum over an elementwise product*:

$$\langle \mathbf{v}, \mathbf{v}' \rangle \overset{\text{def}}{=} \sum_{j=1}^{d} \mathbf{v}_j \mathbf{v}'_j \tag{2.2.1}$$

In this vector space, a set of d vectors $\mathbf{b}_j$ forms an orthonormal basis when the following condition is satisfied:

$$\langle \mathbf{b}_j, \mathbf{b}_{j'} \rangle = \mathbb{1}\left\{ j = j' \right\} \tag{2.2.2}$$

Given an orthonormal basis, any vector $\mathbf{v}$ that belongs to the vector space can be represented as a sum of orthonormal bases $\mathbf{b}_j$ weighted by coefficients c_j. This is also true of orthogonal bases, however, orthonormal bases are normalized to allow recovering the coefficients c_j very conveniently using the inner product:

$$\mathbf{v} = \sum_{j=1}^{d} c_j \mathbf{b}_j \text{ where } c_j = \langle \mathbf{v}, \mathbf{b}_j \rangle \tag{2.2.3}$$

For an *imaginary vector space of infinite dimensionality*, or a vector space of functions, all of these operations have an analogous definition. Setting aside some details, the inner product between well-behaved functions or *signals* $s : \mathbb{R} \to \mathbb{C}$ and $s' : \mathbb{R} \to \mathbb{C}$ is defined as an *integral over a conjugate product*:

$$\langle s, s' \rangle = \int_{-\infty}^{\infty} s(x)\overline{s'(x)} dx \tag{2.2.4}$$

In this vector space, a finite or infinite set of functions b_j can form an orthonormal basis when the same condition is satisfied:

$$\langle b_j, b_{j'} \rangle = \mathbb{1}\left\{ j = j' \right\} \tag{2.2.5}$$

Given an orthonormal basis, any function s that belongs to the span of the basis can be represented as a sum of orthonormal bases b_j weighted by coefficients c_j. We will derive the two most relevant bases to signals in subsection 2.2.5 and subsection 2.2.15. Generally, the span and size of a basis will depend on the breadth of functions we wish to be able to represent with it. This is left out for now, but subsection 2.2.2 will address this in more detail. The change of basis for functions is conceptually identical to the change of basis for vectors:

$$s = \int c_j b_j dj \text{ where } c_j = \langle s, b_j \rangle \tag{2.2.6}$$

In effect, this means we can re-express functions as sums of other functions, which is a very powerful analytical tool in many situations.

By definition, the inner product (Equation 2.2.1, Equation 2.2.4) has commutativity (Equation 2.2.7), associativity (Equation 2.2.8), bilinearity (Equation 2.2.9), and distributivity (Equation 2.2.10):

$$\langle s, s' \rangle = \langle s', s \rangle \tag{2.2.7}$$

$$\langle s, \langle s', s'' \rangle \rangle = \langle \langle s, s' \rangle , s'' \rangle \tag{2.2.8}$$

$$a \langle s, s' \rangle = \langle as, s' \rangle \tag{2.2.9}$$

$$\langle s, s' + s'' \rangle = \langle s, s' \rangle + \langle s, s'' \rangle \tag{2.2.10}$$

By definition, the change of basis (Equation 2.2.3, Equation 2.2.6), has the same properties.

The functional analysis tools introduced above also generalize to higher dimensions, meaning the input of the decomposed functions can have multiple dimensions. The subsections that follow study signals in this multidimensional setting.

This brief introduction assumes that changes of basis are defined between functions whose $L2$ norm is finite. Functional analysis can be broadened to include generalized functions such as the Dirac delta (Fourier, 1888; Dirac, 1927; Schwartz, 1957, 1958; Strichartz, 2003), which we will heavily rely on later. We do not cover this in depth.

2.2.2. Fundamentals

With the foundational building blocks of functional analysis out of the way, we can define the basic vocabulary that underlies the discussion of signals: *signals* themselves, the *spatial domain*, the *spectral domain*, and *signal classes* (Fourier, 1888; Shannon, 1949; Petersen and Middleton, 1962).

We use *signal* to refer to any function s that is sufficiently well-behaved to be analyzed with the algebraic tools we have introduced in subsection 2.2.1 using some orthonormal basis.

The *spatial domain* of a signal hosts its representation in its original form. This is the starting point for our analysis. We later formalize the implicit *spatial basis* on which this relies in subsection 2.2.15. The *spatial domain* $\mathbf{X}$ is the domain of the *spatial coefficients* of the signal $s : \mathbf{X} \to \mathbb{C}$. Points in this domain are notated $\mathbf{x} \in \mathbf{X}$. Every spatial coefficient $s(\mathbf{x})$ is effectively a weight of a component of some *spatial basis* $b_{\mathbf{x}}$.

The *spectral domain* of a signal hosts its representation in an alternate form that is obtained through a change of basis. This is something we later build towards by explicitly formulating the *spectral basis* in subsection 2.2.5, and by deriving the equations for the change of basis in subsection 2.2.6, with the *Fourier transform* and the *inverse Fourier transform.* The *spectral domain* $\boldsymbol{\Omega}$ is the domain of the *spectral coefficients* of the signal $\widehat{s} : \boldsymbol{\Omega} \to \mathbb{C}$. Points in this domain are notated $\boldsymbol{\omega} \in \boldsymbol{\Omega}$. Every spectral coefficient $\widehat{s}(\boldsymbol{\omega})$ is effectively a weight of a component of some *spectral basis* $b_{\boldsymbol{\omega}}$.

Both domains allow us to define a *signal class* as a set of signals that are within the span of a spatial basis and spectral basis. This is essentially the set of all signals that can appropriately be manipulated by the functional analysis tools of subsection 2.2.1, given a specific subset of spatial basis functions and spectral basis functions. This is often also called a set of bandlimited functions in Shannon (1949); Petersen and Middleton (1962).

A common way to specify classes of signals is to let $\mathbf{X}$ and $\boldsymbol{\Omega}$ specify which spatial coefficients $s(\mathbf{x})$ and spectral coefficients $\widehat{s}(\boldsymbol{\omega})$ are allowed to be nonzero. Necessarily, signals that respect these conditions can disregard complementary spatial basis functions $\{b_{\mathbf{x}} | \mathbf{x} \notin \mathbf{X}\}$ and complementary spectral basis functions $\{b_{\boldsymbol{\omega}} | \boldsymbol{\omega} \notin \boldsymbol{\Omega}\}$ because they will always be associated with zero coefficients. A more refined way of specifying classes of signals considers the role of periodicity in both the spatial domain and spectral domain. We will come back to this in subsection 2.2.11 and in subsection 2.2.12.

Our discussion of signals will assume $\mathbf{X} = \mathbb{R}^d$ and $\boldsymbol{\Omega} = \mathbb{R}^d$ as a continuous starting point, which we will reduce to something discrete in subsection 2.2.12.

Limitations and purpose. Note that assuming $\mathbf{X} = \mathbb{R}^d$ and $\boldsymbol{\Omega} = \mathbb{R}^d$ without any other assumptions breaks our simple analysis tools in various ways. The inner product can become ill-defined, and pathological functions such as the Weierstrass function (Weierstrass, 1895) can be constructed. The many necessary conditions for the correct treatment of this setting are not explored in depth, as it would require a detour into the theory of distributions (Schwartz, 1957, 1958; Strichartz, 2003).

The purpose of this summary is not to propose a completely general theory of signals, but to discuss a perspective of *discretized signals* that allows their manipulation as *continuous signals.* This approach greatly simplifies the discussion of the *discretization of signals* (subsection 2.2.11, subsection 2.2.12, subsection 2.2.13), the *discretization of operators that act on signals* (subsection 2.2.18), and the formalization of *discretization invariance* (subsection 2.2.18), which are central to this work.

2.2.3. The shift operator

The *shift operator* allows us to express function translation with very compact notation (Shannon, 1949; Petersen and Middleton, 1962; Schwartz, 1957, 1958; Strichartz, 2003). It allows us to define exactly what we mean by shift invariance and shift equivariance, which are the fundamental symmetries of many important mathematical objects in the theory of signals, and which underlie all convolutional neural architectures.

We write the shift operator as $\mathfrak{T}_{\mathbf{x}'}$. Given a function s, it translates it by $\mathbf{x}'$:

$$\mathfrak{T}_{\mathbf{x}'}\{s\}(\mathbf{x}) \overset{\text{def}}{=} s(\mathbf{x} - \mathbf{x}') \tag{2.2.11}$$

We can apply the definitions of Equation 2.1.1 and Equation 2.1.2 to write down the conditions under which some function f is invariant under shift (Equation 2.2.12) or equivariant under shift (Equation 2.2.13):

$$f \circ \mathfrak{T}_{\mathbf{x}'} = f \qquad \forall \mathbf{x}' \in \mathbf{X}' \iff f \text{ is invariant under } \mathfrak{T}_{\mathbf{x}'} \qquad \forall \mathbf{x}' \in \mathbf{X}' \tag{2.2.12}$$

$$f \circ \mathfrak{T}_{\mathbf{x}'} = \mathfrak{T}_{\mathbf{x}'} \circ f \qquad \forall \mathbf{x}' \in \mathbf{X}' \iff f \text{ is equivariant under } \mathfrak{T}_{\mathbf{x}'} \qquad \forall \mathbf{x}' \in \mathbf{X}' \tag{2.2.13}$$

Here, we explicitly write down a set $\mathbf{X}'$ that specifies the set of shift operators $\{\mathfrak{T}_{\mathbf{x}'} | \mathbf{x}' \in \mathbf{X}'\}$ under which this symmetry holds. We will see in subsection 2.2.22 that the structure of this set can greatly influence the strength of this symmetry and the difficulty of satisfying its requirements; *discrete shift* and *continuous shift* are two different beasts.

2.2.4. The mirror operator

The *mirror operator* applies another very simple transformation on functions that is useful to in the theory of signals (Shannon, 1949; Petersen and Middleton, 1962; Schwartz, 1957, 1958; Strichartz, 2003). We will later see that both the *spectral basis* (subsection 2.2.5) and *spatial basis* (subsection 2.2.15) relate to this operator in a way that is fundamental to our discussion of discretization.

We write the mirror operator as $\mathfrak{M}$. given a function s, it simply performs a reflection along the origin:

$$\mathfrak{M}\{s\}(\mathbf{x}) \overset{\text{def}}{=} s(-\mathbf{x}) \tag{2.2.14}$$

2.2.5. The spectral basis

For us to define alternate representations of signals through the inner product (Equation 2.2.4) and the change of basis (Equation 2.2.6), we need an orthonormal basis. This subsection introduces the most commonly used basis in the theory of signals, the *spectral basis* $b_{\boldsymbol{\omega}}$, often called the complex exponential basis or Fourier basis (Fourier, 1888; Shannon, 1949; Petersen and Middleton, 1962).

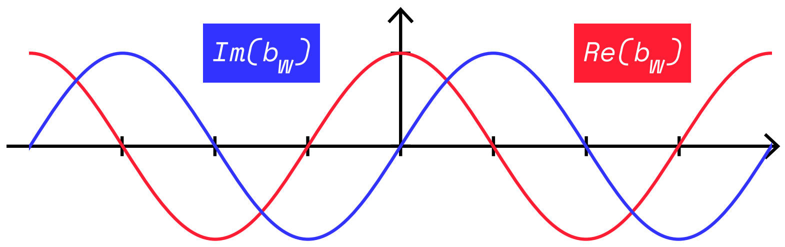


Fig. 2.1. Graph of the spectral basis function.

This shows the spectral basis function in a simple 1-dimensional case. The real part is a cosine, while the imaginary part is a sine.

We define the spectral basis function b_ω as a function that oscillates over the *spatial domain* (points $\mathbf{x} \in \mathbf{X}$) at a frequency specified over the *spectral domain* (points $\boldsymbol{\omega} \in \boldsymbol{\Omega}$). We construct it using the complex exponential, as it is very convenient for the representation of rotations and oscillations:

$$b_\omega(\mathbf{x}) \overset{\text{def}}{=} e^{i2\pi\langle\boldsymbol{\omega},\mathbf{x}\rangle} \tag{2.2.15}$$

Because of the symmetries of the trigonometric functions that compose the complex exponential, the mirrored (subsection 2.2.4, Equation 2.2.14) conjugate $\overline{\mathfrak{M}\{b_\omega\}}$ of a spectral basis is equal to the spectral basis b_ω itself:

$$\overline{\mathfrak{M}\{b_\omega\}}(\mathbf{x}) = \overline{\mathfrak{M}\{\cos(2\pi\langle\boldsymbol{\omega},\mathbf{x}\rangle) + i\sin(2\pi\langle\boldsymbol{\omega},\mathbf{x}\rangle)\}} \tag{2.2.16}$$
$$= \cos(-2\pi\langle\boldsymbol{\omega},\mathbf{x}\rangle) - i\sin(-2\pi\langle\boldsymbol{\omega},\mathbf{x}\rangle) \tag{2.2.17}$$
$$= \cos(2\pi\langle\boldsymbol{\omega},\mathbf{x}\rangle) + i\sin(2\pi\langle\boldsymbol{\omega},\mathbf{x}\rangle) \tag{2.2.18}$$
$$= b_\omega(\mathbf{x}) \tag{2.2.19}$$

This property holds for the *spectral basis*, but we will later see that it also holds for the *spatial basis* (subsection 2.2.15). This will be very important to the way we discuss discretization.

Because the spectral basis is built from complex exponentiation, the usual rules of exponents apply:

$$b_\omega(\mathbf{x})b_{\omega'}(\mathbf{x}) = e^{i2\pi\langle\boldsymbol{\omega},\mathbf{x}\rangle}e^{i2\pi\langle\boldsymbol{\omega}',\mathbf{x}\rangle} \tag{2.2.20}$$
$$= e^{i2\pi\langle\boldsymbol{\omega}+\boldsymbol{\omega}',\mathbf{x}\rangle} \tag{2.2.21}$$
$$= b_{\omega+\omega'}(\mathbf{x}) \tag{2.2.22}$$

$$b_\omega(\mathbf{x})b_\omega(\mathbf{x}') = e^{i2\pi\langle\boldsymbol{\omega},\mathbf{x}\rangle}e^{i2\pi\langle\boldsymbol{\omega},\mathbf{x}'\rangle} \tag{2.2.23}$$
$$= e^{i2\pi\langle\boldsymbol{\omega},\mathbf{x}+\mathbf{x}'\rangle} \tag{2.2.24}$$
$$= b_\omega(\mathbf{x}+\mathbf{x}') \tag{2.2.25}$$

The set of spectral bases $\{b_{\boldsymbol{\omega}} | \boldsymbol{\omega} \in \boldsymbol{\Omega}\}$ can be made orthonormal (Equation 2.2.5) for signal classes defined over certain spectral domains $\boldsymbol{\Omega}$ and spatial domains $\mathbf{X}$. This is always true when constructing spectral sampling sets $\boldsymbol{\Omega}_{\text{spl}}$ using the process outlined later in subsection 2.2.12.

2.2.6. The Fourier transform and inverse Fourier transform

In this subsection, we formulate a bridge between the original domain of the signal, the *spatial domain* $\mathbf{X}$, and a new *spectral domain* $\boldsymbol{\Omega}$, which derives from the spectral basis (subsection 2.2.5). This bridge can be traversed in either direction through the *Fourier transform* or the *inverse Fourier transform.*

These two operators are important in the theory of signals and in many other areas of mathematics because they can often provide analytic solutions to otherwise impenetrable theoretical problems (Fourier, 1888), or computationally efficient solutions to asymptotically hard problems (Shannon, 1949; Petersen and Middleton, 1962; Cooley and Tukey, 1965; Schönhage and Strassen, 1971).

The Fourier transform $\mathfrak{F} : (\mathbf{X} \to \mathbb{C}) \to (\boldsymbol{\Omega} \to \mathbb{C})$ performs a change of basis from the *spatial domain* to the *spectral domain.* This projects from *spatial coefficients* $s(\mathbf{x})$ to *spectral coefficients* $\widehat{s}(\boldsymbol{\omega})$. The Fourier transform is constructed by applying the inner product (Equation 2.2.4) using the spectral basis $b_{\boldsymbol{\omega}}$ (Equation 2.2.15):

$$\mathfrak{F}\left\{\ s\ \right\}(\boldsymbol{\omega}) = \widehat{s}(\boldsymbol{\omega}) \overset{\text{def}}{=} \langle s, b_{\boldsymbol{\omega}} \rangle = \int_{\mathbb{R}^d} s(\mathbf{x}) e^{-i2\pi\langle \boldsymbol{\omega}, \mathbf{x}\rangle} d\mathbf{x} \tag{2.2.26}$$

The inverse Fourier transform $\mathfrak{F}^{-1} : (\boldsymbol{\Omega} \to \mathbb{C}) \to (\mathbf{X} \to \mathbb{C})$ performs a change of basis from the *spectral domain* to the *spatial domain.* This projects from *spectral coefficients* $\widehat{s}(\boldsymbol{\omega})$ to *spatial coefficients* $s(\mathbf{x})$. The inverse Fourier transform is conversely constructed by applying the change of basis (Equation 2.2.6) using the spectral basis $b_{\boldsymbol{\omega}}$ (Equation 2.2.15):

$$\mathfrak{F}^{-1}\left\{\ \widehat{s}\ \right\}(\mathbf{x}) = s(\mathbf{x}) \overset{\text{def}}{=} \int_{\mathbb{R}^d} \widehat{s}(\boldsymbol{\omega})\ b_{\boldsymbol{\omega}}(\mathbf{x}) d\boldsymbol{\omega} = \int_{\mathbb{R}^d} \widehat{s}(\boldsymbol{\omega}) e^{i2\pi\langle \boldsymbol{\omega}, \mathbf{x}\rangle} d\boldsymbol{\omega} \tag{2.2.27}$$

By definition, the Fourier transform performs an inner product (subsection 2.2.1), and therefore inherits associativity with scalar multiplication (Equation 2.2.28) and distributivity (Equation 2.2.29):

$$a\mathfrak{F}\{s\} = \mathfrak{F}\{as\} \tag{2.2.28}$$

$$\mathfrak{F}\{s + s'\} = \mathfrak{F}\{s\} + \mathfrak{F}\{s'\} \tag{2.2.29}$$

By definition, the inverse Fourier transform performs a change of basis (subsection 2.2.1), and therefore also inherits associativity with scalar multiplication (Equation 2.2.30) and distributivity (Equation 2.2.31):

$$a\mathfrak{F}^{-1}\{\widehat{s}\} = \mathfrak{F}^{-1}\{a\widehat{s}\} \tag{2.2.30}$$

$$\mathfrak{F}^{-1}\{\widehat{s} + \widehat{s}'\} = \mathfrak{F}^{-1}\{\widehat{s}\} + \mathfrak{F}^{-1}\{\widehat{s}'\} \tag{2.2.31}$$

2.2.7. The convolution operator

The *convolution operator* is central to the theory of signals, highly important to various other areas of mathematics, and specifically relevant to computer vision and convolutional neural architectures (Fourier, 1888; Shannon, 1949; Petersen and Middleton, 1962; Fukushima, 1980; He et al., 2016a,b; Krizhevsky et al., 2017). We present the convolution operator in this subsection and highlight its key properties.

The convolution operator $*$ takes two signals s and s', and combines them to form a new signal as defined below.

$$\left(s \ * \ s'\right)(\mathbf{x}) \overset{\text{def}}{=} \int_{\mathbb{R}^d} s(\mathbf{x}-\mathbf{x}')s'(\mathbf{x}')d\mathbf{x}' \tag{2.2.32}$$

$$= \int_{\mathbb{R}^d} s(\mathbf{x}')s'(\mathbf{x}-\mathbf{x}')d\mathbf{x}' \tag{2.2.33}$$

Equation 2.2.33 follows from a change of variables, and implies convolution is commutative:

$$\left(s \ * \ s'\right) = \left(s' \ * \ s\right). \tag{2.2.34}$$

Using the shift operator (Equation 2.2.11) and the mirror operator (Equation 2.2.14), we can rewrite convolution as an inner product, which leads to an interpretation based on the intuition of linear algebra:

$$\left(s \ * \ s'\right)(\mathbf{x}) = \int_{\mathbb{R}^d} s(\mathbf{x}')s'(\mathbf{x}-\mathbf{x}')d\mathbf{x}' \tag{2.2.35}$$

$$= \int_{\mathbb{R}^d} s(\mathbf{x}')\mathfrak{M}\{s'\}(\mathbf{x}'-\mathbf{x})d\mathbf{x}' \tag{2.2.36}$$

$$= \int_{\mathbb{R}^d} s(\mathbf{x}')\mathfrak{T}_{\mathbf{x}}\{\mathfrak{M}\{s'\}\}(\mathbf{x}')d\mathbf{x}' \tag{2.2.37}$$

$$= \int_{\mathbb{R}^d} s(\mathbf{x}')\mathfrak{T}_{\mathbf{x}}\{\overline{\overline{\mathfrak{M}\{s'\}}}\}(\mathbf{x}')d\mathbf{x}' \tag{2.2.38}$$

$$= \left\langle s, \mathfrak{T}_{\mathbf{x}}\{\overline{\mathfrak{M}\{s'\}}\}\right\rangle \tag{2.2.39}$$

Convolution effectively creates a continuum of inner products between s and $\overline{\mathfrak{M}\{s'\}}$ that is navigated by $\mathbf{x}$. Each slice $\mathbf{x}$ corresponds to applying a shift $\mathfrak{T}_{\mathbf{x}}$ to $\overline{\mathfrak{M}\{s'\}}$ before taking the inner product with s. This is given a visual intuition in Figure 2.2. The result of Equation 2.2.39 is also important because it allows conceptualizing inner products as convolutions. This is central to our discussion of discretization.

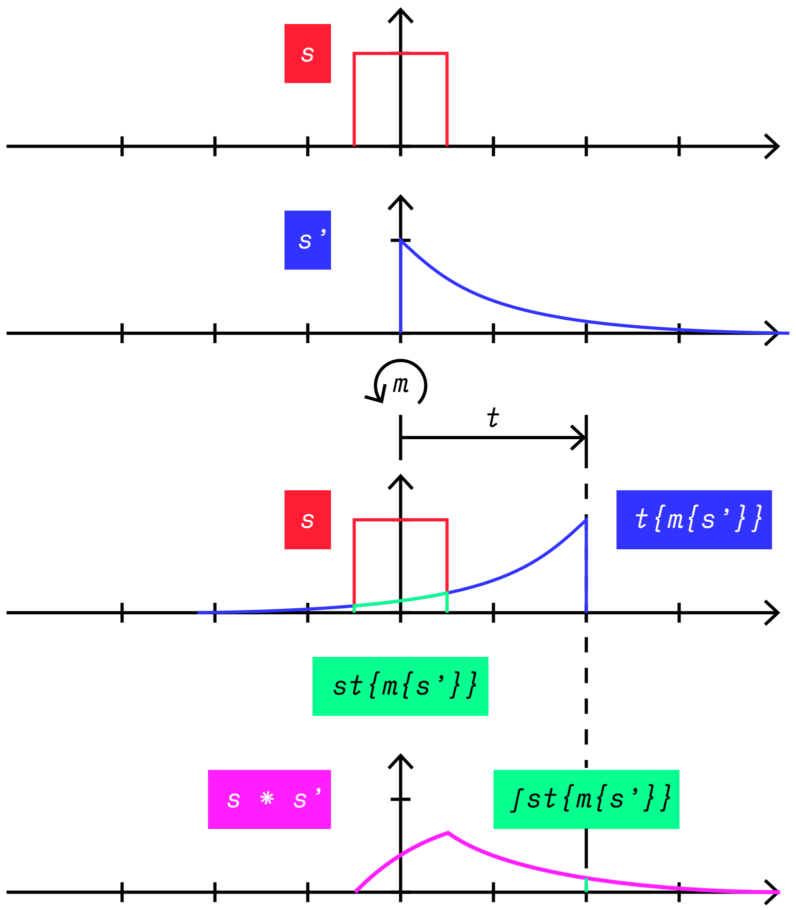


Fig. 2.2. Graph showing convolution as a shifted mirrored inner product.

This shows a convolution between two simple strictly real functions decomposed into the steps of Equation 2.2.39. Here, s is a simple rectangle pulse in red, and s' is a truncated exponential in blue. We take s', mirror and shift it to obtain $\mathfrak{T}_{\mathbf{x}}\{\overline{\mathfrak{M}\{s'\}}\}$, and compute its product with s, shown in green. The area under the curve of this product is equal to the value of the convolution at the corresponding shift $(s \ast s')(\mathbf{x})$. This is effectively the inner product $\left\langle s, \mathfrak{T}_{\mathbf{x}}\{\overline{\mathfrak{M}\{s'\}}\}\right\rangle$.

If we fix $\mathbf{x}$ in Equation 2.2.39, the properties of the inner product seen in subsection 2.2.1 suggest commutativity (Equation 2.2.40 as already shown in Equation 2.2.34), associativity (Equation 2.2.41), bilinearity (Equation 2.2.42), and distributivity (Equation 2.2.43). We can verify these properties do hold true as we change $\mathbf{x}$ by manipulating the integral of the convolution, but we spare the details here:

$$s * s' = s' * s \tag{2.2.40}$$

$$s * (s' \; * \; s'') = (s \; * \; s') * s'' \tag{2.2.41}$$

$$a\,(s \; * \; s') = (as) * s' \tag{2.2.42}$$

$$s * (s' + s'') = (s \; * \; s') + (s \; * \; s'') \tag{2.2.43}$$

Equation 2.2.39 implies convolution is shift equivariant. This property is crucial. Together with linearity, it is the defining characteristic of convolution:

$$\begin{aligned}(\mathfrak{T}_{\mathbf{x}'}\{s\} \; * \; s')\,(\mathbf{x}) &= \left\langle \mathfrak{T}_{\mathbf{x}'}\{s\}, \mathfrak{T}_{\mathbf{x}}\{\overline{\mathfrak{M}\{s'\}}\}\right\rangle & (2.2.44)\\ &= \left\langle s, \mathfrak{T}_{\mathbf{x}-\mathbf{x}'}\{\overline{\mathfrak{M}\{s'\}}\}\right\rangle & (2.2.45)\\ &= \mathfrak{T}_{\mathbf{x}'}\{s * s'\}(\mathbf{x}) & (2.2.46)\end{aligned}$$

We now introduce the *convolution theorem*, which relates the application of convolution in one domain to the application of multiplication in the other domain:

$$\mathfrak{F}\left\{s * s'\right\} = \mathfrak{F}\left\{s\right\}\mathfrak{F}\left\{s'\right\} \tag{2.2.47}$$

$$\mathfrak{F}\left\{s\right\} * \mathfrak{F}\left\{s'\right\} = \mathfrak{F}\left\{ss'\right\} \tag{2.2.48}$$

This is especially useful as it allows transforming many computationally hard problems into much simpler equivalent ones (Cooley and Tukey, 1965; Schönhage and Strassen, 1971).

The convolution theorem (Equation 2.2.47)also implies that the application of the convolution operator is closed under signal classes that restrict the spectral domain:

$$\mathfrak{F}\{s\}(\boldsymbol{\omega}) = 0 \forall \boldsymbol{\omega} \notin \boldsymbol{\Omega} \tag{2.2.49}$$

$$\mathfrak{F}\{s'\}(\boldsymbol{\omega}) = 0 \forall \boldsymbol{\omega} \notin \boldsymbol{\Omega}' \tag{2.2.50}$$

$$\implies \mathfrak{F}\{s * s'\}(\boldsymbol{\omega}) = 0 \forall \boldsymbol{\omega} \notin \boldsymbol{\Omega} \cap \boldsymbol{\Omega}' \tag{2.2.51}$$

The convolution theorem (Equation 2.2.47) provides another intuition for what convolution performs. When two signals are convolved together in the spatial domain, they are multiplied together in the spectral domain, meaning that the spectrum of one signal selectively masks the spectrum of the other.

The idea of using convolution as a masking operation is ubiquitous in signal processing. It will reemerge when we start investigating discretization.

We show pragmatic examples of convolutions applied to images below in Figure 2.3, Figure 2.4, Figure 2.5, Figure 2.6, and Figure 2.7.

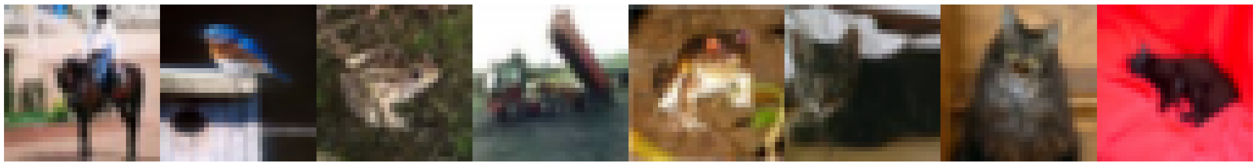


Fig. 2.3. Images before filtering, taken from the CIFAR10 dataset.

Because filtering positive valued signals can produce negative values, all images are normalized to fit the RGB colour range. In the case of the highpass and bandpass filtered images, grey should be interpreted as 0. Note that the visualizations are produced using Gaussian kernels, which do not act like perfectly sharp binary masks.

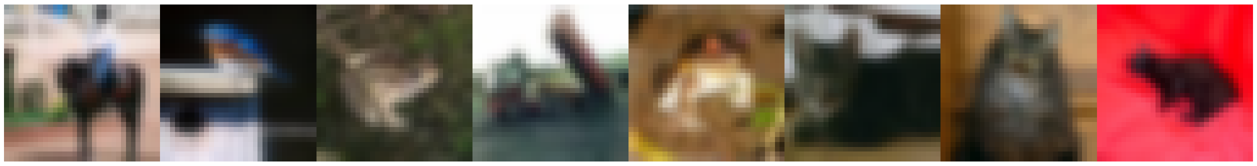


Fig. 2.4. Images under lowpass filtering.

Lowpass filters ϕ^{low} allow low frequencies but block high frequencies:

$$\widehat{\phi^{\text{low}}}(\boldsymbol{\omega}) \approx \mathbb{1}\left\{\boldsymbol{\omega} > \boldsymbol{\omega}'\right\} \tag{2.2.52}$$

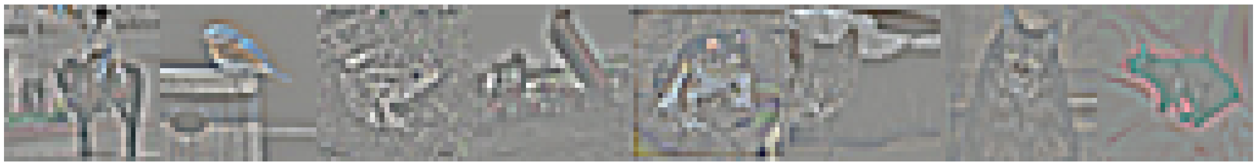


Fig. 2.5. Images under highpass filtering.

Highpass filters ϕ^{high} allow high frequencies but block low frequencies:

$$\widehat{\phi^{\text{high}}}(\boldsymbol{\omega}) \approx \mathbb{1}\left\{\boldsymbol{\omega} < \boldsymbol{\omega}'\right\} \tag{2.2.53}$$

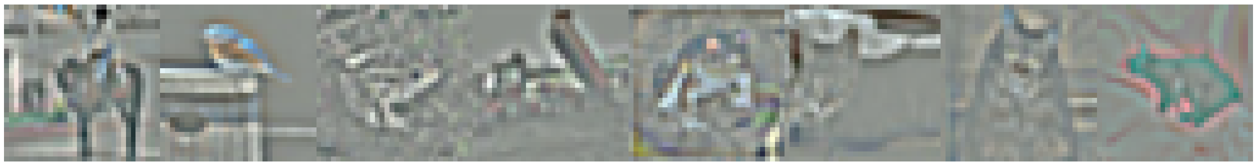


Fig. 2.6. Images under bandpass filtering.

Bandpass filters ϕ^{band} only allow a narrow band of frequencies $\boldsymbol{\Omega}'$, and reject the others:

$$\widehat{\phi^{\text{band}}}(\boldsymbol{\omega}) \approx \mathbb{1}\left\{\boldsymbol{\omega} \in \boldsymbol{\Omega}'\right\} \tag{2.2.54}$$

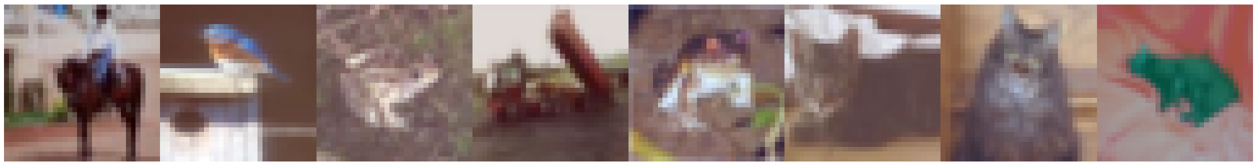


Fig. 2.7. Images under zero blocking filtering.

Zero blocking filters ϕ^{zero} remove the zero frequency component of the signal, leaving the other frequencies intact. This has the effect of subtracting the mean from the signal:

$$\widehat{\phi^{\text{zero}}}(\boldsymbol{\omega}) \approx \mathbb{1}\left\{\boldsymbol{\omega} \neq 0\right\} \tag{2.2.55}$$

2.2.8. The partial derivative operator

The partial derivative operator is often associated with the study of partial differential equations, but it is also very relevant to the study of signals, and to applications in computer vision (Sobel, 1968; Canny, 1986). We introduce it here and show its properties of interest.

We use multi-index notation $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ to designate higher-order multidimensional partial derivatives, where $\partial^\alpha = \partial^{\alpha_1} \ldots \partial^{\alpha_d}$. We shorten the notation to an operator to facilitate readability:

$$\mathfrak{D}_{\mathbf{x}}^{\alpha}\{s\} \overset{\text{def}}{=} \frac{\partial^\alpha s}{\partial \mathbf{x}^\alpha} = \frac{\partial^{\alpha_1}}{\partial \mathbf{x}_1^{\alpha_1}}\left\{ \cdots \frac{\partial^{\alpha_d}}{\partial \mathbf{x}_d^{\alpha_d}}\left\{ s \right\}\right\} \tag{2.2.56}$$

Just like convolution, given sufficiently many times differentiable functions, the partial derivative operator is commutative (Equation 2.2.57), bilinear (Equation 2.2.58), and distributive (Equation 2.2.59):

$$\mathfrak{D}_{\mathbf{x}}^{\alpha}\{\mathfrak{D}_{\mathbf{x}}^{\alpha'}\{s\}\} = \mathfrak{D}_{\mathbf{x}}^{\alpha'}\{\mathfrak{D}_{\mathbf{x}}^{\alpha}\{s\}\} \tag{2.2.57}$$

$$a\mathfrak{D}_{\mathbf{x}}^{\alpha}\{s\} = \mathfrak{D}_{\mathbf{x}}^{\alpha}\{as\} \tag{2.2.58}$$

$$\mathfrak{D}_{\mathbf{x}}^{\alpha}\{s + s'\} = \mathfrak{D}_{\mathbf{x}}^{\alpha}\{s\} + \mathfrak{D}_{\mathbf{x}}^{\alpha}\{s'\} \tag{2.2.59}$$

Just like convolution, again, the partial derivative operator is shift equivariant:

$$\mathfrak{T}_{\mathbf{x}'}\{\mathfrak{D}_{\mathbf{x}}^{\alpha}\{s\}\} = \mathfrak{D}_{\mathbf{x}}^{\alpha}\{\mathfrak{T}_{\mathbf{x}'}\{s\}\} \tag{2.2.60}$$

Multi-indices are summed over the composition of the partial derivative operator, as given by the chain rule:

$$\mathfrak{D}_{\mathbf{x}}^{\alpha}\{\mathfrak{D}_{\mathbf{x}}^{\alpha'}\{s\}\} = \mathfrak{D}_{\mathbf{x}}^{\alpha+\alpha'}\{s\} \tag{2.2.61}$$

This implies we can construct any higher-order partial derivative operator using first-order partial derivative operators that span all d dimensions.

As with convolution, we can show the partial derivative operator is in some sense closed on certain classes of signals. Applying any first-order partial derivative operator $\mathfrak{D}_{\mathbf{x}}^{\alpha}$ on any spectral basis $b_{\boldsymbol{\omega}}$ with $\boldsymbol{\omega} \in \boldsymbol{\Omega}$ yields a spectral basis proportional to $ib_{\alpha\langle\boldsymbol{\omega},\alpha\rangle}$, which is effectively projected over the direction of α and phase shifted:

$$\mathfrak{D}_{\mathbf{x}}^{\alpha}\{b_{\boldsymbol{\omega}}\} = i2\pi \left\langle \boldsymbol{\omega}, \alpha \right\rangle b_{\alpha\langle\boldsymbol{\omega},\alpha\rangle} \text{ assuming } \alpha \in \{0,1\}^d \text{ and } \sum_j^d \alpha_j = 1 \tag{2.2.62}$$

We can leverage the linearity of the Fourier transform (subsection 2.2.6) to construct the first-order partial derivatives of signals through this property.

We can observe that if a signal class is specified with a spectral domain $\boldsymbol{\Omega}$ such that $\alpha \left\langle \boldsymbol{\omega}, \alpha \right\rangle \in \boldsymbol{\Omega}\ \forall \boldsymbol{\omega} \in \boldsymbol{\Omega}$ for any α designating a first-order partial derivative, the signal class is closed under any first-order partial derivative operator. We can generalize this to any higher-order partial derivative operator through Equation 2.2.61 by repeated application of first-order partial derivative operators. This will be useful when approximately discretizing higher-order partial derivative operators in subsection 2.2.21.

We note that the argument above requires some technical caveats that disallow certain pathological functions, such as the Weierstrass function (Weierstrass, 1895). This is never encountered when discussing discretized signals, however.

2.2.9. The sampling operator

The foundation we have introduced for the analysis of signals only considers signals whose representations span an infinite spatial domain $\mathbf{X}$ and an infinite spectral domain $\boldsymbol{\Omega}$. This does not allow computation without some form of discretization, which reduces both of these domains to a finite size. The *sampling operator* can help us formalize this notion.

We first review the notion of the *Dirac delta* (Fourier, 1888; Dirac, 1927; Schwartz, 1957, 1958; Strichartz, 2003), which underlies the way we express discretization in this work. The Dirac delta is a generalized function that can informally be defined as being zero everywhere except at the origin, and as integrating to one:

$$\delta(\mathbf{x}) \overset{\text{def}}{=} \begin{cases} +\infty & \text{if } \mathbf{x} = 0 \\ 0 & \text{if } \mathbf{x} \neq 0 \end{cases} \text{ such that } \int_{\mathbb{R}^d} \delta(\mathbf{x}) d\mathbf{x} = 1 \tag{2.2.63}$$

Evaluating the Dirac delta directly at its origin does not provide us with meaningful information. *We must integrate over it.*

The *sampling operator* (Schwartz, 1957, 1958; Strichartz, 2003; Shannon, 1949; Petersen and Middleton, 1962; Landau, 1967) takes a signal s, and returns a sampled signal $\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}$, which only allows through the values of the signal at points $\mathbf{x}_{\text{spl}}$ of the sampling set $\mathbf{X}_{\text{spl}}$. The value of the signal at those points is let through by taking a product with a set of shifted (Equation 2.2.11) Dirac delta functions δ:

$$\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}(\mathbf{x}) \overset{\text{def}}{=} \sum_{\mathbf{x}_{\text{spl}} \in \mathbf{X}_{\text{spl}}} \mathfrak{T}_{\mathbf{x}_{\text{spl}}}\{\delta\}(\mathbf{x}) s(\mathbf{x}) \tag{2.2.64}$$

Evaluating the sampling operator directly at points of the sampling set is not meaningful, because it amounts to evaluating a Dirac delta directly. *We must integrate over it.*

This definition is not equivalent to simply transforming the continuous signal into a discretized signal that we can index into, like in Shannon (1949); Petersen and Middleton (1962). The representation it provides is useful when we wish to adapt *operators on continuous signals* to *operators on discretized signals*, specifically when these operators work through integration, specifically when we wish to remain in the continuous setting to perform analysis. This is the approach we take to discretization.

2.2.10. The repeating operator

The sampling operator shares a very special relationship with the *repeating operator*, which we will formalize in subsection 2.2.11. But first, we introduce the operator in question.

The *repeating operator* (Schwartz, 1957, 1958; Strichartz, 2003; Shannon, 1949; Petersen and Middleton, 1962; Landau, 1967) takes a signal s, and returns a repeated signal $\mathfrak{R}_{\mathbf{X}_{\text{rep}}}\{s\}$, which sums a set of copies of the signal each offset by some vector $\mathbf{x}_{\text{rep}}$ if the repeating set $\mathbf{X}_{\text{rep}}$:

$$\mathfrak{R}_{\mathbf{X}_{\text{rep}}}\{s\}(\mathbf{x}) \overset{\text{def}}{=} \sum_{\mathbf{x}_{\text{rep}} \in \mathbf{X}_{\text{rep}}} \mathfrak{T}_{\mathbf{x}_{\text{rep}}}\{s\}(\mathbf{x}) \tag{2.2.65}$$

2.2.11. The duality of sampling and repeating

The sampling operator and the repeating operator interact with the Fourier transform in a dual way that is fundamental to the theory of signals (Schwartz, 1957, 1958; Shannon, 1949; Petersen and Middleton, 1962; Strichartz, 2003). We later use this duality to formulate a framework that allows discretizing the Fourier transform in a way that treats *discrete signals* as *continuous signals*, which greatly simplifies our analysis (subsection 2.2.12). We then see that this duality directly leads to the *sampling theorem* (subsection 2.2.13), and provides a simple interpretation for *aliasing* (subsection 2.2.14).

Under some conditions on the *spatial repeating lattice* $\mathbf{\Lambda}_{\text{rep}}$ and *spectral sampling lattice* $\mathbf{\Gamma}_{\text{spl}}$, repeating in space then applying the Fourier transform is identical to applying the Fourier transform then sampling in spectrum:

$$\mathfrak{F}\left\{\mathfrak{R}_{\mathbf{\Lambda}_{\text{rep}}}\{s\}\right\}(\boldsymbol{\omega}) = \int_{\mathbb{R}^d} s(\mathbf{x})\mathfrak{R}_{\mathbf{\Lambda}_{\text{rep}}}\left\{e^{-i2\pi\langle\boldsymbol{\omega},\ \cdot\ \rangle}\right\}(\mathbf{x})d\mathbf{x} \tag{2.2.66}$$

$$= \int_{\mathbb{R}^d} s(\mathbf{x})\mathfrak{S}_{\mathbf{\Gamma}_{\text{spl}}}\left\{e^{-i2\pi\langle\ \cdot\ ,\mathbf{x}\rangle}\right\}(\boldsymbol{\omega})d\mathbf{x} \tag{2.2.67}$$

$$= \mathfrak{S}_{\mathbf{\Gamma}_{\text{spl}}}\left\{\mathfrak{F}\{s\}\right\}(\boldsymbol{\omega}) \tag{2.2.68}$$

The conditions on the lattices are dictated by the equality of Equation 2.2.66 to Equation 2.2.67, which is satisfied if and only if they are reciprocal:

$$\boldsymbol{\Lambda}_{\text{rep}} \overset{\text{def}}{=} \left\{ \sum_{j=1}^{d} \boldsymbol{\lambda}_j \mathbf{i}_j \middle| \mathbf{i} \in \mathbb{Z}^d \right\}$$
$$\text{where } \{\boldsymbol{\lambda}_j | j \in [1,d]\} \text{ is some basis of } \mathbb{R}^d \tag{2.2.69}$$

$$\boldsymbol{\Gamma}_{\text{spl}} \overset{\text{def}}{=} \left\{ \sum_{j=1}^{d} \boldsymbol{\gamma}_j \mathbf{i}_j \middle| \mathbf{i} \in \mathbb{Z}^d \right\}$$
$$\text{where } \left\{\boldsymbol{\gamma}_j \middle| j \in [1,d]\right\} \text{ is chosen such that } \left\langle \boldsymbol{\gamma}_j, \boldsymbol{\lambda}_{j'} \right\rangle = \mathbb{1}\left\{j = j'\right\} \tag{2.2.70}$$

Note that the basis $\{\boldsymbol{\lambda}_j | j \in [1,d]\}$ is not required to be orthonormal, neither orthogonal. The lattices are thus not required to be cubic.

The fact that repeating the spatial domain samples the spectral domain is reasonable if we consider which spectral coefficients can be nonzero for such a signal. Because the signal is made spatially periodic, we expect spectral coefficients to integrate to zero if the spectral basis is not periodic over the same period, and to otherwise concentrate its mass in a single point if the spectral basis is periodic over the same period. This is exactly what is implied by spectral sampling, and by the fact that its lattice is reciprocal.

The relationship we have just outlined holds identically with permuted domains. In what follows, we provide a way of relating the lattices that unites both relationships.

Under some conditions on the *spatial sampling lattice* $\boldsymbol{\Lambda}_{\text{spl}}$ and *spectral repeating lattice* $\boldsymbol{\Gamma}_{\text{rep}}$, sampling in space then applying the Fourier transform is identical to applying the Fourier transform then repeating in spectrum:

$$\mathfrak{F}\left\{ \mathfrak{S}_{\boldsymbol{\Lambda}_{\text{spl}}}\{s\} \right\}(\boldsymbol{\omega}) = \int_{\mathbb{R}^d} s(\mathbf{x}) \mathfrak{S}_{\boldsymbol{\Lambda}_{\text{spl}}}\left\{ e^{-i2\pi\langle \boldsymbol{\omega},\ \cdot\ \rangle} \right\}(\mathbf{x}) d\mathbf{x} \tag{2.2.71}$$

$$= \int_{\mathbb{R}^d} s(\mathbf{x}) \mathfrak{R}_{\boldsymbol{\Gamma}_{\text{rep}}}\left\{ e^{-i2\pi\langle\ \cdot\ , \mathbf{x}\rangle} \right\}(\boldsymbol{\omega}) d\mathbf{x} \tag{2.2.72}$$

$$= \mathfrak{R}_{\boldsymbol{\Gamma}_{\text{rep}}}\left\{ \mathfrak{F}\{s\} \right\}(\boldsymbol{\omega}) \tag{2.2.73}$$

The conditions on the lattices are dictated by the equality of Equation 2.2.71 to Equation 2.2.72, which is satisfied if and only if the two lattices are reciprocal.

We generally define these two lattices in relationship to the lattices we have defined earlier. If we wish to apply both the sampling operator and the repeating operator in one domain, then it is natural to let one repeating increment be equal to a finite number of sampling increments. We parameterize this relationship with an integer vector $\mathbf{n} \in \mathbb{N}^d$:

$$\boldsymbol{\Lambda}_{\text{spl}} \overset{\text{def}}{=} \left\{ \sum_{j=1}^{d} \boldsymbol{\lambda}_j \mathbf{n}_j^{-1} \mathbf{i}_j \middle| \mathbf{i} \in \mathbb{Z}^d \right\} \tag{2.2.74}$$

$$\boldsymbol{\Gamma}_{\text{rep}} \overset{\text{def}}{=} \left\{ \sum_{j=1}^{d} \boldsymbol{\gamma}_j \mathbf{n}_j \mathbf{i}_j \middle| \mathbf{i} \in \mathbb{Z}^d \right\} \tag{2.2.75}$$

In the equations above, the inverse applies componentwise. Both expressions can be reduced to a simple interpretation.

In Equation 2.2.74, every component of $\mathbf{n}^{-1}$ expresses the number of *spatial sampling* increments that sum to one *spatial repeating* increment along some axis of the lattice.

In Equation 2.2.75, every component of $\mathbf{n}$ expresses the number of *spectral sampling* increments that sum to one *spectral repeating* increment along some axis of the lattice.

The fact that sampling the spatial domain repeats the spectral domain is the converse of the intuition we developed for the opposite case. The periodic nature of spectral bases is essentially identical for conjugated spectral bases and a change of variables, and the inverse Fourier transform can be thought of as a Fourier transform with conjugated spectral bases and a change of variables. The intuition should also hold in the opposite direction.

2.2.12. Discretization

The results of subsection 2.2.11 provide a useful intuition for the discretization of signals in both the spatial domain $\mathbf{X}$ and the spectral domain $\boldsymbol{\Omega}$. If we assume we can apply our analysis tools to the signal class with unrestricted domains $\mathbf{X} = \mathbb{R}^d$ and $\boldsymbol{\Omega} = \mathbb{R}^d$ (which is not really correct by Schwartz (1957, 1958); Strichartz (2003), but is reasonable for the sake of the intuition it provides), we can add the assumption of periodicity in one domain to reduce the nonzero parts of the other domain. This allows deriving variants of our signal analysis tools that can discretize one or both domains.

Discretizing *only the spatial domain* yields the *discrete time Fourier transform* (Shannon, 1949; Petersen and Middleton, 1962):

- If we apply periodicity in the spectral domain, we can assume any spectral coefficients $\widehat{s}$ can be reexpressed as $\mathfrak{R}_{\boldsymbol{\Gamma}_{\text{rep}}}\{\widehat{s'}\}$, and necessarily, the spatial coefficients s are then sampled as $\mathfrak{S}_{\boldsymbol{\Lambda}_{\text{spl}}}\{s'\}$.
- The set of spatial bases $\{b_{\mathbf{x}} | \mathbf{x} \in \boldsymbol{\Lambda}_{\text{spl}}\}$ that correspond to this sampling are always orthonormal.
- If we narrow the spatial domain $\mathbf{X}$ to a bounded region, then we can define a sampled spatial domain $\mathbf{X}_{\text{spl}} = \mathbf{X} \cap \boldsymbol{\Lambda}_{\text{spl}}$ which has a finite number of samples.
  The subset of spatial bases $\{b_{\mathbf{x}} | \mathbf{x} \in \mathbf{X}_{\text{spl}}\}$ is also always orthonormal.

Discretizing *only the spectral domain* yields the *Fourier series* (Fourier, 1888):

- If we apply periodicity in the spatial domain, we can assume any spatial coefficients s can be reexpressed as $\mathfrak{R}_{\mathbf{\Lambda}_{\text{rep}}}\{s'\}$, and necessarily, the spectral coefficients $\widehat{s}$ are then sampled as $\mathfrak{S}_{\mathbf{\Gamma}_{\text{spl}}}\{\widehat{s'}\}$.
- The set of spectral bases $\{b_{\boldsymbol{\omega}} | \boldsymbol{\omega} \in \mathbf{\Gamma}_{\text{spl}}\}$ that correspond to this sampling are always orthonormal.
- If we narrow the spectral domain $\mathbf{\Omega}$ to a bounded region, then we can define a sampled spectral domain $\mathbf{\Omega}_{\text{spl}} = \mathbf{\Omega} \cap \mathbf{\Gamma}_{\text{spl}}$ which has a finite number of samples.
  The subset of spectral bases $\{b_{\boldsymbol{\omega}} | \boldsymbol{\omega} \in \mathbf{\Omega}_{\text{spl}}\}$ is also always orthonormal.

Discretizing *both domains* yields the *discrete Fourier transform* (Shannon, 1949; Petersen and Middleton, 1962). This is the setting we are most interested in. Then, the bounded regions of $\mathbf{X}$ and $\mathbf{\Omega}$ must be contained by some unit cell of the corresponding repeating lattices $\mathbf{\Lambda}_{\text{rep}}$ and $\mathbf{\Gamma}_{\text{rep}}$.

Whenever we wish to evaluate a convolution between a kernel ϕ and a signal s made periodic, or assumed to be periodic, the repeating operator can be moved over to kernel in the convolution, which makes computation feasible in many cases:

$$s * \phi = \mathfrak{R}_{\mathbf{\Gamma}_{\text{rep}}}\{s'\} * \phi = s' * \mathfrak{R}_{\mathbf{\Gamma}_{\text{rep}}}\{\phi\} \tag{2.2.76}$$

In principle, this would be used whenever we apply convolutions on discretized signals in the next sections. This includes *interpolation* (subsection 2.2.16), *rediscretization* (subsection 2.2.17), and *discretizations of linear shift equivariant operators* (subsection 2.2.19).

Convolution is sometimes evaluated without periodicity in practice. Correct discretizations of signals that do not rely on periodicity can be constructed by simply defining finite orthonormal sets of spatial basis functions, and by defining the signal class through the span of that basis.

However, defining corresponding finite orthonormal sets of spectral basis functions is impossible. There, convolution involves finite quantities of information, but the convolution theorem implies infinite quantities of information. Computing convolutions is therefore only possible in the spatial domain. This may be a better compromise in some scenarios. The derivations we present next are compatible with this setting.

2.2.13. The sampling theorem

By explicitly formulating the duality between sampling and repeating in subsection 2.2.11, and by formulating discretization in subsection 2.2.12, we have implicitly formulated the *sampling theorem*, which specifies the necessary conditions for discretizing signals without loss or incorrect capture of information (Shannon, 1949; Petersen and

Middleton, 1962; Landau, 1967; Marvasti, 2012). We formulate this theorem in a more convenient form here, which follows directly from the relationship between the size of the previously derived lattices.

We formalize the criteria for correct discretization using the notion of the volume $\mathfrak{V}$ (Equation 2.2.77) of the spectral domain, and the sampling density $\mathfrak{d}$ (Equation 2.2.78) of the spatial domain:

$$\mathfrak{V}(\boldsymbol{\Omega}) \overset{\text{def}}{=} \int_{\mathbb{R}^d} \mathbb{1}\left\{\boldsymbol{\omega} \in \boldsymbol{\Omega}\right\} d\boldsymbol{\omega} \tag{2.2.77}$$

$$\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}) \overset{\text{def}}{=} \frac{|\mathbf{X}_{\text{spl}}|}{\mathfrak{V}(\mathbf{X})} \tag{2.2.78}$$

Formulating the sampling theorem based on this is not a common approach, but it leads to a result that is easier to apply to our multidimensional analysis, and that is equivalent to the sampling theorem of Shannon (1949); Petersen and Middleton (1962) if the sampling patterns are correctly defined following the process outlined in subsection 2.2.12. It also seamlessly generalizes to a wider class of sampling patterns, as we see next.

Given the spectral and spatial domain of a class of signals (subsection 2.2.2, subsection 2.2.12), it is possible to express the sampling theorem in terms of spatial density and spectral volume (Equation 2.2.79). When this expression is satisfied and the sampling patterns are suitably constructed using lattices, correct sampling is guaranteed:

$$\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}) \geq \mathfrak{V}(\boldsymbol{\Omega}) \tag{2.2.79}$$

This criteria can be extended to apply not only to signals sampled on patterns that are lattices, but to signals sampled on patterns that are randomly formed with a density that is nonuniform (Landau, 1967; Marvasti, 2012). In this case, the criteria applies not on a global sampling pattern density, but on a local sampling pattern density tied to a location $\mathbf{x} \in \mathbf{X}$ that is evaluated in expectation. This also implies the volume of the spectral domain is defined locally. The expression below intuitively captures this relationship, but we do not cover every detail of this complex topic that is better presented by Marvasti (2012):

$$\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}|\mathbf{x}) \geq \mathfrak{V}(\boldsymbol{\Omega}|\mathbf{x}) \tag{2.2.80}$$

2.2.14. Aliasing

Aliasing is an error in discretization that occurs when the sampling theorem (subsection 2.2.13) is violated. It follows as a direct consequence of the duality between sampling in the spatial domain and repeating in the spectral domain (subsection 2.2.11). It occurs precisely through Equation 2.2.73 when we use a spatial sampling lattice $\boldsymbol{\Lambda}_{\text{spl}}$ that gives a corresponding spectral repeating lattice $\boldsymbol{\Gamma}_{\text{rep}}$ on a signal whose spectral domain $\boldsymbol{\Omega}$ cannot be contained by any unit cell of the spectral repeating lattice $\boldsymbol{\Gamma}_{\text{rep}}$.

2.2.15. The spatial basis

While we have explicitly introduced a basis for the spectral domain, we have not explicitly introduced one for the spatial domain. Doing so is desirable as it provides a way to bridge continuous signals and discretized signals in a very useful way. We therefore introduce the *spatial basis*, also called the sine cardinal function, or sinc function (Whittaker, 1915, 1927; Shannon, 1949; Woodward and Davies, 1952; Petersen and Middleton, 1962; Strichartz, 2003).

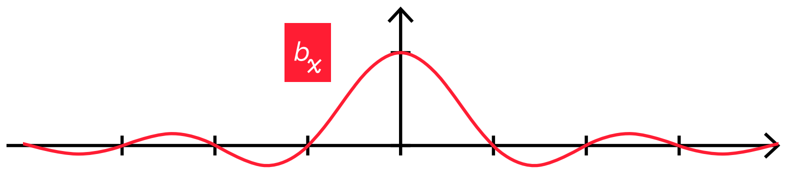


Fig. 2.8. Graph of the spatial basis function.

This shows the spatial basis function in a simple 1-dimensional case. The function evaluates to one at the origin, and to zero at regular intervals, which is important to the orthonormality of sets of spatial basis functions. The function also decays slowly towards zero as we move away from the origin, which prevents exact computation of certain operations that involve it, as we see later. In the illustration, the spatial basis is scaled identically to the spectral basis in Figure 2.8, and the two are compatible with each other, in the sense that sums of one can represent the other.

We can work towards the spatial basis by scrutinizing the process of sampling a signal s at some point $\mathbf{x}'$. What we want is an expression that is identical to evaluating $s(\mathbf{x}')$, but that operates by performing an inner product with a basis function $\langle s, b_{\mathbf{x}'}\rangle$. In the case of the class of unrestricted signals, for which $\mathbf{X}$ and $\mathbf{\Omega}$ cover all of $\mathbb{R}^d$, this can be done with a shifted Dirac delta:

$$\langle s, \mathfrak{T}_{\mathbf{x}'}\{\delta\}\rangle = \int_{\mathbb{R}^d} s(\mathbf{x})\overline{\mathfrak{T}_{\mathbf{x}'}\{\delta\}}(\mathbf{x})d\mathbf{x} = s(\mathbf{x}') \tag{2.2.81}$$

This is not an admissible solution as is, because the Dirac delta is not in the span of any signal class that restricts its spectral domain. This also disregards some finer details of functional analysis (Schwartz, 1957, 1958; Strichartz, 2003).

However, if we project the Dirac delta from its spatial coefficients to its spectral coefficients through $\mathfrak{F}$, reject any part of it that does not belong to the spectral domain $\mathbf{\Omega}$ specific to the signal class, and project back to the spatial coefficients through $\mathfrak{F}^{-1}$, we will obtain the part of the Dirac delta that can be captured by the signal class. We also normalize by the inverse volume (Equation 2.2.77) of the spectral domain $\mathfrak{V}(\mathbf{\Omega})^{-1}$ to satisfy orthonormality.

This is all expressed in the definition below, which leverages some properties of the spectral basis to simplify the result (the conjugate of the spectral basis as in Equation 2.2.19 and the multiplication of the same spectral basis at different points in space as in Equation 2.2.25):

$$
\begin{aligned}
b_{\mathbf{x}'}(\mathbf{x}) &\overset{\text{def}}{=} \mathfrak{F}^{-1}\left\{\mathfrak{F}\{\mathfrak{T}_{\mathbf{x}'}\{\delta\}\}\frac{\mathbb{1}\{\ \cdot\ \in \boldsymbol{\Omega}\}}{\mathfrak{V}(\boldsymbol{\Omega})}\right\}(\mathbf{x}) && (2.2.82)\\
&= \int_{\mathbb{R}^d} \frac{\mathbb{1}\{\ \cdot\ \in \boldsymbol{\Omega}\}}{\mathfrak{V}(\boldsymbol{\Omega})}\left(\int_{\mathbb{R}^d} \mathfrak{T}_{\mathbf{x}'}\{\delta\}(\mathbf{x}'')\overline{b_\omega(\mathbf{x}'')}d\mathbf{x}''\right) b_\omega(\mathbf{x})d\boldsymbol{\omega} && (2.2.83)\\
&= \frac{1}{\mathfrak{V}(\boldsymbol{\Omega})}\int_{\boldsymbol{\Omega}} \overline{b_\omega(\mathbf{x}')}b_\omega(\mathbf{x})d\boldsymbol{\omega} && (2.2.84)\\
&= \frac{1}{\mathfrak{V}(\boldsymbol{\Omega})}\int_{\boldsymbol{\Omega}} b_\omega(\mathbf{x}-\mathbf{x}')d\boldsymbol{\omega} && (2.2.85)\\
&= \frac{1}{\mathfrak{V}(\boldsymbol{\Omega})}\int_{\boldsymbol{\Omega}} \mathfrak{T}_{\mathbf{x}'}\{b_\omega\}(\mathbf{x})d\boldsymbol{\omega} && (2.2.86)
\end{aligned}
$$

By design, we recover the Dirac delta as $\boldsymbol{\Omega}$ tends to $\mathbb{R}^d$. This is not exactly formally correct in the sense of Schwartz (1957, 1958); Strichartz (2003), but for our limited analysis, this is sufficiently accurate:

$$
\lim_{\boldsymbol{\Omega}\to\mathbb{R}^d} \mathfrak{F}^{-1}\left\{\mathfrak{F}\{\mathfrak{T}_{\mathbf{x}'}\{\delta\}\}\frac{\mathbb{1}\{\ \cdot\ \in \boldsymbol{\Omega}\}}{\mathfrak{V}(\boldsymbol{\Omega})}\right\}(\mathbf{x}) = \mathfrak{T}_{\mathbf{x}'}\{\delta\} \qquad (2.2.87)
$$

We can leverage Equation 2.2.86 to show that the spatial basis is shift equivariant, in the sense that it only considers the position of its evaluation $\mathbf{x}$ relative to the position of its anchor $\mathbf{x}'$:

$$
\begin{aligned}
b_{\mathbf{x}'}(\mathbf{x}) &= \frac{1}{\mathfrak{V}(\boldsymbol{\Omega})}\int_{\boldsymbol{\Omega}} \mathfrak{T}_{\mathbf{x}'}\{b_\omega\}(\mathbf{x})d\boldsymbol{\omega} && (2.2.88)\\
&= \mathfrak{T}_{\mathbf{x}'}\left\{\frac{1}{\mathfrak{V}(\boldsymbol{\Omega})}\int_{\boldsymbol{\Omega}} \mathfrak{T}_{0}\{b_\omega\}(\ \cdot\ )d\boldsymbol{\omega}\right\}(\mathbf{x}) && (2.2.89)\\
&= \mathfrak{T}_{\mathbf{x}'}\{b_0\}(\mathbf{x}) && (2.2.90)\\
&= b_0(\mathbf{x}-\mathbf{x}') && (2.2.91)
\end{aligned}
$$

Because the spatial basis is constructed as a sum of spectral bases (Equation 2.2.86) that satisfy the mirrored conjugate equality $\overline{\mathfrak{M}\{b_\omega\}} = b_\omega$ (Equation 2.2.19), the spatial basis necessarily also satisfies the mirrored conjugate equality $\overline{\mathfrak{M}\{b_0\}} = b_0$, independently of the spectral domain $\mathbf{\Omega}$ it is constructed on:

$$b_0(\mathbf{x}) = \frac{1}{\mathfrak{V}(\mathbf{\Omega})} \int_{\mathbf{\Omega}} b_\omega(\mathbf{x}) d\boldsymbol{\omega} \tag{2.2.92}$$

$$= \frac{1}{\mathfrak{V}(\mathbf{\Omega})} \int_{\mathbf{\Omega}} \overline{\mathfrak{M}\{b_\omega\}}(\mathbf{x}) d\boldsymbol{\omega} \tag{2.2.93}$$

$$= \overline{\mathfrak{M}\left\{ \frac{1}{\mathfrak{V}(\mathbf{\Omega})} \int_{\mathbf{\Omega}} b_\omega(\,\cdot\,) d\boldsymbol{\omega} \right\}}(\mathbf{x}) \tag{2.2.94}$$

$$= \overline{\mathfrak{M}\{b_0\}}(\mathbf{x}) \tag{2.2.95}$$

This property can be used to show that the spatial basis is strictly real if the spectral domain has mirror symmetry, meaning $\boldsymbol{\omega} \in \mathbf{\Omega} \iff -\boldsymbol{\omega} \in \mathbf{\Omega}$. Our analysis assumes this scenario without loss of generality.

Because the spatial basis respects the mirrored conjugate equality (Equation 2.2.95), and because it is shift equivariant (Equation 2.2.90), we can reuse the alternate expression of convolution as an inner product against a shifted, mirrored, conjugated signal (Equation 2.2.39) to write the inner product with some spatial basis $\langle s, b_{\mathbf{x}'} \rangle$ as the result of a convolution $(s \ * \ b_0)(\mathbf{x}')$:

$$\langle s, b_{\mathbf{x}'} \rangle = \left\langle s, \mathfrak{T}_{\mathbf{x}'}\{\overline{\mathfrak{M}\{b_0\}}\} \right\rangle \tag{2.2.96}$$

$$= (s \ * \ b_0)(\mathbf{x}') \tag{2.2.97}$$

This reexpression as a convolution will be useful when defining *interpolation* in subsection 2.2.16 and *rediscretization* in subsection 2.2.17. Note that because the Dirac delta fulfills the same mirrored conjugate equality $\delta = \overline{\mathfrak{M}\{\delta\}}$, the relationship that ties the inner product with the Dirac delta to sampling in Equation 2.2.81 can also be written as a convolution using Equation 2.2.39:

$$\langle s, \mathfrak{T}_{\mathbf{x}'}\{\delta\} \rangle = \left\langle s, \mathfrak{T}_{\mathbf{x}'}\{\overline{\mathfrak{M}\{\delta\}}\} \right\rangle \tag{2.2.98}$$

$$= (s \ * \ \delta)(\mathbf{x}') \tag{2.2.99}$$

Again, we have explicitly designed the spatial basis so that taking an inner product $\langle s, b_{\mathbf{x}'} \rangle$ of some signal s is equivalent to sampling the signal $s(\mathbf{x}')$. We can show that equivalence to sampling indeed holds with our spectrally limited Dirac delta workaround. Since the spatial basis leaves no spectral coefficients unaccounted for, the second term in Equation 2.2.102 is zero. Since the sum of the right sides of the two $\mathfrak{F}\{s\}$ terms is equal to the Fourier transform of the limiting result of Equation 2.2.87, the whole expression is a convolution with the Dirac delta. This convolution is identical to the inner product with the Dirac delta by equation

Equation 2.2.99, which in turn is identical to sampling the signal by Equation 2.2.81:

$$\langle s, b_{\mathbf{x}'} \rangle = (s \ * \ b_0)(\mathbf{x}') \tag{2.2.100}$$

$$= \mathfrak{F}^{-1}\left\{ \mathfrak{F}\{s\}\mathfrak{F}\{\delta\} \frac{\mathbb{1}\{\ \cdot\ \in \mathbf{\Omega}\}}{\mathfrak{V}(\mathbf{\Omega})} \right\}(\mathbf{x}') \tag{2.2.101}$$

$$= \mathfrak{F}^{-1}\left\{ \mathfrak{F}\{s\}\mathfrak{F}\{\delta\} \frac{\mathbb{1}\{\ \cdot\ \in \mathbf{\Omega}\}}{\mathfrak{V}(\mathbf{\Omega})} \right\}(\mathbf{x}') + \mathfrak{F}^{-1}\left\{ \mathfrak{F}\{s\}\mathfrak{F}\{\delta\} \frac{\mathbb{1}\left\{\ \cdot\ \in \overline{\mathbf{\Omega}}\right\}}{\mathfrak{V}(\overline{\mathbf{\Omega}})} \right\}(\mathbf{x}') \tag{2.2.102}$$

$$= (s \ * \ \delta)(\mathbf{x}') \tag{2.2.103}$$

$$= \langle s, \mathfrak{T}_{\mathbf{x}'}\{\delta\} \rangle \tag{2.2.104}$$

$$= s(\mathbf{x}') \tag{2.2.105}$$

The sampling property of Equation 2.2.105 holds for any signals whose nonzero spectral coefficients are spanned by the spectral domain $\mathbf{\Omega}$ used to construct the spatial basis. Necessarily, the spatial basis itself fulfills that constraint, meaning it can sample itself through its inner product. This provides a trivial expression for the inner product of two spatial bases $b_{\mathbf{x}}$ and $b_{\mathbf{x}'}$:

$$\langle b_{\mathbf{x}}, b_{\mathbf{x}'} \rangle = b_{\mathbf{x}}(\mathbf{x}') \tag{2.2.106}$$

The set of spatial bases $\{b_{\mathbf{x}'} | \mathbf{x}' \in \mathbf{X}\}$ can be made orthonormal (Equation 2.2.5) for signal classes defined over certain spectral domains $\mathbf{\Omega}$ and spatial domains $\mathbf{X}$. This is always true when constructing spatial sampling sets $\mathbf{X}_{\text{spl}}$ using the process outlined in subsection 2.2.12.

2.2.16. Interpolation

We have introduced the sampling operator in subsection 2.2.9 and the principle of discretization in subsection 2.2.12, which provides us a way to reduce a continuous signal s to a discretized signal $\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}$. This gives us access to s at points $\mathbf{x} \in \mathbf{X}_{\text{spl}}$, but in many scenarios we may want to evaluate s any point $\mathbf{x} \notin \mathbf{X}_{\text{spl}}$. For this, we must somehow reconstruct s from $\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}$. This process is *interpolation* (Whittaker, 1915, 1927; Shannon, 1949; Petersen and Middleton, 1962).

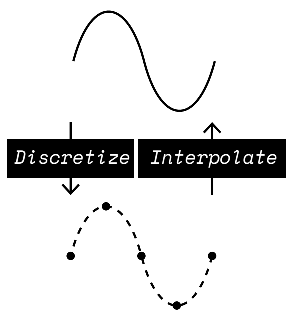


Fig. 2.9. Graph showing interpolation as the inverse of discretization.

This intuitively shows how discretization and interpolation are inverse processes that convert between *continuous signals* and *discrete signals*. This visual convention is reused in later figures that show interpolation, rediscretization, and discretized operators. The impulses of the discretization are not represented directly. Instead, the coordinates of the sample position and sample value are shown with points, and the continuous signal that would be reconstructed through interpolation is drawn with dashed lines.

We can derive interpolation by leveraging the spatial basis (subsection 2.2.15). Since the inner product against the spatial basis represents sampling by definition (Equation 2.2.105), $s(\mathbf{x})$ is equal to $\langle s, b_{\mathbf{x}}\rangle$. Since $\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}$ provides us with the full set of coefficients of an orthonormal set of spatial bases $\{b_{\mathbf{x}'} | \mathbf{x}' \in \mathbf{X}_{\text{spl}}\}$, and since we effectively wish to recover the coefficient of a single spatial basis $b_{\mathbf{x}}$, we can do so by projecting from the spatial coefficients $\{s(\mathbf{x}') | \mathbf{x}' \in \mathbf{X}_{\text{spl}}\}$ to the spatial coefficient $s(\mathbf{x})$ (Equation 2.2.3). We use the self-sampling property of the spatial basis (Equation 2.2.106) and the shift equivariance of the spatial basis (Equation 2.2.90) to simplify, and apply the definition of the convolution (Equation 2.2.32) to arrive at an expression for interpolation:

$$s(\mathbf{x}) = \langle s, b_{\mathbf{x}}\rangle \tag{2.2.107}$$

$$= \sum_{\mathbf{x}' \in \mathbf{X}_{\text{spl}}} \langle s, b_{\mathbf{x}'}\rangle \langle b_{\mathbf{x}'}, b_{\mathbf{x}}\rangle \tag{2.2.108}$$

$$= \int_{\mathbb{R}^d} \mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}(\mathbf{x}') \langle b_{\mathbf{x}'}, b_{\mathbf{x}}\rangle \, d\mathbf{x}' \tag{2.2.109}$$

$$= \int_{\mathbb{R}^d} \mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}(\mathbf{x}') b_0(\mathbf{x} - \mathbf{x}') d\mathbf{x}' \tag{2.2.110}$$

$$\mathfrak{I}\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}\}(\mathbf{x}) \overset{\text{def}}{=} \left(\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} \; * \; b_0\right)(\mathbf{x}) \tag{2.2.111}$$

The form that the interpolation process takes is convenient; it simply consists in convolving the sampled signal with the spatial basis.

When $\mathbf{X}$ and $\mathbf{\Omega}$ are bounded and sampled (subsection 2.2.12), evaluation of the convolution requires summing over a finite number of terms. We can therefore compute it.

When interpolating values $\mathbf{x} \notin \mathbf{X}_{\text{spl}}$, the values at which the spatial basis is evaluated decay very slowly as a function of the distance from the origin. This means it is difficult to reduce computational cost by truncating small values of the kernel. This motivates approximate interpolation, where a better-behaved kernel is chosen instead. This kernel is picked to perform a conceptually similar operation to the spatial basis. It is often formulated the same way we constructed the spatial basis in Equation 2.2.82, except the indicator function is replaced by a soft approximation. We apply this in our method in section 3.4.

2.2.17. Rediscretization

We have introduced interpolation in subsection 2.2.16 with the goal of taking a signal s in its discretized form $\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}$ and evaluating *individual points* $\mathbf{x} \notin \mathbf{X}_{\text{spl}}$ which are not directly part of the discretization. The interpolation process effectively allows *reversing discretization.* The *rediscretization process* is very closely related; it instead *translates from one discretization to another.* It evaluates not *individual points* $\mathbf{x} \notin \mathbf{X}_{\text{spl}}$, but *sets of points* $\mathbf{x} \in \mathbf{X}'_{\text{spl}} \notin \mathbf{X}_{\text{spl}}$ that belong to a different discretization (Whittaker, 1915, 1927; Shannon, 1949; Petersen and Middleton, 1962; Strichartz, 2003).

This process essentially involves interpolating a discretized signal to a continuous signal and then sampling the continuous signal to form a new discretized signal. This sampling step is subject to the sampling theorem (subsection 2.2.13), meaning interpolation cannot be applied blindly. The density of the new discretization must be taken into account.

In the derivations that follow, we assume are already working with a discretization whose sampling patterns are correctly formed (subsection 2.2.12), and whose sampling density is optimally tight given the spectral volume of the signal, meaning $\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}) = \mathfrak{V}(\mathbf{\Omega})$ (subsection 2.2.13).

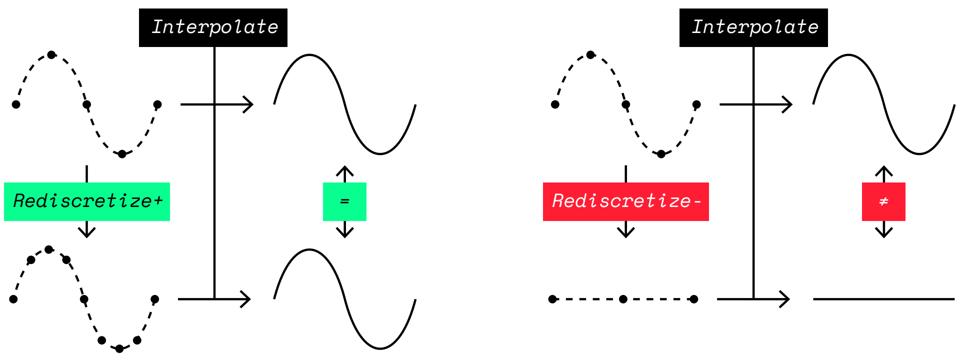


Fig. 2.10. Graph of rediscretization with increased or decreased density.

This shows a signal being rediscretized with either doubled or halved sample density. With increased density, information is preserved, but with decreased density, information can be lost.

If $\mathfrak{d}(\mathbf{X}'_{\text{spl}}, \mathbf{X}') \geq \mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X})$, the sampling theorem (Equation 2.2.79) trivially holds. We can simply use Equation 2.2.111 to recover the values on this new discretization. If $\mathbf{X}$ is also unchanged, this is also true if and only if $|\mathbf{X}'_{\text{spl}}| \geq |\mathbf{X}_{\text{spl}}|$.

If $\mathfrak{d}(\mathbf{X}'_{\text{spl}}, \mathbf{X}') < \mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X})$, the sampling theorem (Equation 2.2.79) is violated. It is impossible to fully capture the information contained in the signal, but it is possible to prevent aliasing (subsection 2.2.14). For this, we must reduce the signal's original spectral domain $\boldsymbol{\Omega}$ to the new, restricted spectral domain $\boldsymbol{\Omega}'$.

This can be achieved easily through convolution with a filter kernel that only allows through spectral coefficients that belong to the new required spectral domain $\boldsymbol{\Omega}'$. This kernel is necessarily equal to the spatial basis b'_0 of the new discretization by definition (Equation 2.2.86):

$$b'_0(\mathbf{x}) \overset{\text{def}}{=} \mathfrak{F}^{-1}\left\{\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\,\in\boldsymbol{\Omega}'\}}{\mathfrak{V}(\boldsymbol{\Omega}')}\right\}(\mathbf{x}) \tag{2.2.112}$$

This allows writing down an equation for the process of rediscretization. Because of the associativity and distributivity of convolution (Equation 2.2.41, Equation 2.2.43), the whole rediscretization process can be expressed as convolution with a single filter kernel $b_0 * b'_0$ from points on $\mathbf{X}_{\text{spl}}$ to $\mathbf{X}'_{\text{spl}}$:

$$\mathfrak{S}_{\mathbf{X}'_{\text{spl}}}\{\mathfrak{I}\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}\} * b'_0\}(\mathbf{x}) = \mathfrak{S}_{\mathbf{X}'_{\text{spl}}}\{\big(\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} \ * \ b_0\big) * b'_0\}(\mathbf{x}) \tag{2.2.113}$$

$$= \mathfrak{S}_{\mathbf{X}'_{\text{spl}}}\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} * (b'_0 \ * \ b_0)\}(\mathbf{x}) \tag{2.2.114}$$

We can further simplify the expression for the kernel. We expand the convolution using the definition of the spatial basis (Equation 2.2.97), and the convolution theorem (Equation 2.2.47). We then separate the spectral basis b_0, which spans the larger spectral domain $\mathbf{\Omega} \supset \mathbf{\Omega}'$ into two parts, which respectively occupy $\mathbf{\Omega}'$, the new spectral domain, and $\mathbf{\Omega} - \mathbf{\Omega}'$, the part of the original spectral domain that is lost. We discard the second convolution as it evaluates to zero:

$$(b_0' \ * \ b_0)(\mathbf{x}') \tag{2.2.115}$$

$$= \mathfrak{F}^{-1}\left\{\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega}')}\right)\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}\}}{\mathfrak{V}(\mathbf{\Omega})}\right)\right\}(\mathbf{x}') \tag{2.2.116}$$

$$= \mathfrak{F}^{-1}\left\{\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega}')}\right)\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega})}\right)\right\}(\mathbf{x}') + \mathfrak{F}^{-1}\left\{\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega}')}\right)\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega} - \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega})}\right)\right\}(\mathbf{x}') \tag{2.2.117}$$

$$= \mathfrak{F}^{-1}\left\{\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega}')}\right)\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega})}\right)\right\}(\mathbf{x}') \tag{2.2.118}$$

We then extract a factor $\mathfrak{V}(\mathbf{\Omega}')/\mathfrak{V}(\mathbf{\Omega})$ to the left of the convolution which reveals we have a scaled self-convolution of the new spatial basis $b_0' \ * \ b_0'$. We add a second convolution whose value is zero, which covers $\overline{\mathbf{\Omega}'}$, the complement of the new spectral domain. The sum of the right sides of the convolutions is then identical to the limiting case of the Dirac delta of Equation 2.2.87. This is equal to direct evaluation of the spatial basis because of Equation 2.2.97 and Equation 2.2.105:

$$(b_0' \ * \ b_0)(\mathbf{x}') \tag{2.2.119}$$

$$= \frac{\mathfrak{V}(\mathbf{\Omega}')}{\mathfrak{V}(\mathbf{\Omega})}\mathfrak{F}^{-1}\left\{\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega}')}\right)\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega}')}\right)\right\}(\mathbf{x}') \tag{2.2.120}$$

$$= \frac{\mathfrak{V}(\mathbf{\Omega}')}{\mathfrak{V}(\mathbf{\Omega})}\mathfrak{F}^{-1}\left\{\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega}')}\right)\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega}')}\right)\right\}(\mathbf{x}') + \frac{\mathfrak{V}(\mathbf{\Omega}')}{\mathfrak{V}(\mathbf{\Omega})}\mathfrak{F}^{-1}\left\{\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\{\,\cdot\, \in \mathbf{\Omega}'\}}{\mathfrak{V}(\mathbf{\Omega}')}\right)\left(\mathfrak{F}\{\delta\}\frac{\mathbb{1}\left\{\,\cdot\, \in \overline{\mathbf{\Omega}'}\right\}}{\mathfrak{V}(\overline{\mathbf{\Omega}'})}\right)\right\}(\mathbf{x}') \tag{2.2.121}$$

$$= \frac{\mathfrak{V}(\mathbf{\Omega}')}{\mathfrak{V}(\mathbf{\Omega})}(b_0' \ * \ \delta)(\mathbf{x}') \tag{2.2.122}$$

$$= \frac{\mathfrak{V}(\mathbf{\Omega}')}{\mathfrak{V}(\mathbf{\Omega})}b_0'(\mathbf{x}') \tag{2.2.123}$$

This finally gives a simple expression for rediscretization with lowered density $\mathfrak{d}(\mathbf{X}'_{\text{spl}}, \mathbf{X}) < \mathfrak{d}(\mathbf{X}'_{\text{spl}}, \mathbf{X})$:

$$\mathfrak{S}_{\mathbf{X}'_{\text{spl}}}\{\mathfrak{I}\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}\} * b'_0\}(\mathbf{x}) = \mathfrak{S}_{\mathbf{X}'_{\text{spl}}}\left\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} * \frac{\mathfrak{V}(\mathbf{\Omega}')}{\mathfrak{V}(\mathbf{\Omega})} b'_0\right\}(\mathbf{x}) \tag{2.2.124}$$

When $\mathbf{X}$ and $\mathbf{\Omega}$ are bounded and sampled (subsection 2.2.12), evaluation of the convolution requires summing over a finite number of terms. We can therefore compute it.

As with exact interpolation in subsection 2.2.16, computational concerns arise, which leads to the implementation of similar approximate rediscretization methods. We implement this in our method in section 3.4.

2.2.18. Discretization invariance

The adaptation of operators that apply on *continuous signals* to operators that *identically* apply on *discretized signals* is central to our work. This notion of an identical operator is formalized by *discretization invariance*, which we now introduce.

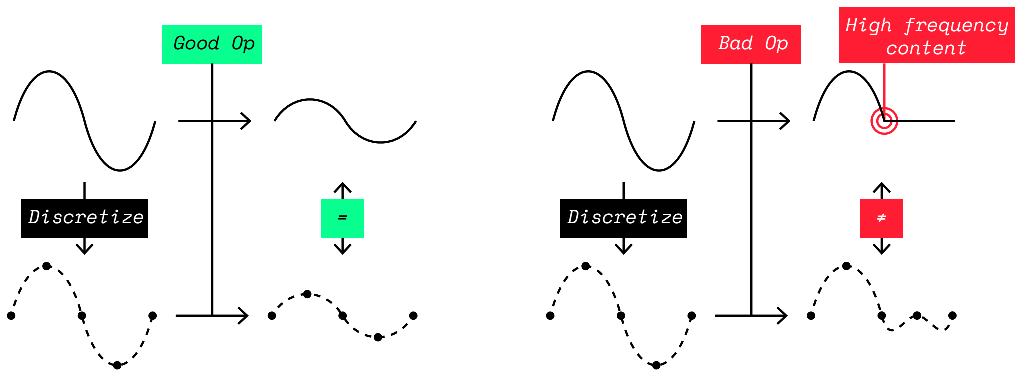


Fig. 2.11. Graph of operators with and without discretization invariance.

This shows the application of an operator with discretization invariance (a simple scaling) and of another operator without discretization invariance (the pointwise ReLU activation function). Intuitively, discretization invariance means that applying the operator on the continuous signal yields the same result as applying the operator on the discrete signal.

An operator on continuous signals g has an adapted operator h on discretized signals that is *discretization invariant* if and only if both operators express the same mapping in the space of continuous signals belonging to some specific signal class (subsection 2.2.2) (where $\mathfrak{S}$ is the sampling operator as defined in subsection 2.2.9, and $\mathfrak{I}$ is the interpolation operator as defined in subsection 2.2.16):

$$g(s) = \mathfrak{I}\left\{h\left(\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}\right)\right\} \tag{2.2.125}$$

Here, it is understood that h can either manipulate the weighted sum of Dirac deltas expressed through $\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}$, or manipulate the pairs of sample positions and sample values $\{(\mathbf{x}_{\text{spl}}, s(\mathbf{x}_{\text{spl}}))|\mathbf{x}_{\text{spl}} \in \mathbf{X}_{\text{spl}}\}$ that compose it. We generally require that h be adaptable to arbitrary discretizations $\mathbf{X}_{\text{spl}}$. We however allow h to require us to respect certain constraints on the sampling density $\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X})$ according to the spectral volume $\mathfrak{V}(\mathbf{\Omega})$.

We can give a condition under which discretization invariance is impossible using the sampling theorem. Given a signal s that belongs to a signal class such that $\widehat{s}(\boldsymbol{\omega}) = 0$ is satisfied for all $\boldsymbol{\omega} \notin \mathbf{\Omega}$ for all s, and given a sampling pattern such that $\mathfrak{V}(\mathbf{\Omega}) = \mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X})$, if the signal $s' = g(s)$ resulting from applying the operator exits the domain of the signal class such that $\widehat{s'}(\boldsymbol{\omega}) \neq 0$ for at least one $\boldsymbol{\omega} \notin \mathbf{\Omega}$ for at least one s, it is impossible to create an operator on discretized signals that could perform the same mapping because its output would violate the sampling theorem its discretization is subject to (subsection 2.2.13).

2.2.19. Discretization of linear shift equivariant operators

We introduce a method for adaptation of operators on *continuous* signals that are *linear and shift equivariant* to operators on *discretized* signals which satisfies discretization invariance (subsection 2.2.18).

We follow the same general process used to derive interpolation in subsection 2.2.16. We apply the operator $\mathfrak{O}$ on some continuous signal, rewrite that signal as an inner product with a spatial basis, and perform a change of basis from the orthonormal set of spatial bases $\{b_{\mathbf{x}'}|\mathbf{x}' \in \mathbf{X}_{\text{spl}}\}$ to a spatial basis $b_{\mathbf{x}}$, effectively projecting from the spatial coefficients $\{s(\mathbf{x}')|\mathbf{x}' \in \mathbf{X}_{\text{spl}}\}$ to the spatial coefficient $s(\mathbf{x})$ (Equation 2.2.3). We use the self-sampling property of the spatial basis (Equation 2.2.106) and the shift equivariance of the spatial basis (Equation 2.2.90) to simplify. We then leverage the fact that the operator only applies to the right side of the equation, is linear, and is shift equivariant to bring it inside the sum. We rewrite as an integral over a sampling operator that corresponds to a convolution (Equation 2.2.32), which gives a very convenient way to express the operator on discretized signals:

$$\begin{aligned}
\mathfrak{O}\{s\}(\mathbf{x}) &= \mathfrak{O}\{\langle s, b_{\,.}\rangle\}(\mathbf{x}) && (2.2.126)\\
&= \mathfrak{O}\left\{\sum_{\mathbf{x}'\in\mathbf{X}_{\text{spl}}} \langle s, b_{\mathbf{x}'}\rangle \langle b_{\mathbf{x}'}, b_{\,.}\rangle\right\}(\mathbf{x}) && (2.2.127)\\
&= \mathfrak{O}\left\{\sum_{\mathbf{x}'\in\mathbf{X}_{\text{spl}}} s(\mathbf{x}')b_0(\,\cdot\, - \mathbf{x}')\right\}(\mathbf{x}) && (2.2.128)\\
&= \sum_{\mathbf{x}'\in\mathbf{X}_{\text{spl}}} s(\mathbf{x}')\mathfrak{O}\{b_0\}(\mathbf{x}-\mathbf{x}') && (2.2.129)\\
&= \int_{\mathbb{R}^d} \mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\}(\mathbf{x}')\mathfrak{O}\{b_0\}(\mathbf{x}-\mathbf{x}')d\mathbf{x}' && (2.2.130)\\
&= \left(\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} \ * \ \mathfrak{O}\{b_0\}\right)(\mathbf{x}) && (2.2.131)
\end{aligned}$$

$$\implies \mathfrak{S}_{\mathbf{X}_{\text{spl}}}\Big\{\mathfrak{O}\{s\}\Big\}(\mathbf{x}) \overset{\text{def}}{=} \mathfrak{S}_{\mathbf{X}_{\text{spl}}}\Big\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} * \mathfrak{O}\{b_0\}\Big\}(\mathbf{x}) \qquad (2.2.132)$$

Because Equation 2.2.126 expresses a continuous signal that is equal to Equation 2.2.131, and because Equation 2.2.132 is a correct discretization of the resulting signal, the discretized operator is necessarily discretization invariant (subsection 2.2.18).

The discretization of a linear shift equivariant operator can be expressed in terms of the transformation it applies to the spatial basis of the discretization. This forms a kernel $\mathfrak{O}\{b_0\}$ which represents the operator.

If the kernel $\mathfrak{O}\{b_0\}$ that represents the operator is nonzero for only a few values of $\mathbf{X}_{\text{spl}}$, few values need to be considered in Equation 2.2.132, meaning its discretization can be implemented exactly with good computational efficiency. Otherwise, approximate methods may be preferred in practice.

2.2.20. Discretization of the convolution operator

We apply our previous results (subsection 2.2.19) to discretize convolution (subsection 2.2.7). We also provide an interpretation of two distinct cases that occur when discretizing convolutions. Note that the discretization of convolution is in many ways similar to the process of rediscretization (subsection 2.2.17); only the kernel changes.

Because the convolution operator (subsection 2.2.7) is linear and shift equivariant, the result of Equation 2.2.132 applies:

$$\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\Big\{s * s'\Big\}(\mathbf{x}) \overset{\text{def}}{=} \mathfrak{S}_{\mathbf{X}_{\text{spl}}}\Big\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} * (s' * b_0)\Big\}(\mathbf{x}) \qquad (2.2.133)$$

If s' has a spectral domain contained within that of s' such that $\mathbf{\Omega}' \subseteq \mathbf{\Omega}$, and given the discretization respects the sampling theorem (Equation 2.2.79) based on $\mathbf{\Omega}$, then $(s' * b_0) = s'$ by the definition of the spatial basis (Equation 2.2.105).

If s' has a spectral domain exceeding that of s' such that $\mathbf{\Omega}' \supset \mathbf{\Omega}$, and given the same assumption on discretization, then $(s' * b_0)$ has the effect of truncating away spectral content from s' that cannot be captured by the discretization, and that does not exist in $s * s'$ either because it does not exist in s, as implied by the convolution theorem (Equation 2.2.47).

2.2.21. Discretization of the partial derivative operator

We apply our previous results (subsection 2.2.19) once more to discretize the partial derivative operator (subsection 2.2.8), which holds an important place in our work. We additionally provide a short discussion of exact and approximate methods for its implementation.

Because the partial derivative operator (subsection 2.2.21) is linear and shift equivariant, the result of Equation 2.2.132 applies:

$$\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\left\{\mathfrak{D}^{\alpha}_{\mathbf{x}}\{s\}\right\}(\mathbf{x}) \overset{\text{def}}{=} \mathfrak{S}_{\mathbf{X}_{\text{spl}}}\left\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} * \mathfrak{D}^{\alpha}_{\mathbf{x}}\{b_0\}\right\}(\mathbf{x}) \tag{2.2.134}$$

Equation 2.2.134 provides an exact discretization for any partial derivative operator. As with interpolation and rediscretization, convolution against a non sparse kernel ($\mathfrak{D}^{\alpha}_{\mathbf{x}}\{b_0\}$) is required for exact computation. Approximation schemes that substitute b_0 for a better numerically behaved kernel are often used for this reason.

2.2.22. Discretization of the shift operator

We apply our previous results (subsection 2.2.19) a final time to discretize the shift operator (subsection 2.2.3). The goal of this derivation is not to arrive at a practical implementation, but to underline a critical distinction between the discretization of the shift operator with *discrete shifts* and *continuous shifts*. This is central to the discussion of *discrete shift equivariance* and *continuous shift equivariance* we later provide in subsection 2.2.23.

Because the shift operator (subsection 2.2.3) is linear and shift equivariant, the result of Equation 2.2.132 applies:

$$\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\left\{\mathfrak{T}_{\mathbf{x}'}\{s\}\right\}(\mathbf{x}) \overset{\text{def}}{=} \mathfrak{S}_{\mathbf{X}_{\text{spl}}}\left\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} * \mathfrak{T}_{\mathbf{x}'}\{b_0\}\right\}(\mathbf{x}) \tag{2.2.135}$$

$$= \mathfrak{S}_{\mathbf{X}_{\text{spl}}}\left\{\mathfrak{S}_{\mathbf{X}_{\text{spl}}}\{s\} * b_{\mathbf{x}'}\right\}(\mathbf{x}) \tag{2.2.136}$$

We can use this discretization to provide an interpretation of both *discrete shifts* and *continuous shifts*.

With *discrete shifts*, because evaluation of the spatial basis is identical to an inner product with the spatial basis (Equation 2.2.106), the term that is convolved against in Equation 2.2.136 is identical to the inner product below. Because $\mathbf{X}_{\text{spl}}$ forms an orthonormal basis, the inner products between basis functions of $\mathbf{X}_{\text{spl}}$ respect the orthonormality constraint (Equation 2.2.5). Because we only consider discrete shifts that correspond to points of this discretization, the basis function is only ever evaluated at these points for which the kernel of the operator discretization evaluates to zero for all points except one, which evaluates to one. This means *the discrete shift operator only applies a permutation between sample values*:

$$b_{\mathbf{x}'}(\mathbf{x}) = \langle b_{\mathbf{x}}, b_{\mathbf{x}'} \rangle \tag{2.2.137}$$

$$= \mathbb{1}\{\mathbf{x} = \mathbf{x}'\} \;\; \forall \; \mathbf{x}, \mathbf{x}' \in \mathbf{X}_{\text{spl}} \tag{2.2.138}$$

With *continuous shifts*, Equation 2.2.138 does not hold, and the shift operator effectively evaluates the spatial basis at intermediate points that are never zero. This means *the continuous shift operator does not merely apply a permutation between sample values.*

2.2.23. Discretization of operators with shift equivariance

With the intuition for the distinction between *discrete shifts* and *continuous shifts* we have introduced in subsection 2.2.22, we can now study how shift equivariance applies to discretizations of operators in two separate scenarios: *discrete shift equivariance* and *continuous shift equivariance*. These two flavours of shift equivariance are not equally strong and come with different requirements that define the implementation of every component of our method (section 3.1).

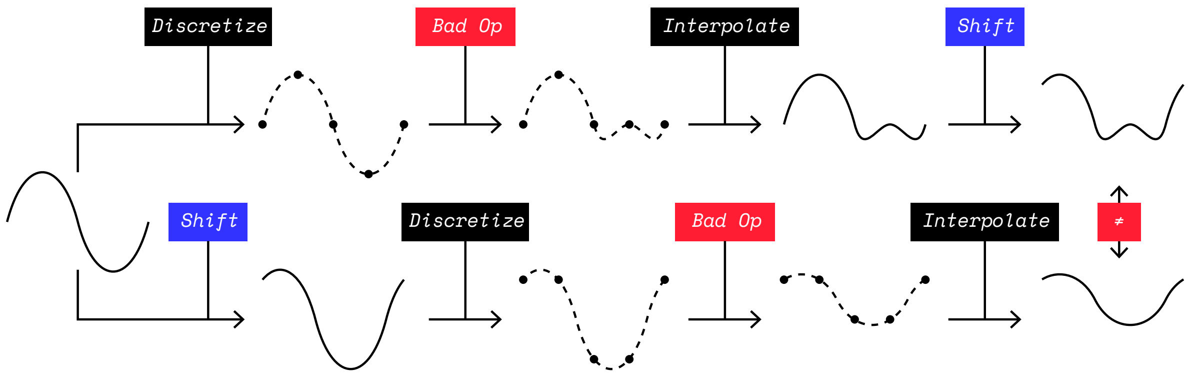


Fig. 2.12. Graph of an operator without continuous shift equivariance.

This shows the application of an operator (the pointwise ReLU activation function) that is not continuous shift equivariant. The discretized operator does not commute with the continuous shift operator.

Discretizations of pointwise operators are discrete shift equivariant. The discrete shift operator is effectively a permutation between sample values (subsection 2.2.22, Equation 2.2.138). Any discretized pointwise operator is equivariant to permutation, and therefore any discretized pointwise operator is equivariant to discrete shift.

Discretizations of pointwise operators are continuous shift equivariant if and only if they are discretization invariant. The continuous shift operator is not equivalent to a permutation between sample values (subsection 2.2.22). Any discretized pointwise operator is therefore not necessarily equivariant to continuous shift.

When a continuous pointwise operator is adapted into a discretized pointwise operator while *respecting* discretization invariance (subsection 2.2.18), the two variants of the operator must express the same map on continuous signals. In this case, since the continuous pointwise operator is continuous shift equivariant by definition, the discretized pointwise operator is also necessarily continuous shift equivariant.

When a continuous pointwise operator is adapted into a discretized pointwise operator while *violating* discretization invariance (subsection 2.2.18), then application of the continuous pointwise operator can yield signals which violate the sampling theorem. In this case, aliasing artifacts are introduced (subsection 2.2.14). These aliasing artifacts are sensitive to continuous shifts, which violates continuous shift equivariance. This scenario can occur specifically with nonlinear pointwise operators, as we later discuss in subsection 2.2.24.

Discretizations of linear shift equivariant operators are continuous shift equivariant. Any continuous linear shift equivariant operator can be adapted into a discrete linear shift equivariant operator trivially while respecting discretization invariance (subsection 2.2.19). Any such discrete operator is continuous shift equivariant by a similar argument.

2.2.24. Discretization of pointwise nonlinear operators

As we have discussed in subsection 2.2.23, pointwise operators are not trivially continuous shift equivariant. Pointwise nonlinear operators are central to convolutional architectures, so the study of the conditions for correct implementation of this equivariance is crucial to our work.

Pointwise nonlinear operators tend to widen the spectral domain, which can lead to violation of the sampling theorem (subsection 2.2.13) and cause aliasing (subsection 2.2.14), which violates discretization invariance (subsection 2.2.18), and thus violates continuous shift equivariance (subsection 2.2.23). We provide an analysis of the error this introduces and propose two mitigation strategies to provide approximate discretization invariance and continuous shift equivariance.

The degree to which a pointwise nonlinearity broadens the spectral domain of a signal can be studied through polynomial decomposition of the pointwise nonlinearity, and through the convolution theorem. The analysis that follows is applied in a single dimension to lighten notation, but it generalizes to higher dimensions. We base our analysis on some nonlinearity $a : \mathbb{R} \to \mathbb{R}$ with polynomial expansion coefficients p_n:

$$a(x) = \sum_{n=0}^{\infty} p_n x^n \tag{2.2.139}$$

We leverage the convolution theorem (Equation 2.2.48) to express s using the polynomial expansion coefficients of s and repeated convolution:

$$\begin{aligned}
\implies a(s(x)) &= \sum_{n=0}^{\infty} p_n s(x)^n && (2.2.140)\\
\implies \mathfrak{F}\{a(s(\,\cdot\,))\}(\omega) &= \mathfrak{F}\left\{\sum_{n=0}^{\infty} p_n s(\,\cdot\,)^n\right\}(\omega) && (2.2.141)\\
&= \sum_{n=0}^{\infty} p_n(\mathfrak{F}\{s\} \ast \overset{n}{\cdots} \ast \mathfrak{F}\{s\})(\omega) && (2.2.142)
\end{aligned}$$

This result becomes meaningful when we remark that repeated convolution in the spectral domain multiplies the span of the spectral domain by n. This is evident when we consider how convolution behaves with respect to zero and non-zero regions of its operands. Without loss of generality, we suppose a simple zero-centered spectral domain $\mathbf{\Omega} = [-\omega'', \omega'']$, meaning $\mathfrak{F}\{s\}(\omega) = 0\ \forall \boldsymbol{\omega} \notin [-\omega'', \omega'']$:

$$\begin{aligned}
(\mathfrak{F}\{s\} \ast \overset{n}{\cdots} \ast \mathfrak{F}\{s\})(\omega) &= \int_{\mathbb{R}} \mathfrak{F}\{s\}(\omega - \omega')(\mathfrak{F}\{s\} \ast \overset{n-1}{\cdots} \ast \mathfrak{F}\{s\})(\omega')d\omega' && (2.2.143)\\
&= 0\ \forall \omega \notin [-\omega''_n, \omega''_n] && (2.2.144)\\
\text{where } \omega''_1 &= \omega'' && (2.2.145)\\
\omega''_n &= \omega''_{n-1} + \omega'' && (2.2.146)\\
\implies \omega''_n &= n\omega'' && (2.2.147)
\end{aligned}$$

This provides an intuition for Equation 2.2.142: when the polynomial coefficients of the pointwise nonlinearity are zero above order n, the resulting signal has a spectral domain at most n times widened. For approximate discretization invariance, we want to minimize the spectral coefficients outside $\mathbf{\Omega}$. Given the analysis we have performed, we can see this can be encouraged in two ways.

The first strategy is to pick a pointwise nonlinearity whose polynomial coefficients decay faster. This is generally the case for smoother pointwise nonlinearities.

The second strategy is to perform antialiasing, which consists in temporarily rediscretizing from the original spatial sampling domain $\mathbf{X}_{\text{spl}}$ to a denser spatial sampling domain $\mathbf{X}'_{\text{spl}}$ that corresponds to a wider spectral domain $\mathfrak{V}(\mathbf{\Omega}') > \mathfrak{V}(\mathbf{\Omega}')$, applying the pointwise nonlinearity, and then rediscretizing back to the original spatial sampling domain $\mathbf{X}_{\text{spl}}$ while filtering excess spectral content appropriately. Given sufficient headroom to avoid significant aliasing, this is approximately discretization invariant.

2.2.25. The Laplacian pyramid

The Laplacian pyramid is a signal processing construction (Burt and Adelson, 1987; Adelson et al., 1984) that allows decomposing signals s into a sum of signals p_n according to the bands of the spectral domain they occupy. This decomposition is useful because it creates a sparse decomposition that localizes its information both spatially and spectrally. Our work leverages the Laplacian pyramid heavily for these properties (section 3.6, section 3.7). The Laplacian pyramid is constructed through the following recurrence relation:

$$p_0^{\text{low}} = s \tag{2.2.148}$$

$$p_n^{\text{low}} = \left(p_{n-1}^{\text{low}} * \phi_n^{\text{low}}\right) \tag{2.2.149}$$

$$p_n^{\text{band}} = p_{n-1}^{\text{low}} - p_n^{\text{low}} \tag{2.2.150}$$

The base case simply takes the original signal s as the starting point for recurrence p_0^{low}.

The recursive case takes the preceding signal p_{n-1}, and splits it into a low frequency part p_n^{low}, and a high frequency part p_n^{band}. Both parts contain spectral content that is complementary to each other.

We can see that each high-frequency part provides us with a focused slice of the spectral content of the signal. As we dive deeper into the layers of the pyramid, the frequency of this slice gets progressively lower.

We can also see by linearity of convolution, and by the fact that recursion always takes a difference from the original signal, that the sum of all high-frequency parts and the final low-frequency part must be equal to the original signal. The Laplacian pyramid is therefore a linear decomposition of the original signal:

$$s = p_m^{\text{low}} + \sum_n^m p_n^{\text{band}} \tag{2.2.151}$$

We also see that as we advance, the volume $\mathfrak{V}(\boldsymbol{\Omega}_n^{\text{low}})$ of the spectral domain of the low frequency part $\boldsymbol{\Omega}_n^{\text{low}} \approx \{\boldsymbol{\omega} | \widehat{\phi_n^{\text{low}}}(\boldsymbol{\omega}) \neq 0\}$ decreases. The great advantage of this is that it allows reducing the sampling density $\mathfrak{d}(\mathbf{X}_{\text{spl}}^n, \mathbf{X})$ while correctly representing each pyramid level, which reduces computational cost.

2.3. Normalizing flows

As surveyed in section 2.1, a large set of deep learning tasks consist in learning some form of mapping from inputs to outputs. In many cases, the specific pairings between inputs and outputs are of interest. But in some cases, the specific pairings generated by the mapping are not the focus — it is rather the structure of the mapping itself that is of interest.

Normalizing flows (Rezende and Mohamed, 2015; Kobyzev et al., 2020) are models that fit this second paradigm. They do not construct a mapping that respects input-to-output pairings explicitly — they rather translate inputs spread according to one distribution to outputs spread according to another distribution. The input-to-output pairings are contingent — they are implicitly learnt under this constraint on distributions.

The general idea behind this form of distribution modelling is that if we are successful in bridging a complex distribution to a simple one, we can benefit from the ease of manipulation of the simple distribution by translating to it from the complex distribution. This includes the ability to sample and evaluate.

We introduce notation to precisely designate the two distributions at play (Rezende and Mohamed, 2015; Kobyzev et al., 2020):

$p : \mathbf{Z} \to \mathbb{R}$**, called the latent distribution.** It is the simple distribution.

- It has a known expression for sample probability that is easy to evaluate.
- It can be sampled infinitely many times through random number generation.

$q : \mathbf{X} \to \mathbb{R}$**, called the observed distribution.** It is the complex distribution. We refer specifically to q as the true distribution, and to q_ϕ as the modelled distribution, parameterized by ϕ.

- It (q) does not have a known expression for sample probability.
- It (q) can be sampled finitely many times through a dataset of observations.

We also introduce notation for the maps that translate from one distribution to the other (Rezende and Mohamed, 2015; Kobyzev et al., 2020):

$n_\phi : \mathbf{X} \to \mathbf{Z}$**, called the normalizing map, parameterized by** ϕ**.**

- It brings samples from the observed distribution to the latent distribution:

$$n_\phi(\mathbf{x}_{\text{spl}}) \text{ with } \mathbf{x}_{\text{spl}} \sim q_\phi \approx q \text{ is distributed as } \mathbf{z}_{\text{spl}} \text{ with } \mathbf{z}_{\text{spl}} \sim p \tag{2.3.1}$$

- It enables approximately evaluating the probability of samples in the observed distribution:

$$q(\mathbf{x}) \approx q_\phi(\mathbf{x}) = p(n_\phi(\mathbf{x})) \left| \det \frac{\partial n_\phi}{\partial \mathbf{x}}(\mathbf{x}) \right| \tag{2.3.2}$$

$g_\phi : \mathbf{Z} \to \mathbf{X}$**, called the generative map, parameterized by** ϕ**.**

- It brings samples from the latent distribution to the observed distribution:

$$g_\phi(\mathbf{z}_{\text{spl}}) \text{ with } \mathbf{z}_{\text{spl}} \sim p \text{ is distributed as } \mathbf{x}_{\text{spl}} \text{ with } \mathbf{x}_{\text{spl}} \sim q_\phi \approx q \qquad (2.3.3)$$

  This enables approximately generating infinitely many samples in the observed distribution.

For the structure between the two maps to be feasible, the maps must be inverses of each other. This imposes strict restrictions on the way normalizing flow architectures are designed.

The ability to translate, generate, and evaluate the probability of samples explained above is always consistent with itself (meaning Equation 2.3.2 and Equation 2.3.3 always hold with regards to q_ϕ), but the modelled distribution is not necessarily consistent with the true distribution unless the model is optimal (meaning Equation 2.3.2 and Equation 2.3.3 only hold with regards to q if $q_\phi = q$).

This naturally brings us to the process of training and architecture design. A simple strategy is to maximize the log-likelihood of the observed dataset using Equation 2.3.2 (Rezende and Mohamed, 2015; Kobyzev et al., 2020):

$$\mathcal{L}_\phi(\mathbf{x}) = \log q_\phi(\mathbf{x}) = \log p(n_\phi(\mathbf{x})) + \log \left| \det \frac{\partial n_\phi}{\partial \mathbf{x}}(\mathbf{x}) \right| \qquad (2.3.4)$$

This learning objective is generally summed over a set of samples drawn from the dataset and optimized using gradient descent on ϕ. Flow architectures usually compose multiple layers together. Like the whole architecture, each layer has both a normalizing and generative map that are inverses of each other.

We use the following notation to designate intermediate latent spaces (Rezende and Mohamed, 2015; Kobyzev et al., 2020):

$\mathbf{z}^0 = \mathbf{x}$ **is the first intermediate latent space.** It is the observed space.

$\mathbf{z}^m = \mathbf{z}$ **is the last intermediate latent space.** It is the latent space.

We use the following notation to designate intermediate normalizing and generative maps (Rezende and Mohamed, 2015; Kobyzev et al., 2020):

$n^n_\phi(\mathbf{z}^{n-1}) = \mathbf{z}^n$ is the n-th normalizing layer.

Chaining all normalizing layers forms the full normalizing map:

$$n_\phi(\mathbf{x}) = n^m_\phi(n^{m-1}_\phi(\ \dots \ n^2_\phi(n^1_\phi(\mathbf{x})) \ \dots \ )) \tag{2.3.5}$$

$g^n_\phi(\mathbf{z}^n) = \mathbf{z}^{n-1}$ is the n-th generative layer.

Chaining all generative layers forms the full generative map:

$$g_\phi(\mathbf{z}) = g^1_\phi(g^2_\phi(\ \dots \ g^{m-1}_\phi(g^m_\phi(\mathbf{z})) \ \dots \ )) \tag{2.3.6}$$

For an architecture designed as a chain of flow layers, the training objective (Equation 2.3.4) can be expressed in a way that uses properties of the determinant and the logarithm to avoid matrix multiplication, which is both more computationally efficient and more numerically stable (Rezende and Mohamed, 2015; Kobyzev et al., 2020):

$$\mathcal{L}_\phi(\mathbf{x}) = \log p(n_\phi(\mathbf{x})) + \log \left| \det \frac{\partial n_\phi}{\partial \mathbf{x}}(\mathbf{x}) \right| \tag{2.3.7}$$

$$= \log p(n_\phi(\mathbf{x})) + \log \left| \det \left(\prod_{n=1}^{m} \frac{\partial n^n_\phi}{\partial \mathbf{z}^{n-1}}(\mathbf{z}^{n-1}) \right) \right| \tag{2.3.8}$$

$$= \log p(n_\phi(\mathbf{x})) + \sum_{n=1}^{m} \log \left| \det \left(\frac{\partial n^n_\phi}{\partial \mathbf{z}^{n-1}}(\mathbf{z}^{n-1}) \right) \right| \tag{2.3.9}$$

Chapter 3

Method

The main goal of this method is to address the difficulties that come with applying convolutional neural network architectures to tasks that involve signals whose discretization and signal bandwidth vary widely, both during training and inference. Our aim is to create an architecture that can seamlessly be adapted across signal discretization and signal bandwidth while maintaining great expressivity, parameter efficiency, ability to enforce desirable equivariances, computational cost, robustness, and training dynamics. The next few paragraphs provide an overview of the way we tackle this problem.

We allow our architecture to adapt to any signal discretization by requiring that every layer satisfy discretization invariance (subsection 2.2.18). This effectively consists in specifying our architecture in terms of operators that apply on continuous signals, which can later be implemented on discretized signals, according to the discretization given by the data. This means the model has no fixed discretization. In practice, this means a model trained for 32×32 images can be used directly with 16×16 images without needing to rediscretize the images to the architecture; it is the architecture that rediscretizes to the images. This also means reductions in discretization size lead to reductions in computational cost.

We address a blind spot in the literature of equivariant convolutional architectures, where the distinction between *discrete shift equivariance* and *continuous shift equivariance* tend to be disregarded (section 3.1).

We formulate a very generic layer based on the partial derivative operator that enables the implementation of desirable equivariances (section 3.2). For instance, invariance or equivariance to shift, rotation, and scale are all achievable within the framework we propose. We find that this approach leads to computationally efficient, expressive, and highly parameter-efficient layers.

We go beyond a simple formulation of a discretization invariant architecture and allow aggressively reducing computational cost when operating on low spectral volume signals. We propose a *Laplacian residual* structure that spectrally disentangles its layers in a way that allows lossless architecture compression. With this structure, a reduction in the size of the discretization implies a reduction in the number of layers required for evaluation. This comes without compromise on evaluation quality. (section 3.6)

We formulate a training augmentation that leverages the spectral disentanglement of *Laplacian residuals* by applying what we call *Laplacian dropout*. This has the effect of letting the architecture perceive signals whose spectrum is augmented. This has a profound impact on the robustness of the architecture to variations in signal bandwidth. (section 3.7)

We propose a simple data-driven weight initialization method that provides great learning dynamics for arbitrary network architectures. This allows rapid exploration of more exotic layer types without requiring symbolically derived initializations. (section 3.8)

We develop an extension to our method that allows dynamically allocating signal bandwidth over signals with nonuniform bandwidth using normalizing flows. This allows our model to adapt to signals such as point clouds. This also allows a form of lossy compression that is adjacent to compressed sensing. (section 3.9)

3.1. Continuous shift equivariant layers

We provide theoretical analysis that formalizes the distinction between *continuous* shift equivariance and *discrete* shift equivariance (subsection 2.2.22). We highlight the fact that *continuous* shift equivariance is a stronger symmetry than *discrete* shift equivariance, and that it comes with the requirement of discretization invariance (subsection 2.2.18), which is not trivial for pointwise operators.

We argue that the signals relevant to machine learning tasks are discretization invariant by nature, and that necessarily, there should not be a distinction in importance between a *single* continuous shift and a *single* discrete shift. While the set of all discrete shifts is *finite*, the set of all continuous shifts is *infinite*, so in fact, we should be very concerned with the correct implementation of continuous shift equivariance in our neural architecture.

The analysis we provide in subsection 2.2.23 exposes the necessary conditions for the creation of continuous shift equivariant discretized operators. We consider these constraints in the realization of every component of our method.

3.2. Steerable equivariant layers

Our architecture comprises steerable equivariant layers similar to Ruthotto and Haber (2020), Shen et al. (2020), and Jenner and Weiler (2021), in that they are defined as functions of the partial derivative operator. However, we always require that discretization invariance be satisfied, and we allow not only the application of a linear combination of a set of partial derivatives $\{\mathfrak{D}_{\mathbf{x}}^{\alpha^1}(s), \cdots, \mathfrak{D}_{\mathbf{x}}^{\alpha^p}(s)\}$, but the application of an arbitrary inner function h to a set of partial derivatives:

$$c(s) = h\left(\mathfrak{D}_{\mathbf{x}}^{\alpha^1}\{s\}, \cdots, \mathfrak{D}_{\mathbf{x}}^{\alpha^p}\{s\}, \theta\right) \tag{3.2.1}$$

The discretization of the partial derivative operator is discussed in depth in subsection 2.2.21. This can be recovered exactly through Equation 2.2.134. This process is computationally expensive, so it can be approximately recovered by substituting b_0 for a kernel that effectively softens the indicator function of Equation 2.2.82, with Gaussian kernels, as in Jenner and Weiler (2021), or with an even simpler kernel that implements the finite difference approximation, as in Shen et al. (2020). We chose this latter approach for its simplicity. Since first-order partial derivative operators can be repeatedly applied to yield higher-order partial derivative operators (Equation 2.2.61), and are closed on the signal class we consider (Equation 2.2.62), this first-order discretization is sufficient.

The discretization of the inner function varies with its nature. With linear inner functions, discretization is trivial, as discussed in subsection 2.2.19. With nonlinear inner functions, discretization is nontrivial. The method outlined to deal with nonlinear activation functions in section 3.3 applies identically here.

For our initial exploration, we simply restrict ourselves to layers defined as linear combinations of the zeroth and first-order partial derivatives, as explored by Ruthotto and Haber (2020); Shen et al. (2020); Jenner and Weiler (2021). This choice is guided by their simplicity, excellent expressivity and parameter efficiency, as claimed by Shen et al. (2020).

For discretization invariance to hold strictly, the padding should itself be discretization invariant. Padding with a repeated or mirrored version of the lattice allows this, however, it can be argued that this is not a reasonable inductive bias for most natural signals. Padding signals with an edge prolongation is a solution that leads to vanishing derivatives at the edges, which can be more desirable, although the vanishing rate is discretization dependent. Padding with zeros is also a possible solution, although it is more likely to lead to errors when pruning the Laplacian residual blocks we introduce in section 3.6. We chose the latter two in our initial exploration. We follow the same pattern in section 3.4.

3.3. Activations

As with all neural network architectures, nonlinearities are necessary for expressivity (Hornik et al., 1989). However, their usual implementation tends to violate the intended equivariances of the architectures they are a part of.

Activations generally violate discretization invariance and continuous shift equivariance. In section 3.1, we highlighted that this is a nontrivial consideration that is generally ignored in the formulation of convolutional architectures. In subsection 2.2.22, we provided an analysis showing that pointwise operators are neither necessarily continuous shift equivariant nor discretization invariant. In subsection 2.2.24, we showed that this is specifically the case for nonlinearities, which tend to induce discretization errors that can be approximately mitigated through two strategies.

The first strategy is to use smoother activation functions with polynomial expansions that decay faster tend to generate less of the offending spectral content that leads to discretization error through aliasing (subsection 2.2.14, Equation 2.2.142, Equation 2.2.144). For this reason, we favour the GELU (Hendrycks and Gimpel, 2016) activation function.

The second strategy is to apply antialiasing. By temporarily rediscretizing the signal to a higher density, we can provide sufficient spectral headroom to avoid significant aliasing. The amount of overhead provided can be expressed as a ratio between the original discretization density $\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X})$ and the antialiased discretization density $\mathfrak{d}(\mathbf{X}'_{\text{spl}}, \mathbf{X}')$. Greater ratios reduce this error but come at a computational cost. The antialiasing ratio should therefore be picked based on an empirical study that aims to strike a balance between these factors.

3.4. Local pooling

Convolutional network architectures generally use a form of local pooling to reduce the size of the discretization between layers. Again, we find that the most common implementations of local pooling are unsuitable for discretization invariance.

Local maximum pooling violates discretization invariance and continuous shift equivariance. The maximum over a window of a discretized signal generally does not match the true maximum of the continuous signal that is represented, because the point $\mathbf{x}$ where lies the true maximum over a spatial window $\mathbf{W}$ is not necessarily part of the corresponding sampled spatial window $\mathbf{W}_{\text{spl}}$, meaning it can be in disagreement with the point $\mathbf{x}_{\text{spl}}$ where lies the maximum over the sampled spatial window:

$$\underset{\mathbf{x}\in\mathbf{W}}{\operatorname{argmax}}\, s(\mathbf{x}) = \underset{\mathbf{x}_{\text{spl}}\in\mathbf{W}_{\text{spl}}}{\operatorname{argmax}}\, s(\mathbf{x}_{\text{spl}}) \iff \underset{\mathbf{x}\in\mathbf{W}}{\operatorname{argmax}}\, s(\mathbf{x}) \in \mathbf{W}_{\text{spl}} \tag{3.4.1}$$

This implies that local maximum pooling is not discretization invariant, nor continuous shift invariant. However, it discrete shift invariant to some degree. Because the windows span an integer number of samples, equivariance can only occur for integer shifts that are integer multiples of the window size, since integer shifts that are fractional multiples of the window size lead to applying the maximum over different windows.

Local average pooling violates discretization invariance and continuous shift equivariance. Local average pooling corresponds to an incorrect implementation of rediscretization that lowers sample density without properly truncating the spectrum of the original discretization, as outlined in Equation 2.2.124. This is because b'_0 is substituted for the box averaging filter, which leads to aliasing. This violates discretization invariance, which consequently violates continuous shift equivariance.

Correctly filtered rediscretization solves these issues. By instead applying Equation 2.2.124 with a correct or approximately correct filter kernel, we can retain discretization invariance and continuous shift equivariance.

For rediscretization, we choose an approximation that relies on truncated Gaussian kernels as they offer a good compromise between discretization error and computational efficiency. They are notably advantageous as they are separable, which can greatly reduce their computational footprint. Gaussian kernels take the same general form as multivariate Normal distributions:

$$\phi^{\text{gauss}}(\mathbf{x}) = (2\pi)^{-d/2} \det(\mathbf{\Sigma})^{-1/2} \exp\left(-\frac{1}{2}\mathbf{x}^\top \mathbf{\Sigma}^{-1}\mathbf{x}\right) \tag{3.4.2}$$

An arbitrary choice is needed to tie sigma to a cutoff frequency. We parameterize this in the case where the density is one and let $\mathbf{\Sigma} = \mathbf{I}\sigma^2$ with $\sigma = (2\pi)^{-1/2}$. This choice of sigma is motivated by the fact it provides good filtering in practice while minimally altering the signal when density is unchanged on regular lattice sampling patterns, with $\phi^{\text{gauss}}(0) = 1$ and $\phi^{\text{gauss}}(\mathbf{x}) \approx 0 \ \forall \mathbf{x} \in \mathbb{R}^d$ with $||\mathbf{x}|| \geq 1$. For other densities, we uniformly scale the filter. Intuitively, in the single-dimensional case, halving the sample density will double the filter radius; this scales the cutoff frequency as desired, and generalizes correctly in higher dimensions:

$$\sigma = (2\pi)^{-1/2} \left(\frac{\eth(\mathbf{X}_{\text{spl}}, \mathbf{X})}{\eth(\mathbf{X}'_{\text{spl}}, \mathbf{X})}\right)^{1/d} \tag{3.4.3}$$

Because Gaussian kernels never completely decay to zero, computation requires some form of truncation. We only consider the values of the kernel up to a radius 3σ.

Because we later (section 3.9) consider cases where the convolution does not take place on a lattice sampling pattern, but on a nonuniform sampling pattern, it is desirable to apply normalization on each discretized kernel so it retains unity gain at zero frequency. This avoids numerical stability issues.

Because it is sometimes possible for individual kernels to lead to near-zero values everywhere, an epsilon parameter $\epsilon = 10^{-9}$ is added to the normalization dividend to avoid the possibility of a division by zero.

3.5. Global pooling

For architectures that take a signal as an input, and yield a vector as an output, some form of global pooling is required, because flattening the discretization to a vector directly is not an option; its dimensionality would be variable. Necessarily, the form of global pooling used must also be discretization invariant. We see again that some typical solutions are unsuitable for discretization invariance.

Global maximum pooling violates discretization invariance and continuous shift invariance. The maximum of a discretized signal generally does not match the true maximum of the continuous signal that is represented, because the point $\mathbf{x}$ where lies the true maximum over the spatial domain $\mathbf{X}$ is not necessarily part of the sampled spatial domain $\mathbf{X}_{\mathrm{spl}}$, meaning it can be in disagreement with the point $\mathbf{x}_{\mathrm{spl}}$ where lies the maximum over the sampled spatial domain:

$$\underset{\mathbf{x}\in\mathbf{X}}{\operatorname{argmax}}\, s(\mathbf{x}) = \underset{\mathbf{x}_{\mathrm{spl}}\in\mathbf{X}_{\mathrm{spl}}}{\operatorname{argmax}}\, s(\mathbf{x}_{\mathrm{spl}}) \iff \underset{\mathbf{x}\in\mathbf{X}}{\operatorname{argmax}}\, s(\mathbf{x}) \in \mathbf{X}_{\mathrm{spl}} \tag{3.5.1}$$

This implies that global maximum pooling is not discretization invariant, nor continuous shift invariant. This can be approximately remedied by antialiasing, but this is needlessly computationally expensive.

Global average pooling solves these issues. The average of a discretized signal is necessarily $\widehat{s}(0)$, which is trivially always part of the spectral domain we consider. It is also very inexpensive to compute, as its discretization is simply the average on the sample set. We therefore choose this form of pooling throughout our method.

3.6. Laplacian residual blocks for compression through spectral disentanglement

Our architecture partitions the spectrum of the signal between layers using a novel *Laplacian residual* structure that exploits the strengths of both residual blocks (subsection 1.1.2, He et al. (2016a,b)) and Laplacian pyramids (subsection 2.2.25, Burt and Adelson (1987); Adelson et al. (1984)). This architectural block is illustrated in Figure 3.1.

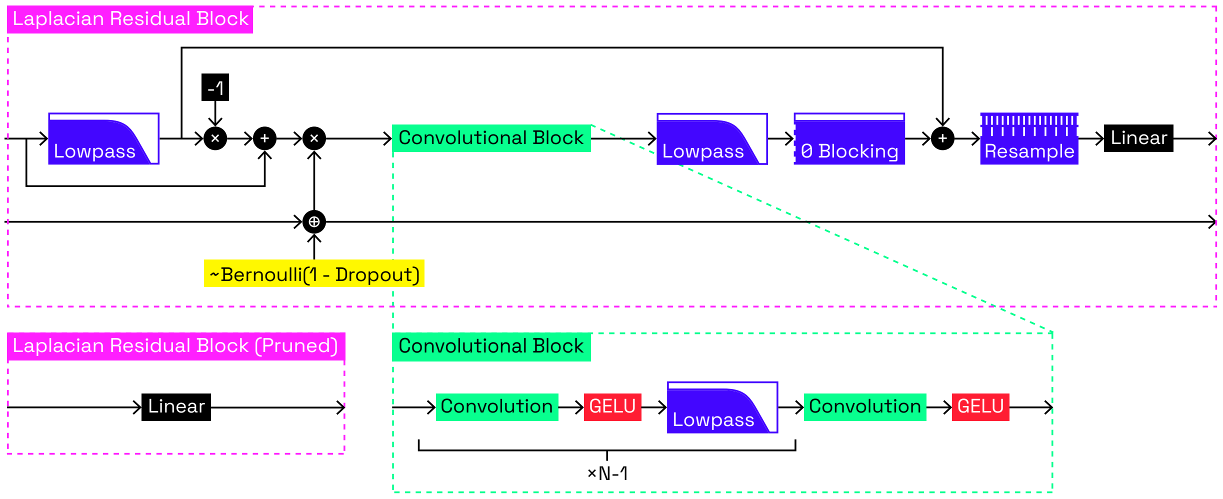


Fig. 3.1. Laplacian residual block diagram.

Laplacian residual blocks $r_n : (\mathbf{X} \to \mathbb{R}^{f_n}) \to (\mathbf{X} \to \mathbb{R}^{f_{n+1}})$ are composed together in a chain, and each contain a convolutional block $b_n : (\mathbf{X} \to \mathbb{R}^{f_n}) \to (\mathbf{X} \to \mathbb{R}^{f_n})$ that can contain multiple convolutions and activation functions. Like in residual blocks, each convolutional block has an additive skip connection. However, filtering operations are involved in the formulation of the connections. Laplacian residual blocks are formulated together recursively, very similarly to Laplacian pyramids:

$$r_0 = \mathbf{A}_0 s \tag{3.6.1}$$

$$r_n^{\text{low}} = \left(r_{n-1} \; * \; \phi_n^{\text{low}}\right) \tag{3.6.2}$$

$$r_n^{\text{band}} = r_{n-1} - r_n^{\text{low}} \tag{3.6.3}$$

$$r_n = \mathbf{A}_n \left(\left(b_n(r_n^{\text{band}}) * \phi_n^{\text{low}} * \phi^{\text{zero}}\right) + r_n^{\text{low}}\right) \tag{3.6.4}$$

The base case of Laplacian residuals and Laplacian pyramids (r_0 in Equation 3.6.1 and p_0 in Equation 2.2.148) both set the 0-th level to be the source signal $s : (\mathbf{X} \to \mathbb{R}^{f_0})$, optionally through a linear projection $\mathbf{A}_0 \in \mathbb{R}^{f_0 \times f_1}$.

Their recursive case both take the preceding signal and split it into a low-frequency part (r_n^{low} in Equation 3.6.2 and p_n^{low} in Equation 2.2.149) and a high-frequency part (r_n^{band} in Equation 3.6.3 and p_n^{band} in Equation 2.2.150). The main distinction between the two processes is that the Laplacian residual applies a convolutional block b_n to each high-frequency part r_n^{band} before passing it to the next Laplacian residual block r_n.

The output of the convolutional block is lowpass filtered by ϕ_n^{low} to respect the structure of the Laplacian pyramid, which always restricts the spectral domain according to $\widehat{\phi_n^{\text{low}}}$. Analogously to pooling operations in standard convolutional networks, this coarsens the feature scale available to the next layers of the network. The output of the convolutional block is then offset-blocked by ϕ^{zero} to ensure $(b_n(0) * \phi_n^{\text{low}} * \phi^{\text{zero}}) = 0$, which can otherwise occur because of bias parameters. The padding mode applied within the convolutional block also needs to be considered for this to hold. The sum of the output of the convolutional block and the residual low frequencies are added, and the resulting signal is projected through a matrix $\mathbf{A}_n \in \mathbb{R}^{f_n \times f_{n+1}}$ which allows changing dimensionality between blocks.

Rediscretization is performed just before the linear projection if applicable, as the spectral volume of r_n or p_n is now smaller than that of r_{n-1} or p_{n-1}, which allows reducing the sampling density to reduce computational cost.

Assuming filters that behave as binary masks, whenever the spectrum of a signal $\widehat{s}$ does not extend outside the spectrum of the lowpass filters $\widehat{\phi_{n'}^{\text{low}}}$ of the first consecutive layers $n' \in [1, n-1]$, the residual blocks $r_{n'}$ act strictly as linear projections given by $\mathbf{A}_{n'}$ because their convolutional block $b_{n'}$ receive no input, and therefore contribute no output. This means r_{n-1} is entirely determined by a linear projection of the input signal $\mathbf{A}_{n-1} \cdots \mathbf{A}_0 s$, and thus evaluating r_n does not require evaluating any of the previous residual blocks:

$$\widehat{s}\;(1 - \widehat{\phi_{n'}^{\text{low}}}) = 0 \;\; \forall 1 \leq n' < n \tag{3.6.5}$$

$$\implies r_1^{\text{low}} = \left(\mathbf{A}_0 s \; * \; \phi_1^{\text{low}}\right) = \mathbf{A}_0 s \tag{3.6.6}$$

$$\implies r_1^{\text{band}} = \mathbf{A}_0 s - \mathbf{A}_0 s = 0 \tag{3.6.7}$$

$$\implies r_1 = \mathbf{A}_1 \left(\left(b_1(0) * \phi_1^{\text{low}} * \phi^{\text{zero}}\right) + \mathbf{A}_0 s\right) = \mathbf{A}_1 \mathbf{A}_0 s \tag{3.6.8}$$

$$\vdots$$

$$\implies r_{n-1}^{\text{low}} = \left(\mathbf{A}_{n-2} \cdots \mathbf{A}_0 s \; * \; \phi_{n-1}^{\text{low}}\right) = \mathbf{A}_{n-2} \cdots \mathbf{A}_0 s \tag{3.6.9}$$

$$\implies r_{n-1}^{\text{band}} = \mathbf{A}_{n-2} \cdots \mathbf{A}_0 s - \mathbf{A}_{n-2} \cdots \mathbf{A}_0 s = 0 \tag{3.6.10}$$

$$\implies r_{n-1} = \mathbf{A}_{n-1} \left(\left(b_{n-1}(0) * \phi_{n-1}^{\text{low}} * \phi^{\text{zero}}\right) + \mathbf{A}_{n-2} \cdots \mathbf{A}_0 s\right) = \mathbf{A}_{n-1} \cdots \mathbf{A}_0 s \tag{3.6.11}$$

This means that at inference time, all consecutive residual blocks that target a region of the spectrum which is absent in the signal can be pruned without affecting the output of the network. Only the matrix product $\mathbf{A}_{n-1} \cdots \mathbf{A}_0$ must be handled; this can simply be precomputed once. In effect, this pruning scheme performs lossless compression on the network itself, which reduces computational cost. While this proof assumes perfectly sharp filters, we find its result is empirically accurate with softer filters in subsection 4.1.3.

This also offers the ability to perform lossy compression at inference to gain computational efficiency by effectively reducing the bandwidth of the signal perceived by the network. This even allows dynamic bandwidth changes in real-time, which can be useful to applications where a variable computation time budget is given, such as in robotics.

We note that discretization invariance is not a requirement for Laplacian residuals. They can be leveraged in standard architectures to create quasi-discretization invariant networks, which can be evaluated at any of their intermediate pooling resolutions while skipping unnecessary residual blocks.

If a series of Laplacian residual blocks are carefully designed to have filters whose spectral contours are shifted copies of each other placed at regular intervals on an exponential scale, scale equivariant networks can be implemented very computationally efficiently following the method outlined earlier (subsection 1.2.3, Worrall and Welling (2019); Sosnovik et al. (2019)). This notably allows for non-integer scale divisions, which is not feasible with Worrall and Welling (2019).

3.7. Laplacian dropout for robustness through spectral augmentation

Similarly to how *Laplacian residual* blocks allow suiting signals of varying bandwidth in a computationally efficient way, *Laplacian dropout* can emulate signals of varying bandwidth during training to encourage robustness. We show how this emerges from the structure of the Laplacian residual, which allows a unique interpretation of dropout (Srivastava et al., 2014).

Another conclusion that comes from Equation 3.6.11 is that zeroing out consecutive high-frequency parts $n' \in [1, n-1]$ of the residuals $r_{n'}^{\text{band}}$ is equivalent to letting the network perceive a signal with a spectrum filtered by $\prod_{0<n'\leq n}(1 - \widehat{\phi_{n'}^{\text{low}}})$. Laplacian dropout simply consists in randomly zeroing out blocks in this way. Figure 3.1 illustrates the network structure that allows this. The consecutiveness constrained is enforced by sampling independent masks d_n^{indep} from a Bernoulli random process with probability p_n of zeroing out, and chaining them from d_{n-1}^{chain} to d_n^{chain} using the logical or operator. r_n^{band} is then redefined to be multiplied by the chained mask:

$$d_n^{\text{indep}} \sim \text{B}(1 - p_n) \tag{3.7.1}$$

$$d_n^{\text{chain}} = d_n^{\text{indep}} \oplus d_{n-1}^{\text{chain}} \tag{3.7.2}$$

$$r_n^{\text{band}} = d_n^{\text{chain}}(r_{n-1} - r_n^{\text{low}}) \tag{3.7.3}$$

We find that this structure is highly efficient in improving the robustness of the network to signals of more restricted spectral domains, all without damaging the performance of the network on signals of wider spectral domains. We do not observe this behaviour on classical convolutional networks that are given equivalently augmented signals (subsection 4.1.2), which suggests that the success of this approach lies in the structure of Laplacian residual blocks. We hypothesize that it reduces the issue of vanishing gradients in the early layers of standard convolutional networks when given low bandwidth signals.

3.8. Weight initialization

Rather than using a classical weight initialization analytically derived from the variance properties of layers (subsection 1.3.1, LeCun et al. (2002); He et al. (2015); Glorot and Bengio (2010)), we use a simple data-driven initialization scheme (subsection 1.3.2, Bengio et al. (2006); Krähenbühl et al. (2015); Mishkin and Matas (2015)).

For every layer $f_n(s, \theta_n^1 \dots \theta_n^k)$, each parameter tensor θ_n^p is initialized from an isotropic Normal distribution with variance exponentially parameterized by ϑ_n^p:

$$\theta_n^p \sim \mathcal{N}(\exp(\vartheta_n^p)) \tag{3.8.1}$$

Starting at $n = 1$, the initialization log variances ϑ_n^p are optimized for all p until the variance of the output of the layer $\text{Var}(f_n(s, \theta_n^1 \dots \theta_n^k))$ is sufficiently close to one. This is defined by thresholding. Optimization then continues for layer $n + 1$. All weights are reinitialized at every optimization step. AdamW (Kingma and Ba, 2014; Loshchilov and Hutter, 2017) is used for this process, where signals s are drawn together in batches, and optimized by summing the signal-wise loss function:

$$\mathcal{L}_\vartheta = (\log \mathrm{Var}(f_n(s, \theta_n^1 \dots \theta_n^k)))^2 \tag{3.8.2}$$

For clarity, we provide pseudocode for our weight initialization algorithm in Algorithm 1.

Because weight initializations are specifically tuned to the data that is provided to the architecture, no assumptions are made on the statistical properties of the data.

Because this process removes the need for hand-derived weight initializations, layers that explore more exotic formulations can be prototyped easily, and layers whose weight initializations cannot be derived symbolically can be used, which is crucial to the adaptation of our method to nonuniform signals (section 3.9, subsection 3.9.8).

Because the weight initializations are optimized in logarithmic space, very large numerical ranges can be covered, and convergence tends to be fast. Because only one layer is introduced at a time, the compounding effects of exploding or vanishing variance are avoided, meaning the a priori values for ϑ_n^p can specify initialization magnitudes that are vastly incorrect without crashing the optimization process. This means very little needs to be known about the architecture to find weight initializations that yield good training dynamics.

Algorithm 1 Weight initialization

```
1:
                    ▽ iterate through targeted layers until all layers have satisfactory initializations.
2:  $n \leftarrow 1$
3:  while true do
                    ▽ reset the optimizer and link it to the initialization ranges of the targeted layer.
4:      optimizer.reset()
5:      optimizer.parameters $\leftarrow [\vartheta_n^1, \dots, \vartheta_n^k]$
                    ▽ iterate through batches of signals and optimize the initialization parameterization of the targeted layer until it is satisfactory.
6:      while true do
7:          $[s_0^1, \dots, s_0^b] \sim$ dataset
                    ▽ run a forward pass only up to the targeted layer while reinitializing all parameters $\theta_{1:n}$ according to $\vartheta_{1:n}$.
8:          for $n'$ from 1 to $n$ do
9:              for $p$ from 1 to $k$ do
10:                 $\theta_{n'}^p \sim \mathcal{N}(\exp(\vartheta_{n'}^p))$
11:             end for
12:             $[s_{n'}^1, \dots, s_{n'}^b] = [f_{n'}(s_{n'-1}^1, \theta_{n'}), \dots, f_{n'}(s_{n'-1}^b, \theta_{n'})]$
13:         end for
                    ▽ compute the mean log variance and check if it is satisfactory by thresholding; if so, go to the next layer, otherwise, continue iterating through batches on the same layer.
14:         variance_log = $\frac{1}{b} \sum_i \log \operatorname{var}(s_n^1)$
15:         if |variance_log| < variance_log_threshold then
16:             break
17:         end if
18:         optimizer.zero()
19:         optimizer.objective $\leftarrow$ variance_log$^2$
20:         optimizer.step()
21:     end while
22:     $n \leftarrow n + 1$
23:     if $n = m$ then
24:         break
25:     end if
26: end while
```

3.9. Adapting to nonuniform sampling patterns

The architecture we have formulated up to now addresses many of the challenges of machine learning tasks involving variety in signal discretization and signal bandwidth. However, it makes the common assumption that signal discretization follows lattice sampling patterns, and that signal bandwidth is uniform over the spatial domain. This is true of a wide range of signals, but not of all signals. A common example is the point clouds produced by 3D sensing and reconstruction systems. They can be conceptualized as signals if we consider that the signal discretization follows an irregular sampling pattern, and that signal bandwidth is locally defined according to point density.

Nonuniform signals are impractical to rediscretize on lattices. Applying architectures meant for uniform signals onto nonuniform signals is unwieldy because nonuniform signals are often very sparse. A capture of a 3D scene clusters its samples densely around the geometry of the objects in the scene, but samples are simply absent in empty space. Because the bounding box that contains the samples is mainly empty space, rediscretizing onto a lattice generally means allocating most of the lattice to empty space. This is illustrated in Figure 3.2

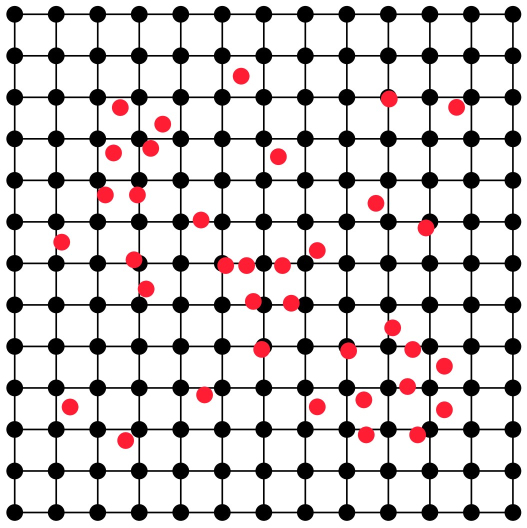


Fig. 3.2. Rediscretizing a nonuniformly sampled signal to a lattice directly.

This shows that rediscretizing a sparse nonuniform sampling pattern $\mathbf{X}_{\text{spl}}$ (in red) to a lattice sampling pattern $\mathbf{X}_{\text{lat}}$ (in black) directly has very poor sample efficiency. Most samples are wasted to empty space.

More formally, because the nonuniform sampling theorem (subsection 2.2.13, Equation 3.9.1) requires a minimal local sampling density based on the local spectral volume, and because a lattice sampling set $\mathbf{X}_{\text{lat}}$ necessarily has a global sampling density identical to its local sampling density everywhere, it must satisfy the local spectral volume everywhere with its global sampling density, leading to the worst case everywhere. This greatly inflates the number of required samples relative to an ideal nonuniform sampling set $\mathbf{X}_{\text{spl}}$, which is very computationally disadvantageous:

$$\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}|\mathbf{x}) \geq \mathfrak{V}(\mathbf{\Omega}|\mathbf{x}) \tag{3.9.1}$$

$$\mathfrak{d}(\mathbf{X}_{\text{lat}}, \mathbf{X}) = \mathfrak{d}(\mathbf{X}_{\text{lat}}, \mathbf{X}|\mathbf{x}) \ \forall \ \mathbf{x} \in \mathbf{X} \tag{3.9.2}$$

$$\implies \mathfrak{d}(\mathbf{X}_{\text{lat}}, \mathbf{X}) \geq \mathfrak{V}(\mathbf{\Omega}|\mathbf{x}) \ \forall \ \mathbf{x} \in \mathbf{X} \tag{3.9.3}$$

$$\implies \mathfrak{d}(\mathbf{X}_{\text{lat}}, \mathbf{X}) \geq \max_{\mathbf{x} \in \mathbf{X}} \mathfrak{V}(\mathbf{\Omega}|\mathbf{x}) \tag{3.9.4}$$

$$\implies |\mathbf{X}_{\text{lat}}| \gg |\mathbf{X}_{\text{spl}}| \tag{3.9.5}$$

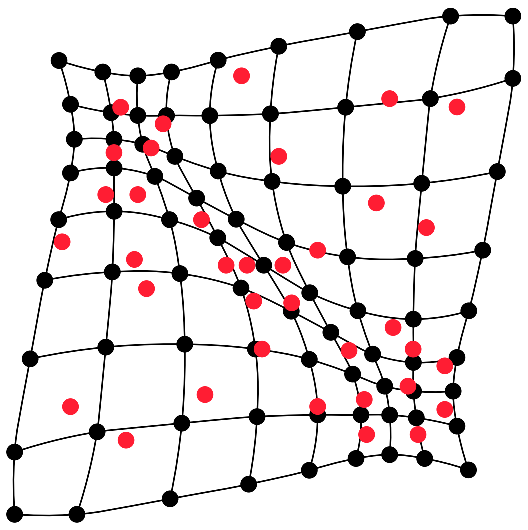
Nonuniform signals can be transformed into uniform signals for lattices. The novel solution we propose leverages normalizing flows (section 2.3) to transform nonuniform signals into uniform signals that can efficiently be rediscretized onto lattices. This is illustrated more intuitively in Figure 3.3a and Figure 3.3b.

We treat the nonuniform sampling set $\mathbf{X}_{\text{spl}}$ as coming from an observed distribution q, which can be mapped to a uniform sampling set $\mathbf{Z}_{\text{spl}}$ on a latent distribution p. We use the normalizing flows as a bridge that can be learnt between the two distributions. This solution allows for better sample efficiency in principle, because if the uniform sampling set $\mathbf{Z}_{\text{spl}}$ is close to truly uniform, rediscretizing to a lattice sampling set $\mathbf{Z}_{\text{lat}}$ can be done while respecting the sampling theorem with ideal sample count:

$$|\mathbf{Z}_{\text{lat}}| = |\mathbf{X}_{\text{spl}}| \tag{3.9.6}$$

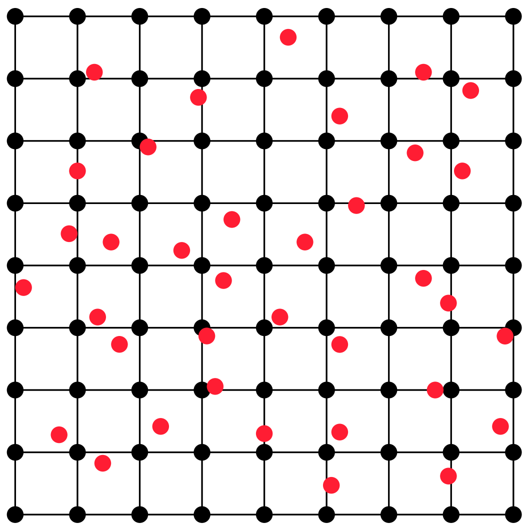
3.9.1. Formalizing density matching

While the solution we propose may seem intuitively valid, we wish to back it up more formally. This subsection provides a short proof of the correctness and sample efficiency of our method. This explicitly lays out the assumptions we make.



(a) View of the observed domain X. The nonuniform sampling pattern $\mathbf{X}_{\text{spl}}$ is defined here and shown in red. It lies on a distribution q modelled by the normalizing flows.

The inverse lattice sampling pattern $\mathbf{X}_{\text{lat}}$ is formed by the generative map and is shown in black. Its purpose is only to show how the normalizing flows bend space to match distributions.



(b) View of the latent domain Z. The uniform sampling pattern $\mathbf{Z}_{\text{spl}}$ is formed by the normalizing map and is shown in red. It lies on a uniform distribution p which is fixed by the normalizing flows.

The lattice sampling pattern $\mathbf{Z}_{\text{lat}}$ is defined here and is shown in black. Its purpose is to allow easy computation on the signal, as it can be rediscretized to the lattice while using fewer samples than needed otherwise. This is immediately apparent when comparing to Figure 3.2, which shares the same nonuniform sampling pattern, but uses far more samples to cover it.

Fig. 3.3. Rediscretizing a nonuniformly sampled signal to a lattice through normalizing flows.

Assumption 1. *We assume that the flow converges to an optimal solution which matches the true density of the sampling pattern:*

$$\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}|\mathbf{x}) \propto q(\mathbf{x}) = q_\phi(\mathbf{x}) = \left|\det \frac{\partial n_\phi}{\partial \mathbf{x}}(\mathbf{x})\right| \tag{3.9.7}$$

This can be reexpressed as a function of a point in the latent domain (because Jacobians of the normalizing and generative maps are necessarily inverses of one another):

$$\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}|g_\phi(\mathbf{z})) \propto q(g_\phi(\mathbf{z})) = q_\phi(g_\phi(\mathbf{z})) = \left|\det \frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right|^{-1} \tag{3.9.8}$$

Assumption 2. *We assume that the provided sampling pattern allocates its local sample density proportionally to the local spectral volume for all points in the observed domain:*

$$\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}|\mathbf{x}) \propto \mathfrak{V}(\boldsymbol{\Omega}|\mathbf{x}) \tag{3.9.9}$$

This means the discretization in the observed domain is not biased towards any specific region of the signal. This can be reexpressed for points in the latent domain:

$$\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}|g_\phi(\mathbf{z})) \propto \mathfrak{V}(\boldsymbol{\Omega}|g_\phi(\mathbf{z})) \tag{3.9.10}$$

Assumption 3. *We assume that the provided sampling pattern fulfils the local sample density criteria of Equation 2.2.78:*

$$\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}|\mathbf{x}) \geq \mathfrak{V}(\boldsymbol{\Omega}|\mathbf{x}) \tag{3.9.11}$$

This means the discretization in the observed domain correctly captures the signal.

Assumption 4. *We assume that the local spectral domain is isotropic and that any anisotropy introduced by the flow and the lattice rediscretization contributes negligible discretization error.*

This assumption allows us to establish a simple relationship between the local spectral volume of the latent domain and the local spectral volume of the observed domain by considering the local effect of the flow as a linear transformation that projects the spectral basis functions, which leads to an expression based on the Jacobian of the generative map:

$$\mathfrak{V}(\boldsymbol{\zeta}|\mathbf{z}) = \mathfrak{V}(\boldsymbol{\Omega}|g_\phi(\mathbf{z})) \left|\det \frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right| \tag{3.9.12}$$

Assumption 5. *We assume that the size of the sampling set in the latent space $|\mathbf{Z}_{spl}|$ is greater or equal to the size of the sampling set in the observed space $|\mathbf{X}_{spl}|$.*

This is a trivial assumption we can always satisfy by picking a sufficiently large discretization in the latent domain.

Assumption 1 and Assumption 2 imply that the normalizing flows assign probability density proportionally to the local spectral volume in the observed domain. It is therefore proportional to a function of the generative map:

$$\mathfrak{V}(\boldsymbol{\Omega}|g_\phi(\mathbf{z})) \propto \left|\det \frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right|^{-1} \tag{3.9.13}$$

Assumption 4 provides a relationship between the spectral volume in the latent domain and the spectral volume in the observed domain that can be rewritten by substituting in the previous result. The spectral volume in the latent domain appears proportional to two terms that are inverses of one another, meaning the local spectral volume in the latent domain is globally constant:

$$\mathfrak{V}(\boldsymbol{\zeta}|\mathbf{z}) \propto \left|\det \frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right|^{-1} \left|\det \frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right| \tag{3.9.14}$$

$$\implies \mathfrak{V}(\boldsymbol{\zeta}|\mathbf{z}) \text{ is constant} \tag{3.9.15}$$

Assumption 5 implies the discretization in the observed domain has at most as many degrees of freedom as the discretization in the latent domain, meaning it necessarily satisfies the sampling criteria:

$$\mathfrak{d}(\mathbf{Z}_{\text{spl}}, \mathbf{Z}|\mathbf{z}) \geq \mathfrak{V}(\boldsymbol{\zeta}|\mathbf{z}) \tag{3.9.16}$$

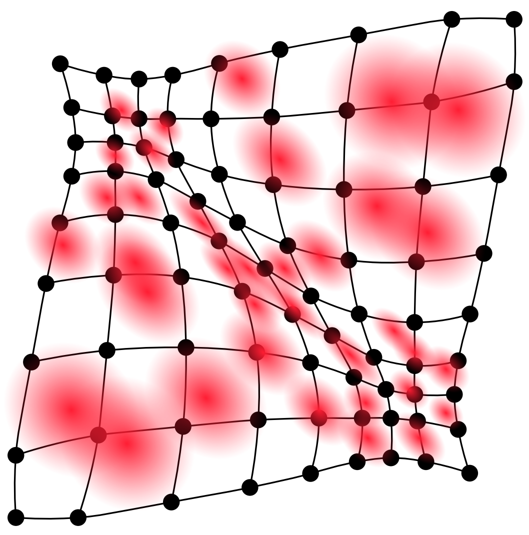
Assumption 4 implies the sampling criteria proved above is sufficient to declare the discretization in the latent domain is valid if the discretization in the observed domain is also valid, which is given by Assumption 3. Thus, the discretization of the signal in the latent domain, whose local spectral volume is uniform, correctly captures the signal in the observed domain, whose local spectral volume is nonuniform. While $\mathbf{Z}_{\text{spl}}$ is not a lattice, it is uniform, meaning it can efficiently be rediscretized onto a lattice $\mathbf{Z}_{\text{lat}}$ with $|\mathbf{Z}_{\text{lat}}| = |\mathbf{Z}_{\text{spl}}|$.

Interpretation of the result. We have successfully transformed a nonuniform signal processing problem into a uniform signal processing problem. We have improved upon the result of Equation 3.9.5, where instead of requiring a drastically higher sample count on the lattice discretization than on the ideal nonuniform discretization, $|\mathbf{X}_{\text{lat}}| \gg |\mathbf{X}_{\text{spl}}|$, we match it exactly with $|\mathbf{Z}_{\text{lat}}| = |\mathbf{X}_{\text{spl}}|$. This is an optimal result that cannot be improved further, because it would violate the sampling theorem. This means using lattices in this framework does not inflate computational cost through sample count, although it does come with additional complexity.

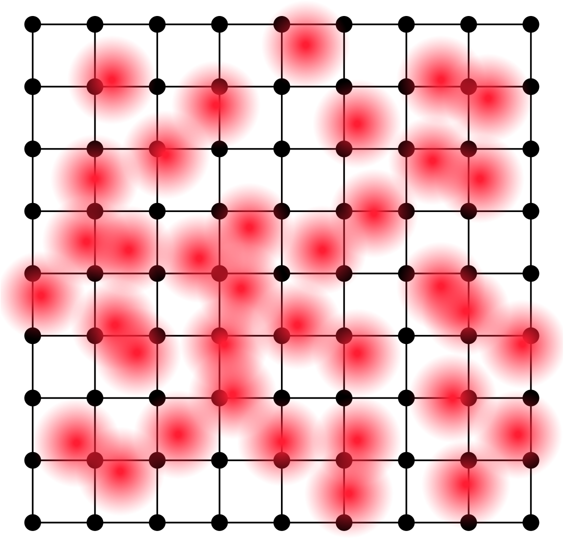
3.9.2. Rediscretization

We also have not yet established the specifics of rediscretization from $\mathbf{Z}_{\text{spl}}$ to $\mathbf{Z}_{\text{lat}}$. We address this in this subsection.

Rediscretization between a signal sampled on a set of uniform density $\mathbf{Z}_{\text{spl}}$ *that is not a lattice* to a signal sampled on a set $\mathbf{Z}_{\text{lat}}$ *that is a lattice* can be approximately implemented



(a) View of the observed domain X. The Gaussian filter kernels are applied in this domain indirectly through the generative map.



(b) View of the latent domain Z. The Gaussian filter kernels are applied in this domain directly.

Fig. 3.4. Equivalent views of Gaussian filtering in the latent domain used in rediscretization. In both figures, the envelope of the Gaussian filter kernels that correspond to each sample is illustrated. Because convolution applies in the latent domain, the kernels are effectively bent in the observed domain, becoming more compact as density increases. This is conceptually similar to the form of filtering we apply in *uniform* density interpolation (as seen in subsection 2.2.16), where the kernels *globally* become more compact with density. Instead, with *nonuniform* density interpolation, the kernels *locally* become more compact with density

by following the method outlined in subsection 2.2.17 and section 3.4. The nonuniform interpolation and rediscretization formulas are much more complex in truth (Landau, 1967; Marvasti, 2012), but the Gaussian approximation we propose is sufficient in practice. In effect, we perform a normalized convolution with the Gaussian kernel $\mathfrak{S}_{\mathbf{Z}_{\text{lat}}}\{\mathfrak{S}_{\mathbf{Z}_{\text{spl}}}\{s\} * \phi^{\text{gauss}}\}$. This is illustrated in Figure 3.4a and Figure 3.4b.

3.9.3. Local coordinate frames

We have examined the *implicit* structure learnt by flows. This is the way the normalizing and generative map expand and contract space to connect distributions, in this case, a nonuniform distribution on $\mathbf{X}$ and a uniform distribution on $\mathbf{Z}$.

We have not examined the *explicit* structure learnt by flows, however. This comprises the specific way in which the flows map exact values between $\mathbf{X}$ and $\mathbf{Z}$, and the ways this affects the discretization of the continuous operators on signals we use to build our method.

We address this through the concept of local coordinate frames, which we also define in this subsection.

Taking operators on continuous signals and adapting them as operators on discretized signals (subsection 2.2.18) requires that we take into account the way flows bend $\mathbf{X}$ into $\mathbf{Z}$. The map that the flows learn is only implicitly constrained to match densities, but the way the map explicitly connects points of the distributions is arbitrary. Because the learning process is stochastic, $\mathbf{X}$ and $\mathbf{Z}$ may be bridged in countless ways. Basic properties such as the orientation of the signal are not guaranteed to stay invariant, so we must somehow invert the effect of this arbitrary transformation.

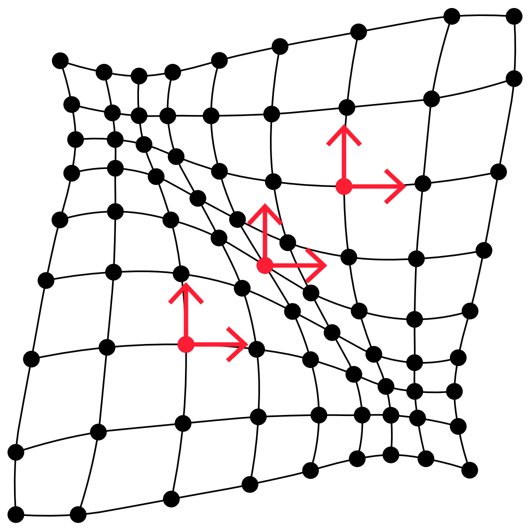
Because the operators we consider have a compact footprint, and because the maps expressed by the flows are invertible and can be designed to be generally well-behaved (as we later discuss in subsection 3.9.10), we can approximate the effect of this distortion by only considering the local change of frame of reference that is performed by the flows in linear terms.

The notion of this local frame of reference is encapsulated in the Jacobian $\partial g_\phi / \partial \mathbf{z}$ of the generative map. This expresses the link between partial derivatives in the latent domain $\partial \mathbf{z}$ to partial derivatives in the observed domain $\partial \mathbf{x}$. This contains the same information as the first-order Taylor expansion of $g_\phi(\mathbf{z})$ at $\mathbf{z}$:

$$g_\phi\Big(\mathbf{z} + \partial \mathbf{z}\Big) \approx g_\phi(\mathbf{z}) + \frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\partial \mathbf{z} \tag{3.9.17}$$

3.9.4. Adapting the partial derivative operator

In order to implement our steerable equivariant layers (section 3.2), we must adapt the discretization of the partial derivative operator to apply in $\mathbf{Z}$ while preserving the local frame of reference of $\mathbf{X}$. We do this by applying the concept of local coordinate frames introduced in subsection 3.9.3. This is illustrated in Figure 3.5a and Figure 3.5b.


(a) View of the observed domain X. The first-order partial derivatives are defined in this domain.

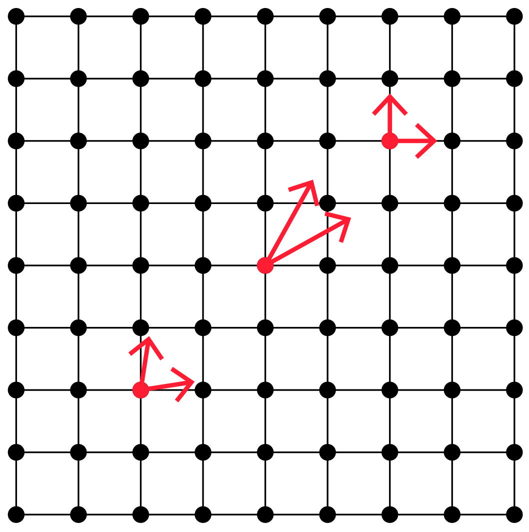

(b) View of the latent domain Z. The first-order partial derivatives are bent by the normalizing map.

Fig. 3.5. Equivalent views of partial derivatives in the observed domain.

Both figures illustrate pairs of partial derivatives whose original frame of reference is the observed domain. Because the latent domain is formed by bending the observed domain through the normalizing map, the derivatives are bent in the frame of reference of the latent domain. When we operate on the lattice in the latent domain $\mathbf{Z}_{\text{lat}}$, we must be mindful to invert this transformation.

We can derive an expression for the first-order partial derivatives $\mathfrak{D}_{\mathbf{x}}\{s\}$ in the original observed domain given the first-order partial derivatives $\mathfrak{D}_{\mathbf{z}}\{s\}$ in the bent latent domain by using the chain rule (Equation 3.9.17):

$$\frac{\partial s(g_\phi(\,\cdot\,))}{\partial \mathbf{z}}(\mathbf{z}) = \frac{\partial s}{\partial \mathbf{x}}(g_\phi(\mathbf{z}))\frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z}) \tag{3.9.18}$$

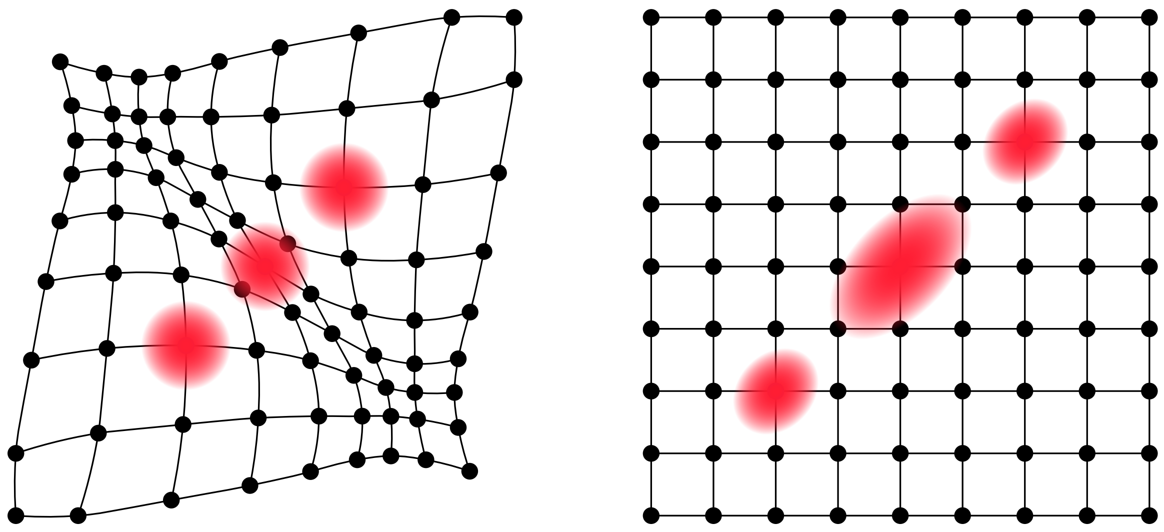
$$\implies \frac{\partial s}{\partial \mathbf{x}}(g_\phi(\mathbf{z})) = \frac{\partial s(g_\phi(\,\cdot\,))}{\partial \mathbf{z}}(\mathbf{z})\left(\frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right)^{-1} \tag{3.9.19}$$

We can then discretize this process. Once the signal has been bent from $\mathbf{X}_{\text{spl}}$ to $\mathbf{Z}_{\text{spl}}$, and rediscretized onto a lattice $\mathbf{Z}_{\text{lat}}$, we can follow the method outlined in section 3.2 to obtain $\mathfrak{D}_{\mathbf{z}}\{s\}$, and apply Equation 3.9.19 to obtain $\mathfrak{D}_{\mathbf{x}}\{s\}$.

This process is only discretization invariant if the inverse Jacobian term of Equation 3.9.19 does not introduce spectral content that cannot be captured appropriately, which depends on the smoothness of the flows. The inverse is always well-defined because the flows are invertible.

3.9.5. Adapting Gaussian filtering

In order to implement local pooling (section 3.4) and Laplacian residuals (section 3.6), we need a way to compute convolutions against a Gaussian kernel ϕ^{gauss} in $\mathbf{Z}$ while respecting its definition in $\mathbf{X}$. We do this by applying again the concept of local coordinate frames introduced in subsection 3.9.3. This is illustrated in Figure 3.6a and Figure 3.6b.



(a) View of the observed domain X. The Gaussian kernels are defined in this domain.

(b) View of the latent domain Z. The Gaussian kernels are bent by the normalizing map.

Fig. 3.6. Equivalent views of Gaussian filtering in the observed domain.

Both figures illustrate Gaussian filter kernels whose original frame of reference is the observed domain. Because the latent domain is formed by bending the observed domain through the normalizing map, the kernels are bent in the frame of reference of the latent domain. When we operate on the lattice in the latent domain $\mathbf{Z}_{\text{lat}}$, we must be mindful to invert this transformation.

We can derive an expression for this operation on continuous signals by performing a change of variables in the integral that defines convolution (Equation 2.2.32). This requires approximating the argument to the Gaussian using the change of coordinate frame laid out in Equation 3.9.17. This is a reasonable approximation because the mass of the kernel is mostly clustered around its origin, and because the flows act approximately as linear transformations on a sufficiently small scale:

$$(s * \phi^{\text{gauss}})(\mathbf{x}) = \int_{\mathbf{X}} s(\mathbf{x}')\phi^{\text{gauss}}(\mathbf{x} - \mathbf{x}')d\mathbf{x}' \qquad (3.9.20)$$

$$\mathbf{x}' = g_\phi(\mathbf{z}') \tag{3.9.21}$$

$$\mathbf{x} - \mathbf{x}' \approx g_\phi(\mathbf{z}) - g_\phi(\mathbf{z}') \tag{3.9.22}$$

$$\approx (\mathbf{z} - \mathbf{z}')\frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z}) \tag{3.9.23}$$

$$(s * \phi^{\text{gauss}})(g_\phi(\mathbf{z}')) \approx \int_{\mathbf{Z}} s(g_\phi(\mathbf{z}'))\phi^{\text{gauss}}\left((\mathbf{z} - \mathbf{z}')\frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right)\left|\det \frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right| d\mathbf{z}' \tag{3.9.24}$$

We can rewrite this expression to be directly rooted in the latent domain by formulating it as a somewhat unusual convolution. By manipulating the definition of the Gaussian kernel (Equation 3.4.2) and Equation 3.9.24, we can see that convolving in the observed domain with the spatially fixed Gaussian kernel ϕ^{gauss} with covariance matrix $\mathbf{\Sigma}$ is approximately equal to convolving in the latent domain with a spatially varying Gaussian kernel $\phi^{\text{gauss}}_{\mathbf{z}}$ with covariance matrix $\mathbf{\Sigma}(\mathbf{z})$:

$$(s * \phi^{\text{gauss}})(g_\phi(\mathbf{z}')) \approx (s(g_\phi(\,\cdot\,)) * \phi^{\text{gauss}}_{\mathbf{z}})(\mathbf{z}) \tag{3.9.25}$$

$$\mathbf{\Sigma}(\mathbf{z}) = \left(\left(\frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right)^{\top}\left(\frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z})\right)\right)^{-1}\mathbf{\Sigma} \tag{3.9.26}$$

We can then discretize this process. Once the signal has been bent from $\mathbf{X}_{\text{spl}}$ to $\mathbf{Z}_{\text{spl}}$, and rediscretized onto a lattice $\mathbf{Z}_{\text{lat}}$, we discretize the convolution against the Gaussian kernel using the scheme outlined in subsection 2.2.17 and section 3.4 adapted to implement the local covariance matrix (Equation 3.9.26).

This process is only discretization invariant if the absolute Jacobian determinant term of Equation 3.9.24 does not introduce spectral content that cannot be captured appropriately, which depends on the smoothness of the flows. The absolute value is not problematic because the Jacobian determinant cannot have a zero crossing. The generative map would otherwise not be invertible. The degree of discretization invariance also depends on the importance of the error in the approximate changes of variable of Equation 3.9.22 and Equation 3.9.23, which also depend on the smoothness of the flows.

Limitations. Because every Gaussian kernel has a unique covariance matrix, the truncation radius of every kernel is unique. The radius can be obtained through the singular value decomposition of $\mathbf{\Sigma}(\mathbf{z})$, but the details are spared here. The radius is unbounded, meaning improperly trained flows can produce an excessive radius requiring enormous kernel neighbourhood sizes.

Because vectorization on GPU hardware requires that memory allocation and computation be carried out according to the worst-case neighbourhood size over the spatial domain of every signal, and across every signal of the batch, this leads to severely inflated computational cost in most practical cases, and to memory allocation crashes with improperly trained flows, if a hard limit on the neighbourhood size is not implemented.

Because filtering occurs nonuniformly over the signal in the latent domain $\mathbf{Z}$ according to $\boldsymbol{\Sigma}(\mathbf{z})$, even given that the local spectral volume in the latent domain $\mathfrak{V}(\boldsymbol{\zeta}^n|\mathbf{z})$ is uniform at some layer n, the local spectral volume in the filtered latent domain $\mathfrak{V}(\boldsymbol{\zeta}^{n+1}|\mathbf{z})$ is necessarily nonuniform at the next layer $n+1$. This has the counterintuitive consequence of invalidating the guarantee that we can rediscretize $\mathbf{Z}^n_{\text{lat}}$ to $\mathbf{Z}^{n+1}_{\text{lat}}$ while reducing the sample count by a static ratio a such that $\mathfrak{d}(\mathbf{Z}^{n+1}_{\text{lat}}, \mathbf{Z}) = a\ \mathfrak{d}(\mathbf{Z}^n_{\text{lat}}, \mathbf{Z})$ after every filtering operation. We can only reduce the sample count uniformly such that the worst case is covered, meaning $\mathfrak{d}(\mathbf{Z}^{n+1}_{\text{lat}}, \mathbf{Z}) = \max_{\mathbf{z}} \mathfrak{V}(\boldsymbol{\zeta}^{n+1}|\mathbf{z})$, which is again vulnerable to improperly trained flows. Unlike a classical convolutional neural network, local pooling does not always reduce the discretization size. This is extremely computationally disadvantageous and makes the implementation of deep networks mostly infeasible.

These shortcomings were only recognized late in the development of this method. They severely compromise its viability, but we nonetheless choose to document it. We believe it may be possible to overcome these limitations by instead applying convolutions sparsely in the spectral domain, effectively combining spectral neural operators (subsection 1.2.4, Fanaskov and Oseledets (2022)) and compressed sensing (Donoho, 2006).

3.9.6. Adapting global pooling

In order to correctly implement all components of our model, we must also consider how global pooling is influenced by the change of coordinate frame from $\mathbf{Z}$ to $\mathbf{X}$. We apply again the concept of local coordinate frames introduced in subsection 3.9.3, this time conceptualized as a change of density.

We can adapt global pooling by following the change of variables used in subsection 3.9.5. As discussed in section 3.5, we can define global pooling on continuous signals as $\widehat{s}(0)$, which we can write out as a very simple inner product on $\mathbf{X}$. We then reexpress this on $\mathbf{Z}$. Note that b_0 denotes the 0 frequency *spectral basis* rather than the 0 offset *spatial basis*:

$$\widehat{s}(0) = \langle s, b_0 \rangle \tag{3.9.27}$$

$$= \int_{\mathbf{X}} s(\mathbf{x}) d\mathbf{x} \tag{3.9.28}$$

$$= \int_{\mathbf{Z}} s(g_\phi(\mathbf{z})) \left| \det \frac{\partial g_\phi}{\partial \mathbf{z}}(\mathbf{z}) \right| d\mathbf{z} \tag{3.9.29}$$

We discretize this by performing a weighted average over $\mathbf{Z}_{\text{lat}}$ governed by the determinant term.

This process is only discretization invariant if the absolute Jacobian determinant term of Equation 3.9.29 does not introduce spectral content that cannot be captured appropriately, which depends on the smoothness of the flows. The absolute value is not problematic because the Jacobian determinant cannot have a zero crossing. The generative map would otherwise not be invertible.

3.9.7. Learning to uniformize without breaking the loss

We have now defined how we may operate on nonuniform signals that have been bent into uniform signals by normalizing flows. We have not defined how we train the normalizing flows yet, however.

Since our goal is to transform the signal sampling distribution $\mathbf{X}_{\text{spl}} \sim q$ to a uniform distribution $\mathbf{Z}_{\text{spl}} \sim p$, we may think we can simply use the uniform distribution as our latent space and learn using the standard loss, but we would be mistaken: this causes the learning process to collapse.

We discuss why this is the case, and propose two solutions that avoid this situation. The first substitutes *direct* optimization of a uniform latent space for *indirect* optimization of a uniform latent space with the classic loss. The second uses a different loss altogether.

Learning on the standard loss (Equation 2.3.9) collapses with uniform latent spaces because of the step-like nature of the distribution. If we let the latent domain host a uniform distribution over $\mathbf{Z}$, the evaluation of the likelihood of the latent distribution for a point $\mathbf{x} \in \mathbf{X}$ can lead to one of two cases:

$$p(n_\phi(\mathbf{x})) = \begin{cases} \mathfrak{V}(\mathbf{Z})^{-1} & \text{if } n_\phi(\mathbf{x}) \in \mathbf{Z} \\ 0 & \text{otherwise} \end{cases} \tag{3.9.30}$$

This is problematic for the evaluation of the gradient $\partial \log q_\phi / \partial \phi$ of the training loss (Equation 2.3.9) relative to flow parameters, because an intermediate step involves computing the gradient $\partial \log p / \partial \mathbf{z}$ of the log-likelihood of the latent distribution (Equation 3.9.30) relative to the point in latent space

$$\frac{\partial \log p}{\partial \mathbf{z}}(n_\phi(\mathbf{x})) = \begin{cases} 0 & \text{if } n_\phi(\mathbf{x}) \in \mathbf{Z} \\ \text{undefined} & \text{otherwise} \end{cases} \tag{3.9.31}$$

The first case neither pushes nor pulls points towards the latent space when the points already land in $\mathbf{Z}$. The second case completely destroys the loss if the points land outside $\mathbf{Z}$.

Optimizing using a uniform latent distribution is not *directly* possible, but it is *indirectly* possible. We can achieve our goal if we instead optimize on a surrogate distribution that is well-behaved. After the fact, we can use the quantile function of the distribution to bring back the samples to a uniform distribution. This is the same process used in inverse transform sampling (Neumann, 1951; Devroye, 1986). In this work, we explore two surrogate latent distributions for optimization: a normal distribution, illustrated in Figure 3.7a, and a uniform distribution with Gaussian edges, illustrated in Figure 3.7b. Their derivations are conceptually simple but tedious, so they are spared here.

Another solution to this problem is to use another loss altogether. The sliced Wasserstein loss can be used for training normalizing flows without evaluating the likelihood of the latent distribution (Coeurdoux et al., 2022). We do not cover it in detail here.

3.9.8. Good learning dynamics with weight initialization search

Weight initialization of the flow model. Because a normalizing flow has to be learnt for every individual signal processed by the convolutional model, both at training and inference, good training dynamics are absolutely essential in order to reach a feasible computational cost. We tackle this in part by applying our weight initialization algorithm (section 3.8) to the normalizing flow model. The search algorithm is applied once before starting to train the convolutional model, and the weight initialization range parameters ϑ are reused to generate all subsequent reinitializations of the flow parameters ϕ when fitting a new sampling pattern.

Weight initialization of the convolutional model. Because the statistical properties of the input to the convolutional model depends on the learning process of the flow model, data-driven weight initialization is necessary for deeper networks, because weight initializations are difficult or impossible to derive symbolically in this setting.

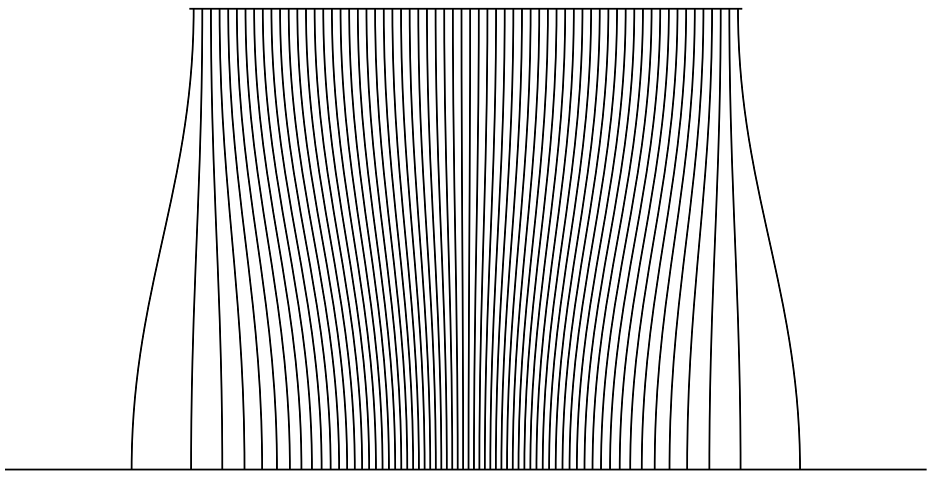
3.9.9. Good learning dynamics with normalizing layers

The need for impeccable learning dynamics motivates the use of batch normalization at the first layer of the flow model. This restricts the variance at the start of the training process to a consistent range, which plays well with the weight initialization algorithm.

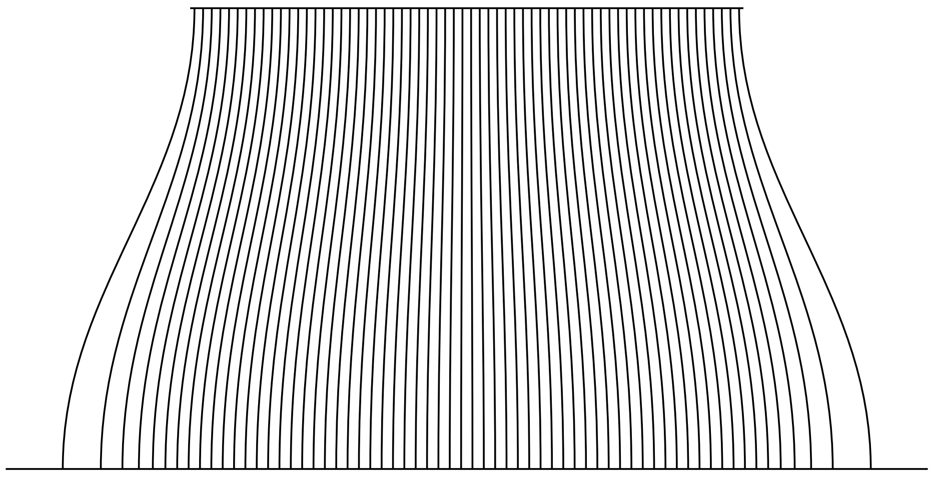
This normalization does not correspond to the form of batch normalization found in most neural networks. The key difference is that it is treated as a layer that deterministically learns once at the start of the flow learning process, and that operates independently for every batch element. We specifically use Gaussian normalization with an arbitrary covariance matrix. We also compare against a baseline using strictly a bounding box normalizer in our experiments (subsection 4.2.1).

3.9.10. Expressivity and smoothness with continuous flows

For our method to be effective, the flow architecture we use must satisfy a few conditions. Because point clouds and other types of nonuniformly sampled signals often contain separated clumps of points (section 3.9), the flow architecture must be able to effectively



(a) Gaussian. Parameters are $\sigma = 0.5$ for the Gaussian.



(b) Gaussian plateau. Parameters are $\sigma = 0.25$ for the Gaussian tail and $[-1, 1]$ for the span of the uniform plateau.

Fig. 3.7. Surrogate latent space mapping diagrams.

The diagrams above illustrate the mapping between the uniform domain (at the top) that is used to perform operations on the signal, and the surrogate domain (at the bottom) that is used during flow training. Each curved segment illustrates how regularly spaced points in the surrogate domain map to points in the uniform domain. This conveys at a glance how the mapping expands or contracts space. Both domains are drawn at identical scales, with the uniform domain spanning $[-1, 1]$.

model complex distributions with multiple modes. We demonstrate an extreme case of this in subsection 4.2.1.

Because training the flows becomes quickly computationally expensive, high parameter efficiency and good training dynamics are required.

Because of the conditions for discretization invariance of the partial derivative operator (subsection 3.9.4), of Gaussian filtering (subsection 3.9.5), of global pooling (subsection 3.9.6), and for correctness of the sampling scheme (subsection 3.9.1), the flow architecture must have a tendency to express well behaved, smooth mappings. Undifferentiable activation functions are therefore inadmissible.

For all of these reasons, we choose continuous flows (Weinan, 2017; Chen et al., 2018) using the GELU (Hendrycks and Gimpel, 2016) activation function.

3.9.11. Local spectral volume heuristic for nonuniform compression

The method we have formulated so far enables us to work on signals with nonuniform local spectral volume, but makes some assumptions on sampling patterns which restricts its usefulness.

Signals can have a *nonuniform sampling pattern* whose local sampling density corresponds to the local spectral volume. This is what enables us to train the flows.

Signals can also have a *lattice sampling pattern* while still having a nonuniform local spectral volume, however. In fact, most natural signals tend to have a sparse local spectral volume, meaning they can often be represented with a nonuniform sampling pattern whose size is smaller than that of the original lattice sampling pattern (Donoho, 2006). This can be leveraged by our method if we can formulate a way to train the flows without direct access to a nonuniform sampling pattern.

We next discuss the notion of sparsity more formally, and introduce a heuristic that roughly estimates the local spectral volume, and that allows forming a nonuniform sampling pattern that can train the flows.

Sparsity exists in the local spectral volume. Given a lattice discretization with global sampling density $\mathfrak{d}(\mathbf{X}_{\text{lat}}, \mathbf{X})$, any signal that it can represent necessarily has a local spectral volume $\mathfrak{V}(\boldsymbol{\Omega}|\mathbf{x}) \in [0, \mathfrak{d}(\mathbf{X}_{\text{lat}}, \mathbf{X})]\ \forall\ \mathbf{x} \in \mathbf{X}$ (Equation 2.2.79), meaning that a nonuniform discretization with a tightly fitted local sampling density $\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}|\mathbf{x}) = \mathfrak{d}(\mathbf{X}_{\text{lat}}, \mathbf{X})$ will always require fewer samples $|\mathbf{X}_{\text{spl}}| < |\mathbf{X}_{\text{lat}}|$ unless the local spectral volume is maximal everywhere, which is extremely rare in natural signals (Donoho, 2006).

Sparsity can be leveraged for compression. If we can somehow rediscretize $\mathbf{X}_{\text{lat}}$ to $\mathbf{X}_{\text{spl}}$ such that $\mathfrak{d}(\mathbf{X}_{\text{spl}}, \mathbf{X}|\mathbf{x}) = \mathfrak{V}(\boldsymbol{\Omega}|\mathbf{x})$, we should be able to compress the signal to use fewer samples, and we should then be able to use our method to adapt our convolutional architecture to the nonuniformly compressed signal.

We would ideally like to exactly compute the local spectral volume at every point of $\mathbf{X}_{\text{lat}}$, and use this as a distribution to form $\mathbf{X}_{\text{spl}}$.

We have not exactly developed the notion of local spectral volume in subsection 2.2.13, but we note that we can think of it as a measure of the span of a signal decomposition that is localized both in space and spectrum. We can construct it using wavelets or tuned banks of filters, as in Knutsson et al. (1994), or as in the Laplacian pyramid (subsection 2.2.25, Burt and Adelson (1987); Adelson et al. (1984)).

We instead go for a much simpler heuristic $H\{s\}(\mathbf{x})$ to facilitate implementation. We use $L2$ norm of the Hessians of the signal s, where we also take the $L2$ norm over feature channels $k \in [1, f]$.

$$H\{s\}(\mathbf{x}) = \left(\sum_{\substack{\alpha \in \{0,2\}^d \\ \sum_j^d \alpha_j = 2}} \sum_{k=1}^{f} \mathfrak{D}_{\mathbf{x}}^{\alpha}\{s_k\}^2(\mathbf{x}) \right)^{1/2} \tag{3.9.32}$$

The intuition behind this choice of heuristic is that it produces high values when the *spatial rate of change* of the *spatial rate of change* of the signal is high. This tends to occur neither on constantly sloped areas, nor at the center of edges, but at either extremities of edges. Knowledge of this information tends to allow reconstructing the edge itself and the area that surrounds it. Similar heuristics have been very commonly used for edge and keypoint detectors in signal processing and computer vision (Torre and Poggio, 1986; Ziou et al., 1998).

We rediscretize by leveraging the structure we have constructed with normalizing flows (subsection 3.9.2) to allow nonuniform antialiasing at a lower computational cost.

We form a *reduced nonuniformly distributed* sample set in *observed* space $\mathbf{X}_{\text{spl}}$ by sampling points from a multinomial distribution without replacement that assigns a probability proportional to the heuristic $H\{s\}(\mathbf{x}_{\text{lat}})$ to every point of the *lattice* sample set in *observed* space $\mathbf{x}_{\text{lat}} \in \mathbf{X}_{\text{lat}}$. We lower the number of samples through this operation: $|\mathbf{X}_{\text{spl}}| \ll |\mathbf{X}_{\text{lat}}|$.

We pay no attention to rediscretization of the new set at this stage, as it only is instrumental to training the flows. We train the flows to to map the *reduced nonuniformly distributed* sample set in *observed* space $\mathbf{X}_{\text{spl}}$ to the *reduced uniformly distributed* sample set in *latent* space $\mathbf{Z}_{\text{spl}}$. We only then consider rediscretization.

Pushing only the *reduced nonuniformly distributed* sample set in *observed* space $\mathbf{X}_{\text{spl}}$ through the trained normalizing map and rediscretizing onto the *lattice* sample set in *latent* space $\mathbf{Z}_{\text{lat}}$ would result in aliasing unless the conditions for lossless compression are met, because no antialiasing is performed. Samples have simply been removed by sampling through the heuristic.

Pushing instead the *lattice* sample set in *observed* space $\mathbf{X}_{\text{lat}}$ through the trained normalizing map and rediscretizing onto the *lattice* sample set in *latent* space $\mathbf{Z}_{\text{lat}}$ allows for antialiasing.

Whereas applying antialiasing in the *observed* domain would have required bending the filter kernels (subsection 3.9.5), applying antialiasing in the *latent* domain through this strategy does not involve bending the filter kernels, which is more computationally efficient.

Lossless compression is only guaranteed if the whole process results in satisfying the local sampling theorem through the flow (subsection 3.9.1). This depends on the quality of the heuristic $H\{s\}(\mathbf{x}_{\text{lat}})$ as a surrogate for the true local spectral volume $\mathfrak{V}(\mathbf{\Omega}|\mathbf{x}_{\text{lat}})$, on the number of samples of $|\mathbf{X}_{\text{lat}}|$ and $|\mathbf{Z}_{\text{lat}}|$, on the sparsity of s, on the convergence of the flow training process, and on the accuracies of the various approximations used in rediscretization. Lossless compression is therefore generally hard to guarantee.

Lossy compression is admissible in principle. Because the rediscretization technique we implement by leveraging normalizing flows provides a reasonable degree of antialiasing, harsh artifacts are mitigated. Because the compression adapts to the local information density on the signal, it can be more effective than fixing a global information density on the signal.

Chapter 4

Experiments

We perform a set of experiments to assess the success of our method at handling the challenges of discretization in deep learning tasks on signals. We cover most main components of our method and provide comparisons against prior work in this field. We first cover the central aspects of our method in section 4.1, then cover its adaptation form *uniform* signals to *nonuniform* signals in section 4.2.

4.1. Testing the core of the method

We verify the adaptability of our model to different discretizations with *Laplacian residuals* in subsection 4.1.1. We measure robustness to spectral degradation with *Laplacian dropout* in subsection 4.1.2. We finally empirically validate the correctness of *Laplacian residual pruning* in subsection 4.1.3.

Datasets. In all experiments within section 4.1, we perform a signal classification task on the CIFAR10 (Krizhevsky et al., 2009) and FashionMNIST (Xiao et al., 2017) image classification datasets. CIFAR10 consists of 60K 32×32 colour images split evenly between 10 classes, while FashionMNIST contains 70K 28×28 monochrome images divided identically.

Architectures. We compare our model against the popular ResNet18 (11M parameter) and ResNet101 (44M parameter) residual convolutional network architectures, along with the larger ViT-B/16 (87M parameter) (Dosovitskiy et al., 2020) vision transformer. We also compare a Fourier Neural Operator (18M parameter) network that is partially discretization invariant (Li et al., 2020).

Figure 4.1 shows the architecture used on CIFAR10 without pruning. The configuration used for FashionMNIST omits the first Laplacian residual block, and its first linear layer maps from $\mathbb{R}^1 \to \mathbb{R}^{32}$ rather than $\mathbb{R}^3 \to \mathbb{R}^{32}$.

Laplacian residual pruning is always performed except when we explicitly test the effect it has in subsection 4.1.3.

Laplacian dropout is enabled when we test it in subsection 4.1.2, and the probability of each independent dropout connection is set to 0.5. This effectively creates an exponentially shaped distribution of spectral augmentations. The equivalent augments that are used for comparison on other methods follow this same distribution.

We note that we have not performed hyperparameter search because of time constraints, which puts us at a severe disadvantage, as the architectures we compare to have been refined through years of research. We also note that the architectures we compare to have vastly higher parameter counts; this ranges from a factor of 14 to 140. Parameter counts that account for Laplacian residual pruning at inference are shown in table Table 4.1.

Table 4.1. Architecture parameter counts including Laplacian residual pruning at inference.

Model	Discretization	CIFAR10	FashionMNIST
ResNet18	-	11M	11M
ResNet101	-	44M	44M
ViT-B/16	-	87M	87M
FNO	-	18M	18M
Ours	32×32	0.75M	-
	28×28	0.75M	0.74M
	24×24	0.74M	0.74M
	20×20	0.74M	0.73M
	16×16	0.71M	0.71M
	12×12	0.69M	0.68M
	8×8	0.59M	0.59M

Training. All models are trained for 100 epochs on CIFAR10, and 30 epochs on Fashion-MNIST. Our model was optimized using AdamW (Kingma and Ba, 2014; Loshchilov and Hutter, 2017), a learning rate of 10^{-3}, a batch size of 128, betas $\beta_1 = 0.9$ and $\beta_2 = 0.999$, weight decay of 10^{-2}, and cosine annealing to zero learning rate in 100 epochs. Gradient skipping on NaNs and infinities along with 2-norm clipping thresholded at 10 were active, but all of these measures were monitored and never took effect during training. Competing models were optimized using SGD, a learning rate of 0.1, a batch size of 128, and identical cosine annealing.

For our weight initialization algorithm, the absolute log variance threshold was set to 10^{-1}. Training used AdamW (Kingma and Ba, 2014; Loshchilov and Hutter, 2017) with a learning rate of 10^{-1}, identical batch size and betas, no weight decay, and no annealing. Gradient skipping on NaNs and infinities (but not clipping) were active, but were monitored and never took effect during weight initialization. This applies to all experiments across chapter 4.

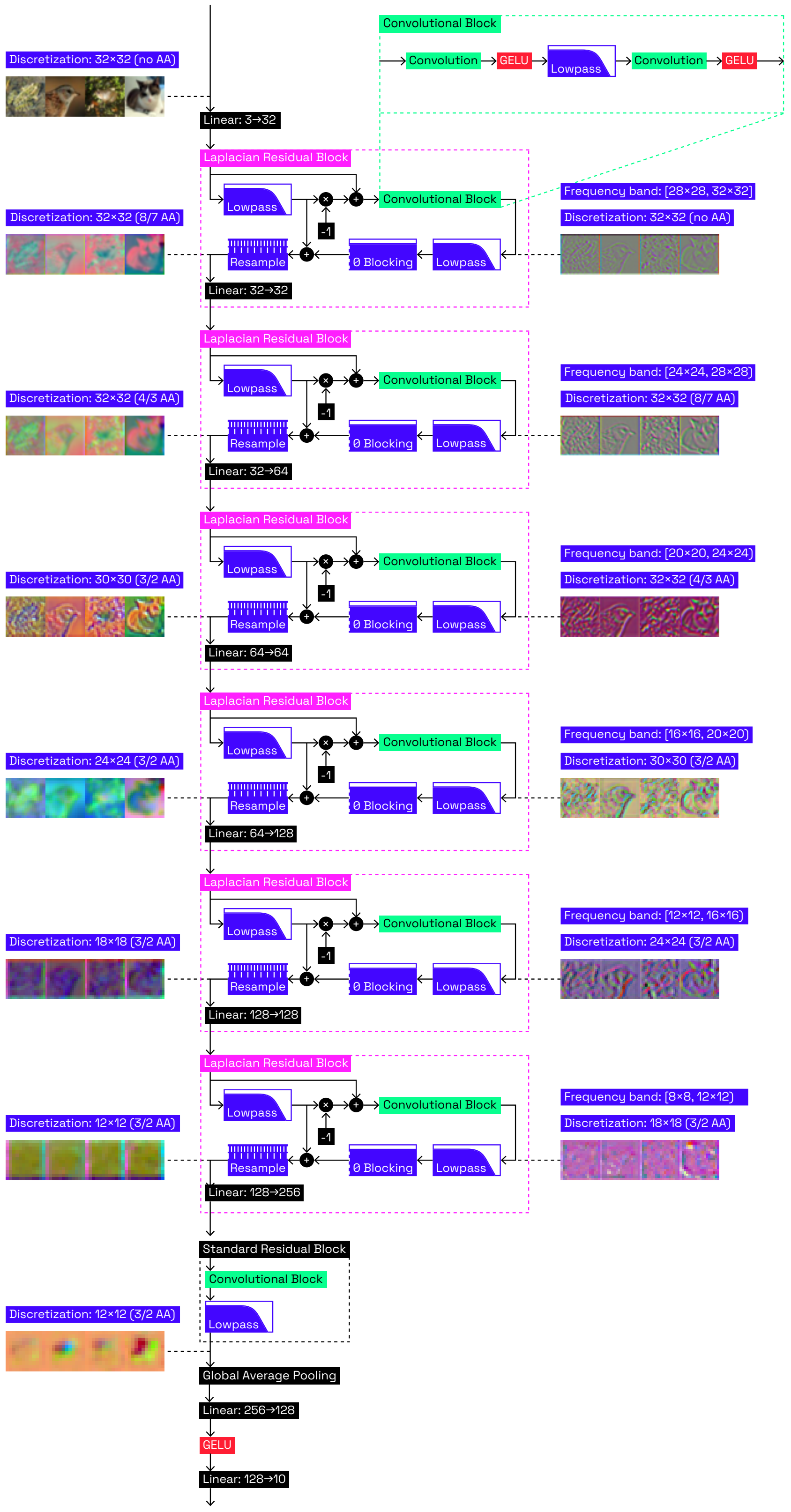

Convolutional Block
Convolution
GELU
Lowpass
Convolution
GELU
Discretization: 32×32 (no AA)
Linear: 3→32
Laplacian Residual Block
Lowpass
Convolutional Block
-1
Resample
0 Blocking
Lowpass
Frequency band: [28×28, 32×32]
Discretization: 32×32 (no AA)
Discretization: 32×32 (8/7 AA)
Linear: 32→32
Laplacian Residual Block
Frequency band: [24×24, 28×28]
Discretization: 32×32 (8/7 AA)
Discretization: 32×32 (4/3 AA)
Linear: 32→64
Laplacian Residual Block
Frequency band: [20×20, 24×24]
Discretization: 32×32 (4/3 AA)
Discretization: 30×30 (3/2 AA)
Linear: 64→64
Laplacian Residual Block
Frequency band: [16×16, 20×20]
Discretization: 30×30 (3/2 AA)
Discretization: 24×24 (3/2 AA)
Linear: 64→128
Laplacian Residual Block
Frequency band: [12×12, 16×16]
Discretization: 24×24 (3/2 AA)
Discretization: 18×18 (3/2 AA)
Linear: 128→128
Laplacian Residual Block
Frequency band: [8×8, 12×12]
Discretization: 18×18 (3/2 AA)
Discretization: 12×12 (3/2 AA)
Linear: 128→256
Standard Residual Block
Convolutional Block
Lowpass
Discretization: 12×12 (3/2 AA)
Global Average Pooling
Linear: 256→128
GELU
Linear: 128→10

Fig. 4.1. Network diagram.

Training was hardware accelerated using individual NVIDIA A100, A6000, A5000, A4000 and RTX8000 GPUs. Experiments shown in section 4.1 were completed in roughly 24 hours.

4.1.1. Discretization invariance

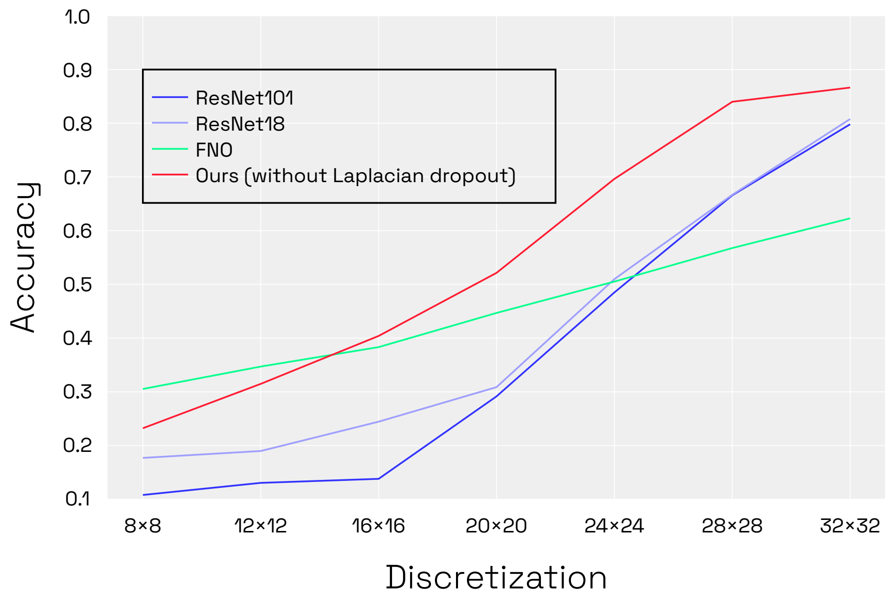
This experiment evaluates the ability of different architectures to handle *inference* discretizations that do not match their *training* discretization. Models are trained at the native dataset discretization (32×32 or 28×28 depending on the dataset), then, at inference, images are downsampled to a smaller discretization (28×28 if applicable, 24×24, 20×20, 16×16, 12×12, 12×12). The models are forced to accept this discretization. No training augmentations nor Laplacian dropout is used.

Interpretation of the results. As expected, Figure 4.2a and Figure 4.3a show that standard convolutional perform very poorly as they are not discretization invariant. Vision transformers are simply excluded as they cannot be rediscretized. Since our model can be rediscretized seamlessly, it outperforms both residual networks. It also notably outperforms Fourier neural operators, which only display discretization invariance with weak robustness. Our model dominates this experiment while undercutting every other method in parameter count by a factor of 14 to 140, which is consistent with the parameter efficiency claimed by Shen et al. (2020).

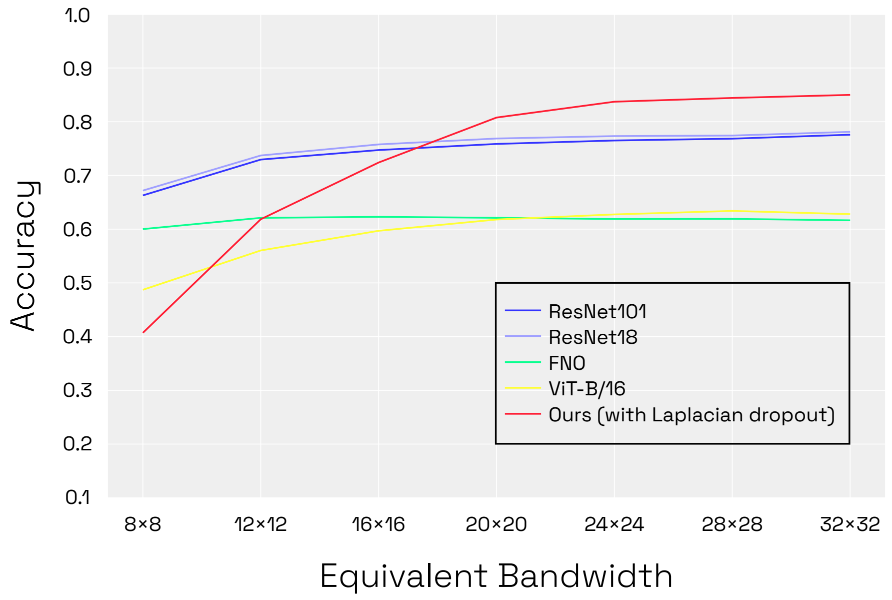
4.1.2. Bandwidth robustness

This experiment evaluates the ability of architectures to develop robustness to reduced bandwidth signals. This includes signal spectrum degradation both during training and inference. Like the previous setup, at inference time, reduced bandwidth signals are given to the models (with spectral content equivalent to what can be represented by 28×28 if applicable, 24×24, 20×20, 16×16, 12×12, or 8×8 discretizations). However, the competing models are allowed to run inference at their training discretization (32×32 or 28×28); the images are only filtered, not rediscretized. This represents a more practical scenario that benefits these methods, as this overcomes their lack of discretization invariance. Our method uses Laplacian dropout in this setup, and other methods perform a data augmentation that reproduces the effect of Laplacian dropout by randomly filtering the images according to the same distribution. This is done by randomly downsampling to a lower resolution, and upsampling back to the original resolution using bilinear interpolation. This allows a fair comparison for training and inference sets that vary in bandwidth.

Interpretation of the results. With the simpler FashionMNIST dataset, (Figure 4.3b) all architectures are generally able to develop robustness to the variation in signal bandwidth,

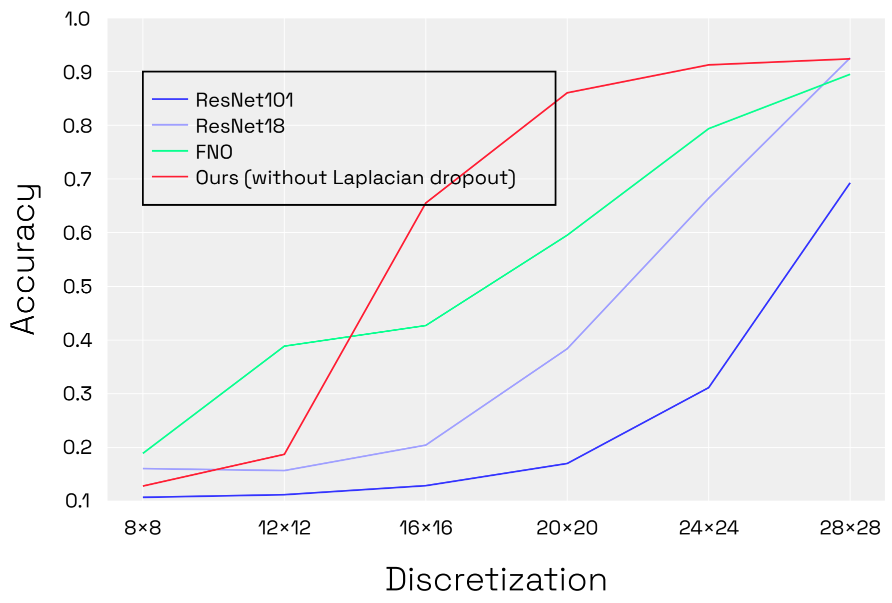


(a) CIFAR10 trained on 32×32 only, evaluated with various discretizations.

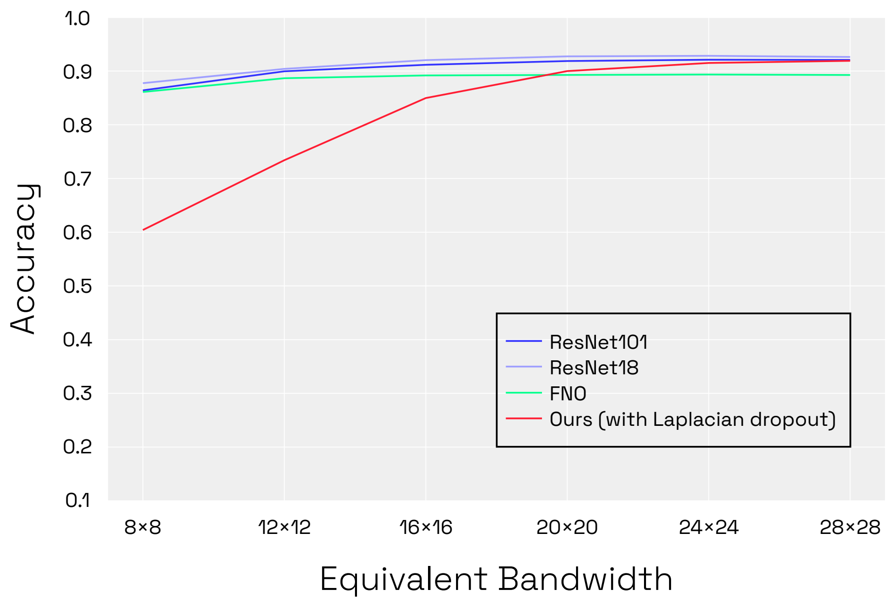


(b) CIFAR10 trained on a wide range of signal bandwidths, evaluated at the native discretization.

Fig. 4.2. Evaluation of our method and prior work on the Fashion CIFAR10 dataset.



(a) FMNIST trained on 28×28 only, evaluated with various discretizations.



(b) FMNIST trained on a wide range of signal bandwidths, evaluated at the native discretization.

Fig. 4.3. Evaluation of our method and prior work on the Fashion MNIST dataset.

however, our model displays slightly weaker performance for low bandwidth signals. With the more complex CIFAR10 dataset (Figure 4.2b), however, other models display a poor accuracy that plateaus at very low bandwidth, meaning they are unable to leverage the information present in high-bandwidth signals. In contrast, our method achieves significantly higher performance at higher resolutions and progressively degrades as information is removed from the signal, meaning it is able to learn effectively with signals of various bandwidths while maintaining robustness.

Overall, all architectures significantly improve. This means Laplacian dropout is effective, whether implemented in an architecture that possesses Laplacian residuals, or whether implemented as an equivalent data augmentation and applied to other architectures. However, the results suggest Laplacian dropout has a stronger affinity for networks that have Laplacian residuals.

4.1.3. Pruning correctness

In section 3.6, we showed theoretically that a network architecture built from Laplacian residual blocks can be pruned without affecting its output. This requires that the spectrum of the input signal does not overlap with the spectrum associated with the pruned residual blocks. Equation 3.6.11 is derived from the assumption that filters are perfectly sharp, which is not true with our implementation, which relies on Gaussian kernels (section 3.4).

However, if one block less is pruned relative to the prescription of the theoretical result, its conclusion applies: the output of a pruned network is nearly identical to the output of a network that was not pruned. Figure 4.5 illustrates the pruning process in detail for the network trained in subsection 4.1.2, then pruned for two different discretizations. Starting at the first block, all consecutive blocks that have no overlap with the frequency range captured by the discretization are pruned, except the last non-overlapping block, which is kept to account for signal leakage caused by soft filters. This is shown in the 20×20 example. For signal discretizations that fall at or below the middle of block frequency ranges, setting aside small discretization sizes, pruning can be performed more aggressively, as shown with the 14×14 example.

Interpretation of the results. Figure 4.4 shows that the performance degradation caused by pruning is negligible; this is illustrated by evaluating the model trained in subsection 4.1.2 both with pruning (full line) and without pruning (dashed line). Excluding the smallest discretization size, the largest performance degradation observed is 0.25%. At the smallest size, this climbs up to 3.85%. This variation in fidelity is somewhat expected, as the frequency ratios between successive layers are not constant, meaning they each produce varying amounts of frequency bleed-through. All experimental results discussed in subsection 4.1.1 and subsection 4.1.2 use this form of pruning during evaluation.

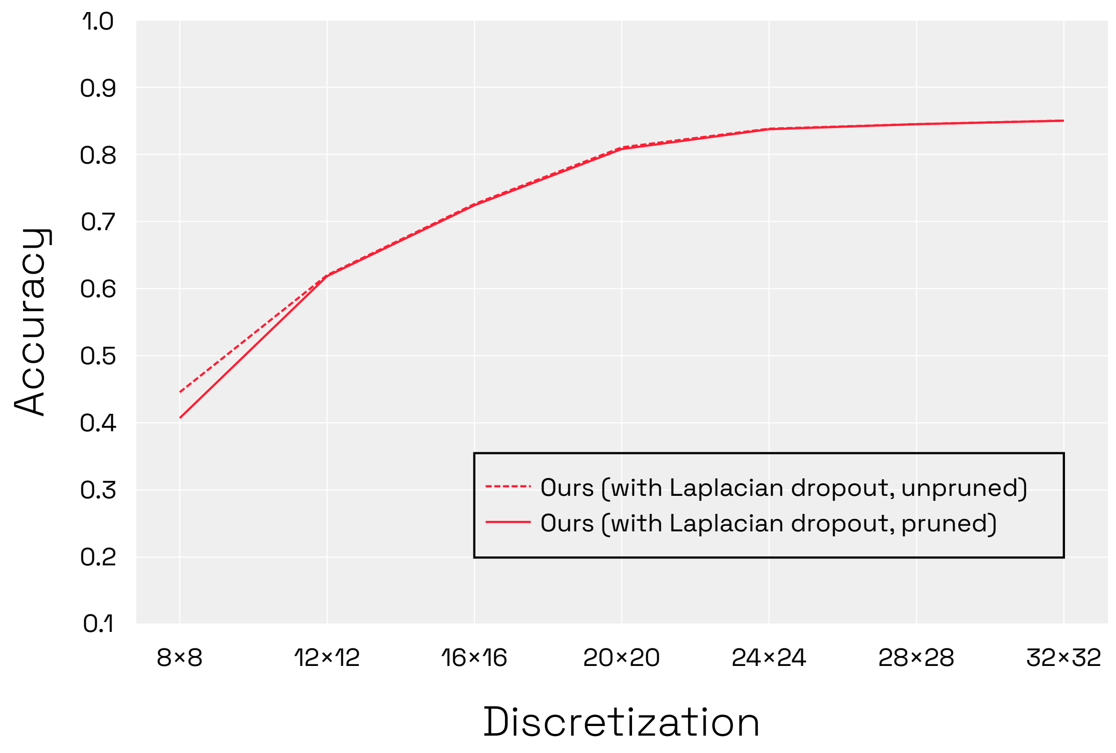
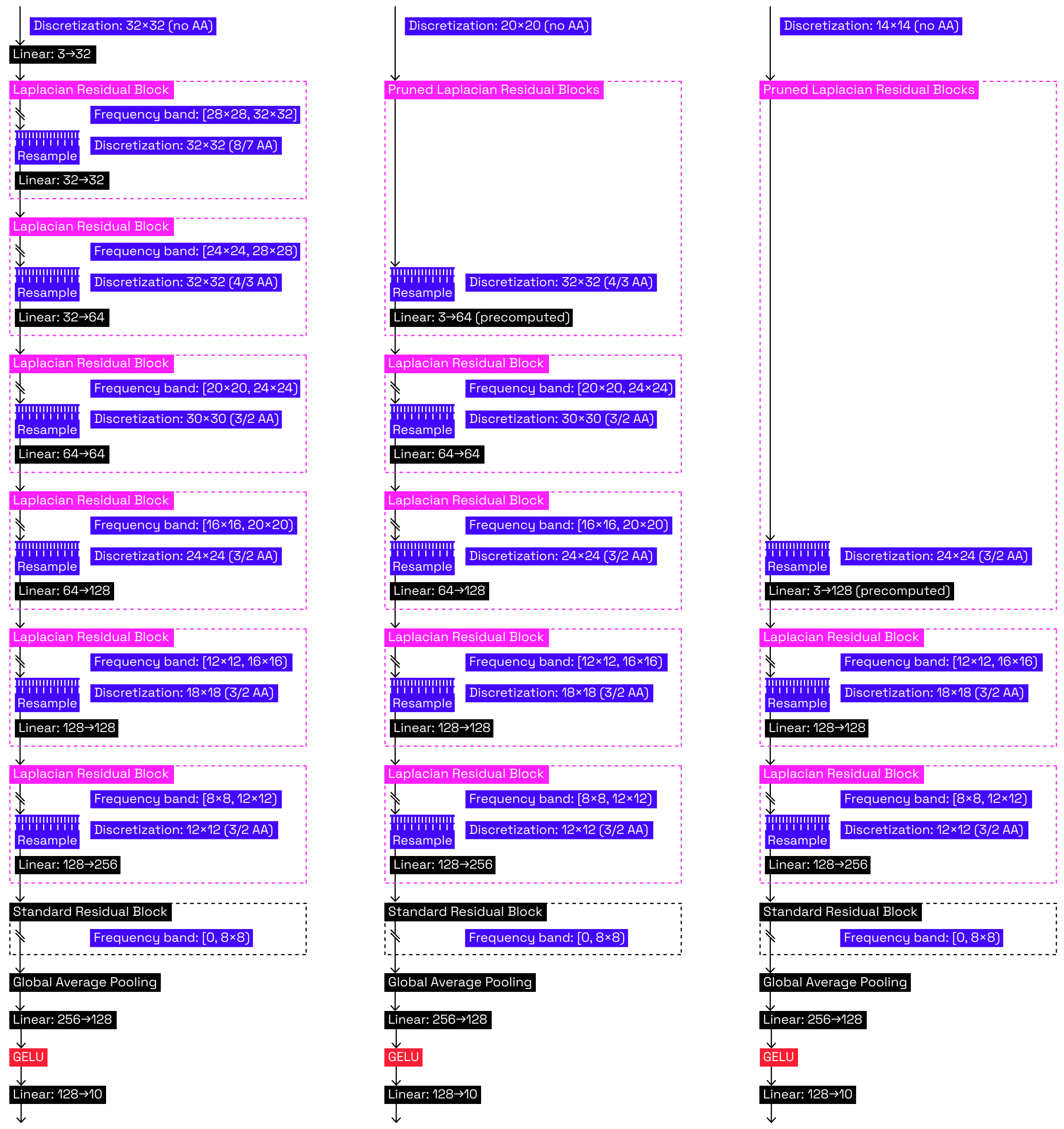


Fig. 4.4. Evaluation of our method with and without pruning on the CIFAR10 dataset.



Discretization: 32×32 (no AA)
Linear: 3→32
Laplacian Residual Block
Frequency band: [28×28, 32×32]
Resample
Discretization: 32×32 (8/7 AA)
Linear: 32→32
Laplacian Residual Block
Frequency band: [24×24, 28×28]
Resample
Discretization: 32×32 (4/3 AA)
Linear: 32→64
Laplacian Residual Block
Frequency band: [20×20, 24×24]
Resample
Discretization: 30×30 (3/2 AA)
Linear: 64→64
Laplacian Residual Block
Frequency band: [16×16, 20×20]
Resample
Discretization: 24×24 (3/2 AA)
Linear: 64→128
Laplacian Residual Block
Frequency band: [12×12, 16×16]
Resample
Discretization: 18×18 (3/2 AA)
Linear: 128→128
Laplacian Residual Block
Frequency band: [8×8, 12×12]
Resample
Discretization: 12×12 (3/2 AA)
Linear: 128→256
Standard Residual Block
Frequency band: [0, 8×8]
Global Average Pooling
Linear: 256→128
GELU
Linear: 128→10
Discretization: 20×20 (no AA)
Pruned Laplacian Residual Blocks
Resample
Discretization: 32×32 (4/3 AA)
Linear: 3→64 (precomputed)
Discretization: 14×14 (no AA)
Pruned Laplacian Residual Blocks
Resample
Discretization: 24×24 (3/2 AA)
Linear: 3→128 (precomputed)

Fig. 4.5. Network pruning diagram.

Crossed-out connections inside Laplacian residual blocks stand for the filters and convolutional blocks that are omitted in the drawing. The diagram follows the same conventions as Figure 4.1. **Left:** the full network used, shown on the native 32×32 discretization. **Center:** an example showing the network pruned for a 20×20 discretization. **Right:** an example showing the network pruned for a 14×14 discretization.

4.2. Adapting the method to nonuniform bandwidth signals

We provide a proof of concept that demonstrates the capability of our adaptation scheme based on normalizing flows (section 3.9) in subsection 4.2.1. We perform ablations to show the importance of weight initialization in subsection 4.2.2. We perform similar ablations on flow normalization in subsection 4.2.3. We highlight that the combination of both flow normalization and weight initialization is crucial. We finally test our compressed inference method (subsection 3.9.11) in subsection 4.2.4 by evaluating a model trained in subsection 4.1.2.

Datasets. In all experiments aside from the last, we use a very simple synthetic dataset for signal classification, which is designed to present a best-case scenario comparison for our method, where the nonuniform sampling pattern is extremely sparse. Figure 4.6 provides an illustration of a few samples from the synthetic dataset.

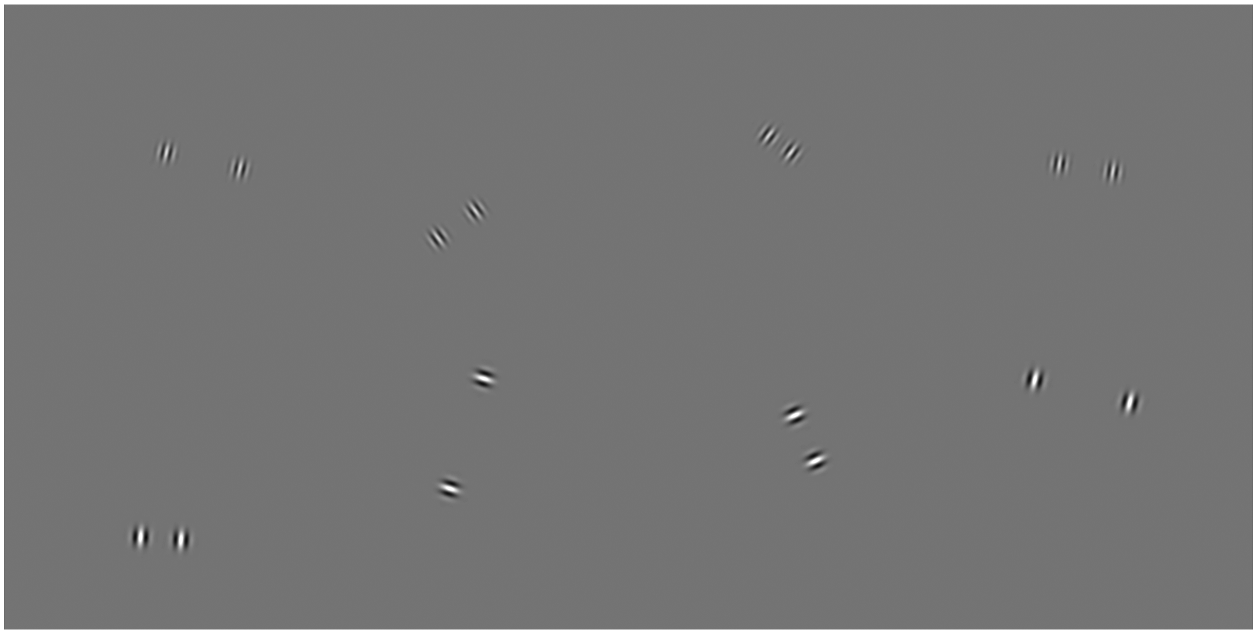


Fig. 4.6. Synthetic dataset rendered at high resolution.

This image shows eight dataset examples, which each occupy a square area. The four in the top half belong to the high-frequency class, and the four in the lower half belong to the low-frequency class. For the purpose of visualization, the discretization used here is based on a lattice sampling pattern and has a very high sampling density. For the experiments, the discretization is a nonuniform sampling pattern whose density is concentrated on the wavelets. Notice how the wavelets are always aligned between pairs. This attenuates anisotropy issues in our method.

The signals in the synthetic dataset are pairs of one-dimensional spatial basis functions (Equation 2.2.86) projected to two-dimensional space, which belongs to one of two classes that are constructed on subtly different spectral domains. The task consists in identifying these two classes. The spatial basis functions are each centered randomly according to a Gaussian with a large variance. The signals are sampled nonuniformly with a density that corresponds to a combination of two Gaussians with small variance each centered on a spatial basis function. This makes the sampling pattern extremely sparse, relative to a uniform sampling pattern which matches the peak density of the distribution everywhere, which is precisely the property that can be leveraged by our method (subsection 3.9.1).

To make the exploitation of sparsity not just an efficiency advantage, but a critical requirement for solving the task, we design the content of the signals to have challenging bandwidth requirements. In all experiments, the size of the first lattice discretization is fixed to 24×24. This very tight sample budget requires efficient bandwidth allocation to satisfy the sampling theorem (Equation 2.2.79). With a good bandwidth allocation, as with our adapted method, it is possible to correctly discretize the signal with a good margin of error. With a classical method which allocates bandwidth uniformly, the task is impossible to solve if the entire signal is to be discretized because severe aliasing occurs (subsection 2.2.14).

We eliminate other confounding factors in the task by constraining the dataset to a very small size (16384 signals).

We also design the dataset in a way that reduces the anisotropy issues that are not perfectly covered by our method (Assumption 4). This is done by making the frequency of the spatial basis functions aligned between pairs. This is illustrated more clearly in Figure 4.6.

Architectures. The convolutional architecture has sufficient but extremely limited capacity for the task, with only 394 parameters. It consists of a linear projection to 8 feature channels, a filtering layer, a convolutional layer and activation, another filtering layer for antialiasing, a global average pooling layer, a linear layer mapping to 16 feature channels, and another linear layer mapping to two feature channels.

The flow architecture relies on continuous flows (subsection 3.9.10), with two hidden layers each with 32 feature channels, for a total of 1250 parameters. We Euler integrate it over 16 steps. In the case where normalization is not ablated, this is preceded by a deterministic arbitrary covariance Gaussian normalizer (subsection 3.9.9). We optimize it with AdamW (Kingma and Ba, 2014; Loshchilov and Hutter, 2017) in only 16 steps, with a learning rate of 10^{-3}, with both betas at 0.9, and no weight decay.

With the standard loss (section 2.3, Equation 2.3.9), the latent space is a surrogate Gaussian plateau (subsection 3.9.7) with $\sigma = 0.125$, a uniform domain $[-1, 1]^2$, and a rediscretization cropped to contain only the uniform part of the distribution.

With the sliced Wasserstein loss (subsection 3.9.7), the loss is computed with 64 uniformly distributed slices using the $L2$ norm computed in the latent domain, where a uniform distribution is optimized directly.

With the baseline case using a standard uniform bandwidth approach without flows, the rediscretization of each signal is set to fit the bounding box of each signal's sampling pattern.

Training. Training is only performed over a single epoch, as the task is extremely simple, and the architecture is small.

We optimize using AdamW (Kingma and Ba, 2014; Loshchilov and Hutter, 2017), a learning rate of 10^{-2}, a batch size of 16, betas $\beta_1 = 0.9$ and $\beta_2 = 0.999$, weight decay of 10^{-3}. Otherwise, all other parameters are identical to those used in section 4.1.

Table 4.2 shows the test accuracies achieved with the different configurations. The tables interpretation is discussed next.

Table 4.2. Evaluation of our method on synthetic sparse bandwidth signal classification dataset.

Method	Loss	Ablation	Test Accuracy
Uniform			50.07%
Uniform		WI (C)	50.07%
Adaptive	Standard		99.76%
Adaptive	Standard	WI (C)	99.65%
Adaptive	Standard	WI (F)	72.50%
Adaptive	Standard	WI (F+C)	73.85%
Adaptive	Standard	N	WI (F) Convergence Fail
Adaptive	Standard	N + WI (C)	WI (F) Convergence Fail
Adaptive	Standard	N + WI (F)	50.70%
Adaptive	Standard	N + WI (F+C)	50.07%
Adaptive	Wasserstein		86.15%
Adaptive	Wasserstein	WI (C)	97.94%
Adaptive	Wasserstein	WI (F)	75.85%
Adaptive	Wasserstein	WI (F+C)	75.52%
Adaptive	Wasserstein	N	WI (F) Convergence Fail
Adaptive	Wasserstein	N + WI (C)	WI (F) Convergence Fail
Adaptive	Wasserstein	N + WI (F)	50.63%
Adaptive	Wasserstein	N + WI (F+C)	50.07%

4.2.1. Nonuniform adaptation

We validate the effectiveness of our adaptive bandwidth method by comparing it against a uniform bandwidth method in the task setting which heavily stresses this ability. The results are shown in Table 4.2.

Interpretation of the results. Uniform bandwidth methods have practically random classifier accuracy, while our adaptive bandwidth method attains nearly perfect accuracy with the standard loss, and slightly lower accuracy with the Wasserstein loss. This clearly shows a case where our adaptive bandwidth method is successful, since the task is designed to isolate the ability of a model to allocate its bandwidth effectively, and since the gap in accuracy is effectively at its theoretical maximum.

The visualization in Figure 4.7 shows that the uniform method is unable to discretize the signal without severe aliasing except for outliers, while the visualization in Figure 4.8 shows our uniform method discretizes it appropriately in all cases even given the small number of samples on the lattice. Note that all visualizations correspond to the flows trained with the standard loss, and show eight elements of a batch, concatenated side by side.

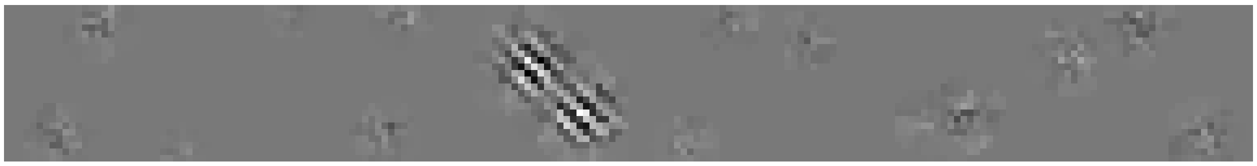


Fig. 4.7. Rediscretized signal with the uniform method.

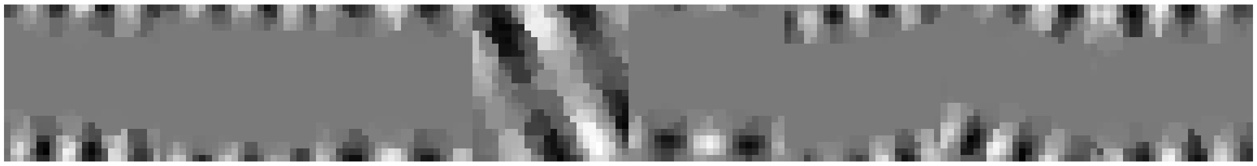


Fig. 4.8. Rediscretized signal with the nonuniform method.

4.2.2. Ablation of weight initialization

We investigate the usefulness of weight initialization (section 3.8, subsection 3.9.8) in our adaptive bandwidth method by performing ablations on weight initialization. A few permutations of this are possible, because weight initialization applies both to the convolutional architecture and the flow architecture. We notate the results in Table 4.2 with **WI (C)** for ablation of the convolutional initialization only; **WI (F)** for ablation of the flow initialization only; **WI (F+C)** for ablations of both.

Ablation: WI (C). Ablating weight initialization of the convolutional architecture (section 3.8) seems to have a negligible impact in this scenario. This cannot be interpreted as an indication that it is ineffective, however, since the network is so shallow that the effects of vanishing or exploding gradients are unlikely to be important. It is well understood that deep neural networks cannot be trained without sensible weight initialization (subsection 1.3.1, subsection 1.3.2).

Ablation: WI (F). Ablating weight initialization of the flow architecture (subsection 3.9.8) appears to have a great impact on the effectiveness of our method, as the ablations show significantly worse performance.

The visualizations in Figure 4.9 show how this ablation results in the adaptive bandwidth mapping learnt by the flow architecture converging poorly, which strains the ability of the convolutional architecture to solve the classification task effectively, as it is given heavily distorted or incomplete information.

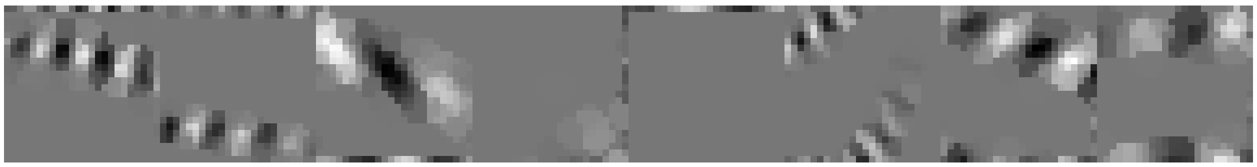


Fig. 4.9. Rediscretized signal when ablating flow weight initialization.

Ablation: WI (C+F). When both ablations are performed simultaneously, the penalty incurred by the flow weight initialization seems to dominate, which is consistent with the hypothesized mechanism causing performance degradation.

4.2.3. Ablation of normalization

We similarly investigate the effectiveness of flow normalization (subsection 3.9.9) in our adaptive bandwidth by performing ablations on normalization, notated with **N** in Table 4.2

Ablation: N / N + WI (C). Ablating normalization while also ablating flow weight initialization causes our adaptive method regresses to random classifier accuracy.

The visualizations in Figure 4.10 show how the flow architecture fails to converge to a usable mapping, destroying useful information before the convolutional architecture begins operating on the signal.

Fig. 4.10. Rediscretized signal when ablating flow weight initialization and normalization.

Ablation: N + WI (F) / N + WI (C + F). Ablating normalization without ablating flow weight initialization causes the weight initialization of the flow architecture fails to converge, which makes training impossible. This is reasonable given that the magnitude of the weight initialization of a continuous flow does not directly translate to the magnitude of a linear transformation it expresses, meaning that deriving a weight initialization which raises or lowers variance is nontrivial. This shows that unless we restrict our choice of normalizing flow layers, the union of both normalization and weight initialization of the flow architecture is necessary for our adaptive method to succeed.

4.2.4. Nonuniform compression

We study a more realistic scenario for the application of our adaptive bandwidth method by applying the compression technique introduced in subsection 3.9.11 to run inference on the CIFAR10 network trained in subsection 4.1.2.

We simply reuse the architecture used in section 4.1, with weights obtained in subsection 4.1.2. However, we limit the Gaussian kernel neighbourhood radius to 8 samples, to avoid the memory allocation crashes described in subsection 3.9.5. This possibly introduces discretization errors.

The flow architecture again relies on continuous flows (subsection 3.9.10), with two hidden layers each with 64 feature channels, for a total of 4546 parameters. We Euler integrate it over 16 steps. This is preceded by deterministic arbitrary covariance Gaussian normalizer (subsection 3.9.9. We choose to optimize the standard loss on a surrogate Gaussian plateau (subsection 3.9.7) with $\sigma = 0.125$, a uniform domain $[-1, 1]^2$, and a rediscretization cropped to contain only the uniform part of the distribution. We optimize the flow architecture with AdamW (Kingma and Ba, 2014; Loshchilov and Hutter, 2017) in 64 steps, with a learning rate of $10^{-2.5}$, with both betas at 0.9, and no weight decay.

Interpretation of the results. Figure 4.11 shows how the accuracy of the trained model varies as it is evaluated with various discretizations. The compression technique performs poorly except at the lowest resolutions. This is likely due to a combination of errors coming from the approximations that are made in the compression heuristic and in the adaptation of the many components of our method to the nonuniform setting.

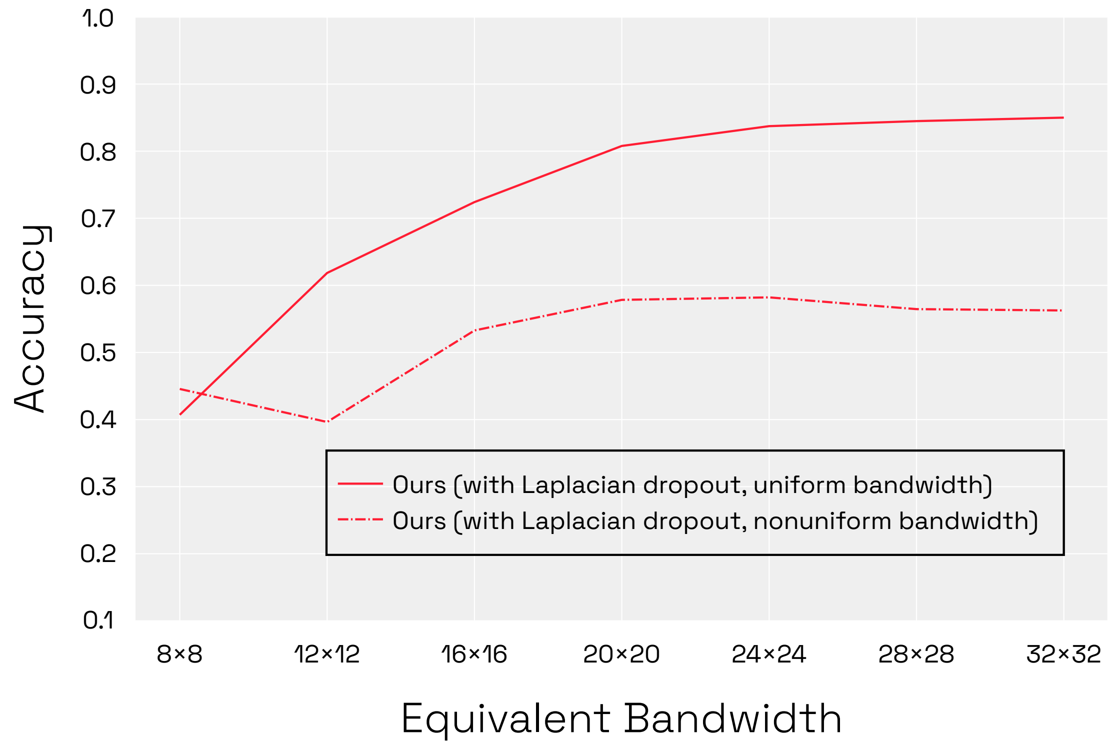


Fig. 4.11. Evaluation of our method with and without nonuniform compression on the CIFAR10 dataset.

The computational performance is also notably awful. Evaluation times on the nonuniform method are 15 to 70 times higher than that of the uniform method, and the 32×32 variant requires up to 70GB of memory at a batch size of 128, which is absurd for a model using less than a million parameters. This stems mostly from the flaws outlined in subsection 3.9.5.

The benefits of the better sample efficiency provided by compression are far outweighed by poor performance and by the computational burden incurred by the adaptation method.

Chapter 5

Discussion

We lead this work with the central of formulating deep learning architectures for signals that are adaptable, robust, and computationally efficient. We observe there is great variety in the way sensor systems translate continuous signals to discretized signals, yet there is little work which studies how architectures can deal with this constraint gracefully.

We set out to examine this question from a theoretical perspective. In subsection 2.1.2, we introduce the notions of invariances and equivariances in the context of architecture design as inductive biases. These two notions allow distinguishing what part of some information constitutes its quality of interest, and what part is contingent, and they can be enforced as mathematical constraints in architecture design to enforce this selectivity to information. In section 2.2, we then build towards a perspective of signals that allows treating continuous signals and discretized signals interchangeably, and that critically enables operators, invariances and equivariances defined on continuous signals to be adapted on discretized signals. We aim for generality by allowing extension to multiple dimensions, discretization of *either or both* the spatial domain and spectral domain, support for arbitrary lattices, and nonuniform distributions. In subsection 2.2.25, we introduce the Laplacian pyramid, which provides a way of decomposing a signal into a sum of signals that each correspond to a spectral partition. In section 2.3, we finally discuss normalizing flows, which enable the creation of models that allow matching distributions, which we find useful when manipulating nonuniform signals.

With this set of tools in mind, we identify many paths for improvement left unexplored by prior work: we recognize that architectures can be made freely adaptable to any discretization by formulating them through discretization invariant operators; that robustness may be improved by implementing continuous shift equivariance, which is stronger than discrete shift equivariance; that computational cost and memory footprint can be improved by lossless compression through *Laplacian residuals*, which combine features of both Laplacian pyramids and residual connections to allow the removal of inactive high-bandwidth layers when operating on low-bandwidth signals; that robustness can be improved by a form of *Laplacian dropout* that is simple to implement, which simulates low-bandwidth signals during training; that sample efficiency can be improved by nonuniform compression through deformation of the receptive field using normalizing flows.

We formulate an architecture that exploits these opportunities and show that many of these improvements are effective. In subsection 4.1.1, we show that unlike prior architectures, our architecture is able to adapt to arbitrary discretizations very gracefully, and also possesses excellent parameter efficiency and training dynamics. In subsection 4.1.2, also unlike prior architectures, we show that *Laplacian residuals* and *Laplacian dropout* are very effective at promoting robustness to low-bandwidth signals; we also find that our architecture thrives when training on heterogeneous-bandwidth signals. In subsection 4.1.3, we empirically validate our theoretical guarantee for lossless compression using Laplacian residuals; we can in fact discard high-bandwidth layers when operating on low-bandwidth signals without disturbance and without any form of re-training. In subsection 4.2.1, we find that our nonuniform compression scheme is effective at promoting sample efficiency in a simple synthetic task designed for extreme sensitivity to sample efficiency. In subsection 4.2.2 and subsection 4.2.3, we show that the design decisions behind our nonuniform compression scheme are effective through a set of ablation studies. In subsection 4.2.4, however, we find that nonuniform compression performs very poorly on more realistic tasks, likely due to the many approximations involved in the formulation of the method; its computational cost also disqualifies it as a viable method.

In summary, we believe that this theoretical approach to deep learning for signals is still underused; many of the theoretical flaws of current architectures are not well characterized, and many avenues for improved architectures exist. While the mathematics that underlie continuous signals and discretized signals are well established, the bridge between the two is often ineffective or obscured. We therefore wish to make the perspective of signals we have developed more easily accessible to the community. We also wish to improve on the limitations of this work: the modifications we implement into our architecture can be difficult to translate to other architectures; the code that implements our architecture is needlessly inefficient, simply because the code evolved organically over months without a clearly defined specification; the experiments we perform rely on relatively low-bandwidth signals, which prevents us from peering into the capabilities of our architecture on high-bandwidth signals, where our theoretical advantages may be especially useful. We specifically want to streamline the idea of *Laplacian residuals* and *Laplacian dropout*, simplify their integration within state-of-the-art architectures, provide code that is very computationally efficient, and provide a set of larger-scale experiments so that our contribution can be more useful to the community.

References

Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. (1984). Pyramid methods in image processing. *RCA engineer*, 29(6):33–41.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Burt, P. J. and Adelson, E. H. (1987). The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Coeurdoux, F., Dobigeon, N., and Chainais, P. (2022). Sliced-wasserstein normalizing flows: beyond maximum likelihood training. *arXiv preprint arXiv:2207.05468*.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.

Cohen, T. S., Geiger, M., and Weiler, M. (2019). A general theory of equivariant cnns on homogeneous spaces. *Advances in neural information processing systems*, 32.

Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301.

Devroye, L. (1986). *Non-uniform random variate generation*. New York: Springer-Verlag. ISBN: 0-387-96305-7.

Dirac, P. A. M. (1927). The physical interpretation of the quantum dynamics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 113(765):621–641.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Elizar, E., Zulkifley, M. A., Muharar, R., Zaman, M. H. M., and Mustaza, S. M. (2022). A review on multiscale-deep-learning applications. *Sensors*, 22(19):7384.

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. (2018). Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68.

Fanaskov, V. and Oseledets, I. (2022). Spectral neural operators. *arXiv preprint arXiv:2205.10573*.

Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. (2020). Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR.

Fourier, J. B. J. (1888). *Théorie analytique de la chaleur*. Gauthier-Villars et fils.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Han, Y., Huang, G., Song, S., Yang, L., Wang, H., and Wang, Y. (2021). Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

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

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France. PMLR.

Jenner, E. and Weiler, M. (2021). Steerable partial differential operators for equivariant neural networks. *arXiv preprint arXiv:2106.10163*.

Kabri, S., Roith, T., Tenbrinck, D., and Burger, M. (2023). Resolution-invariant image classification based on fourier neural operators.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. (2021). Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863.

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

Knutsson, H., Westin, C.-F., and Granlund, G. (1994). Local multiscale frequency and bandwidth estimation. In *Proceedings of 1st International Conference on Image Processing*, volume 1, pages 36–40 vol.1.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. (2020). Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979.

Kondor, R. and Trivedi, S. (2018). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pages 2747–2755. PMLR.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*.

Krähenbühl, P., Doersch, C., Donahue, J., and Darrell, T. (2015). Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Lai, W.-S., Huang, J.-B., Ahuja, N., and Yang, M.-H. (2017). Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 624–632.

Landau, H. (1967). Necessary density conditions for sampling and interpolation of certain entire functions.

Lang, L. and Weiler, M. (2020). A wigner-eckart theorem for group equivariant convolution kernels. *arXiv preprint arXiv:2010.10952*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (2002). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Marcos, D., Kellenberger, B., Lobry, S., and Tuia, D. (2018). Scale equivariance in cnns with vector fields. *arXiv preprint arXiv:1807.11783*.

Marvasti, F. (2012). *Nonuniform sampling: theory and practice*. Springer Science & Business Media.

Mishkin, D. and Matas, J. (2015). All you need is a good init. *arXiv preprint arXiv:1511.06422*.

Neumann, V. (1951). Various techniques used in connection with random digits. *Notes by GE Forsythe*, pages 36–38.

Petersen, D. P. and Middleton, D. (1962). Sampling and reconstruction of wave-number-limited functions in n-dimensional euclidean spaces. *Information and Control*, 5(4):279–323.

Poli, M., Massaroli, S., Berto, F., Park, J., Dao, T., Ré, C., and Ermon, S. (2022). Transform once: Efficient operator learning in frequency domain. *Advances in Neural Information Processing Systems*, 35:7947–7959.

Recasens, A., Kellnhofer, P., Stent, S., Matusik, W., and Torralba, A. (2018). Learning to zoom: a saliency-based sampling layer for neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 51–66.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

Ruthotto, L. and Haber, E. (2020). Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62:352–364.

Schönhage, A. and Strassen, V. (1971). Fast multiplication of large numbers. *Computing*, 7:281–292.

Schwartz, L. (1957). Théorie des distributions à valeurs vectorielles. i. In *Annales de l'institut Fourier*, volume 7, pages 1–141.

Schwartz, L. (1958). Théorie des distributions à valeurs vectorielles. ii. In *Annales de l'institut Fourier*, volume 8, pages 1–209.

Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21.

Shen, Z., He, L., Lin, Z., and Ma, J. (2020). Pdo-econvs: Partial differential operator based equivariant convolutions. In *International Conference on Machine Learning*, pages 8697–8706. PMLR.

Singh, S. R., Yedla, R. R., Dubey, S. R., Sanodiya, R., and Chu, W.-T. (2021). Frequency disentangled residual network. *arXiv preprint arXiv:2109.12556*.

Sobel, I. (1968). An isotropic 3x3 image gradient operator. *Presentation at Stanford A.I. Project 1968*.

Sosnovik, I., Szmaja, M., and Smeulders, A. (2019). Scale-equivariant steerable networks. *arXiv preprint arXiv:1910.11093*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Strichartz, R. S. (2003). *A guide to distribution theory and Fourier transforms*. World Scientific Publishing Company.

Torre, V. and Poggio, T. A. (1986). On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):147–163.

Wang, D., Shelhamer, E., Olshausen, B., and Darrell, T. (2019). Dynamic scale inference by entropy minimization. *arXiv preprint arXiv:1908.03182*.

Weierstrass, K. (1895). On continuous functions of a real argument which possess a definite derivative for no value of the argument. *Koniglich Preussichen Akademie der Wissenschaften, Mathematische Werke von Karl Weierstrass*, pages 71–74.

Weiler, M. and Cesa, G. (2019). General e (2)-equivariant steerable cnns. *Advances in neural information processing systems*, 32.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. (2018a). 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *Advances in Neural Information Processing Systems*, 31.

Weiler, M., Hamprecht, F. A., and Storath, M. (2018b). Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858.

Weinan, E. (2017). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 1(5):1–11.

Whittaker, E. (1915). On the functions which are represented by the expansion of interpolating theory. In *Proc. Roy. Soc. Edinburgh*, volume 35, pages 181–194.

Whittaker, J. M. (1927). On the cardinal function of interpolation theory. *Proceedings of the Edinburgh Mathematical Society*, 1(1):41–46.

Woodward, P. M. and Davies, I. L. (1952). Information theory and inverse probability in telecommunication. *Proceedings of the IEE-Part III: Radio and Communication Engineering*, 99(58):37–44.

Worrall, D. and Welling, M. (2019). Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. (2017). Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5028–5037.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, Y., Xiao, T., Zhang, J., Yang, K., and Zhang, Z. (2014). Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*.

Ziou, D., Tabbone, S., et al. (1998). Edge detection techniques-an overview. *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, 8:537–559.