Université de Montréal

Apprentissage d'Espaces Sémantiques

par Grégoire Mesnil

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

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Janvier, 2015

Et le singe devint con. *François Cavanna*, “Histoire de l’Humanité”

Résumé

Dans cette dissertation, nous présentons plusieurs techniques d'apprentissage d'espaces sémantiques pour plusieurs domaines, par exemple des mots et des images, mais aussi à l'intersection de différents domaines. On utilisera différentes techniques d'apprentissage machine pour apprendre des représentations avec des propriétés intrinsèques intéressantes. Ces propriétés peuvent aller d'un meilleur mixage (un espace où il est plus aisé de faire de l'échantillonnage), à une meilleure séparabilité linéaire (un espace où il est plus facile de discriminer avec un hyperplan). Un espace de représentation est appelé sémantique si des entités jugées similaires par un être humain, ont leur similarité préservée dans cet espace. Les 5 articles qui forment le corps de cette thèse correspondent à des avancées dans les techniques permettant d'apprendre de telles représentations ou d'en tirer profit.

La première publication présente un enchaînement de méthodes d'apprentissage incluant plusieurs techniques d'apprentissage non supervisé utilisées qui nous a permis de remporter la compétition "Unsupervised and Transfer Learning Challenge" en 2011. La particularité de cette compétition était qu'aucun a priori sur la structure des données ne pouvait être utilisé car les différents ensembles de données (reconnaissance de caractères manuscrits, actions humaines, traitement du langage naturel, données écologiques, classification d'images) étaient rendus anonymes par des permutations aléatoires. Le deuxième article présente une manière d'extraire de l'information à partir d'un contexte structuré (177 détecteurs d'objets à différentes positions et échelles). On montrera que l'utilisation de la structure des données combinée à un apprentissage non supervisé permet de réduire la dimensionnalité de 97% tout en améliorant les performances de reconnaissance de scènes de +5% à +11% selon l'ensemble de données.

Dans le troisième travail, on s'intéresse à la structure apprise par les réseaux de neurones profonds utilisés dans les deux précédentes publications. Plusieurs hypothèses sont présentées et testées expérimentalement montrant que l'espace appris a de meilleures propriétés de mixage (facilitant l'exploration de différentes classes durant le processus d'échantillonnage).

Pour la quatrième publication, on s'intéresse à résoudre un problème d'analyse syntaxique et sémantique avec des réseaux de neurones récurrents appris sur des fenêtres de contexte de mots. Avec l'approche proposée, nous obtenons une nette amélioration de l'état de l'art sur plusieurs ensembles de données et mettons en évidence d'intéressantes propriétés de l'espace sémantique associé au langage. Une recherche sur l'apprentissage d'un espace sémantique à l'intersection des mots et des images est présentée dans notre cinquième travail. On propose une façon d'effectuer de la recherche d'image "augmentée" en apprenant un espace sémantique où une recherche d'image contenant un objet retournerait aussi des images des parties de l'objet, par exemple une recherche retournant des images de "voiture" retournerait aussi des images de "pare-brises", "coffres", "roues" en plus des images initiales.

Summary

In this work, we focus on learning semantic spaces for multiple domains, but also at the intersection of different domains. The semantic space is where the learned representation lives. This space is called semantic if similar entities from a human perspective have their similarity preserved in this space. We use different machine learning algorithms to learn representations with interesting intrinsic properties.

The first article presents a pipeline including many different unsupervised learning techniques used to win the Unsupervised and Transfer Learning Challenge in 2011. No prior knowledge on the structure of the data could be used during this competition since the different datasets were anonymous. Using the learned representations improved the performance of a weak classifier compared to using the raw data. It also transfered well to different subsets of classes for the 5 different domain datasets in this competition.

In the second article, we present a pipeline largely inspired from the one above but taking advantage of the structure of the data for a scene classification problem. We present a way to learn from structured context, consisting of 177 object detections at different poses and scales. We show that using the structure of the data combined with the Contractive Auto-Encoder, an unsupervised learning algorithm, allows us to drastically reduce the dimensionality while improving significantly on the scene recognition accuracy.

The third article focuses on the space structure learned by deep representations. Our experiments use greedy layer-wise unsupervised training of Contractive Auto-Encoders and Restricted Boltzmann Machines. Several hypothesis are experimentally tested and show that abstract representation spaces have better mixing properties. We conclude that performing the sampling procedure from the representation space explores more of the different classes.

In the fourth article, we tackle a semantic parsing problem with several Recurrent Neural Network architectures taking as input context windows of word embeddings. We show an improvement over the state of the art previously obtained with Conditional Random Fields on different datasets. Learning context windows of word embeddings leads to a word semantic space where words with the same output class are clustered together. Depending on the dataset, performing Viterbi decoding with the probabilities of the model is crucial to obtain good performance.

In the fifth article, an investigation on learning a single semantic space at the intersection of words and images is presented. As we lack labeled images of objects and their parts, we use the Word-Net part-of relationship to obtain a training proxy. Through the semantic space intersecting words and images domains, we propose a way to perform ”augmented search” where a search on an image containing an object would also return images of the object’s parts.

Articles

1. Grégoire Mesnil, Yann Dauphin, Xavier Glorot, Salah Rifai, Yoshua Bengio, et al. **Unsupervised and Transfer Learning Challenge: a Deep Learning Approach** (2012) Journal of Machine Learning Workshop and Conference Papers, Proceedings of the Unsupervised and Transfer Learning challenge and workshop, pages 97-110
2. Grégoire Mesnil, Salah Rifai, Antoine Bordes, Xavier Glorot, Yoshua Bengio and Pascal Vincent **Unsupervised Learning of Object Detections Semantics for Scene Categorization** (2015) Pattern Recognition Applications and Methods, Advances in Intelligent Systems and Computing, pages 209-224
3. Yoshua Bengio[1], Grégoire Mesnil[1], Yann Dauphin and Salah Rifai **Better Mixing via Deep Representations** (2013) Proceedings of the 30th International Conference on Machine Learning
4. Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu and Geoffrey Zweig **Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding** (2015) IEEE Transactions on Audio, Signal and Language Processing, in press
5. Grégoire Mesnil, Antoine Bordes, Jason Weston, Gal Chechik and Yoshua Bengio, **Learning Semantic Representations Of Objects and Their Parts** (2013) Machine Learning Journal, volume 94, pages 281–301

1. indique une contribution similaire

Table des matières

Table des figures

Liste des tableaux

Remerciements

Pour commencer, je tiens à remercier Yoshua pour son ouverture à toujours discuter de nouvelles idées et sa rapidité d'exécution pour les mettre sous presse. Le LISA est un environnement de recherche d'exception, un des rares endroits où l'on peut voir un laboratoire surpasser la recherche industrielle avec des moyens académiques. J'aimerais aussi remercier mes co-directeurs Pascal et Alain pour m'avoir fait confiance et octroyé une grande liberté dans ma recherche.

Toute l'équipe du GAMME, j'y ai recontré des collègues et amis qui me sont chers: mon wingman faucon bassiste préféré Yann, notre futur iron-man national Xavier. Mais aussi toutes les bouteilles de champagne et l'écumage de la rue Laurier en mode stealth et beurre argentin avec Salah le stoïcien et Xavier notre père à tous. Je leur souhaite à tous une réussite à la hauteur de leurs ambitions.

Les chercheurs talentueux que j'ai pu rencontrer en industrie dans l'équipe de Microsoft Research à Redmond, notamment Xiaodong et Li. Puis à Facebook Articial Intelligence Research en Californie avec Antoine, Marc'Aurelio, Tomas et Jason.

Parmi mes racines françaises, Simon qui est passé devant dans la forêt aléatoire, ma gang du Cotentin, avec Alban le californien et Florence la femme fatale en tête. Toujours, la famille avec mes parents sans qui je ne serais sans doute pas là, et mes deux petites soeurs, Margaux et Clémence, déjà rendues bien grandes.

Mes belles rencontres montréalaises avec Michelle et Séb, pour les vibrations dans le placard et les colons ; Baptiste, pour les soirées Michelle Drucker dans l'Unité, Marlène, Val, Marie-Anne, Thibault les meilleur(e)s colocataires du monde, Maude, Pat et Miki, mes photographes préférés, et la gang du ski de la Fondation des Aveugles. Et bien sûr mes recontres califoniennes, avec Bob et ma très chère Amina.

J'aimerais aussi remercier ceux qui ont rendu mon doctorat possible au travers de leurs financements et de leurs moyens de calcul: le DIRO, l'ANR, la FESP et le Consul de France ; NSERC, Calcul Québec, Compute Canada et CIFAR et toute l'équipe derrière Theano, Frédéric en tête.

Pour terminer, j'aimerais remercier tout le monde, j'espère ainsi que je n'ai oublié personne.

1 Apprentissage Machine d'Architectures Profondes

1.1 Introduction à l'Apprentissage Statistique

L'apprentissage machine est un champ de recherche de l'intelligence artificielle. Il permet notamment d'extraire des informations utiles à l'aide à la décision. L'objectif ici est de pouvoir extraire de manière quasiment automatique à partir de grandes masses de données une information pertinente. C'est l'immense quantité de données (ou leur dimensionnalité trop élevée) qui rend cette tâche difficile, parfois impossible, à accomplir pour un être humain.

Ce chapitre introduit les quelques notions et techniques de base de l'apprentissage statistique sur lesquelles s'appuient nos travaux.

1.2 Applications de cette thèse

Les chercheurs en apprentissage machine résolvent une grande variété de problèmes sitôt qu'un ensemble suffisant de données est disponible. Dans ce travail, on considère les tâches suivantes:

— **Apprentissage de représentation non supervisé.** L'ensemble de données ne contient pas d'étiquettes, seulement des entrées $x \in \mathbb{R}^d$. L'objectif est d'apprendre une représentation des données $h(x)$ de meilleure qualité que la représentation brute x. Ces qualités vont de la réduction de dimensionnalité (Chapitres 4, 6), une meilleure séparabilité linéaire de leurs classes naturelles (Chapitres 4, 6), de meilleures propriétés de mixage (Chapitre 8) ou une meilleure initialisation des paramètres pour un apprentissage supervisé (Chapitre 6).

— **Transfert de domaine.** Le résultat de l'algorithme est utilisé sur un ensemble de données cible $\tilde{\mathcal{D}}$ dans un autre domaine que l'ensemble d'entraînement $\mathcal{D}$. Cela permet d'avoir une représentation plus spécifique à l'ensemble cible tout en obtenant un gain de performances dû par exemple, à l'utilisation d'un plus grand ensemble de données d'entraînement. Dans ce travail, nous présenterons l'utilisation d'un apprentissage par procuration (Chapitre 12), d'une analyse en composantes principales transductive (Chapitre 4) et d'un finetuning sur l'ensemble cible (Chapitre 6).

— **Apprentissage d'espaces sémantiques multi-domaines.** L'idée ici est d'avoir une représentation vectorielle dans un espace euclidien pour chaque entité. Un *mot* est associé à un vecteur $x_{mot} \in \mathbb{R}^d$ qui est ajusté lors de l'apprentissage. Ces espaces possèdent une sémantique propre dans le sens où des vecteurs proches dans cet espace ont un sens similaire. Par exemple des mots voisins vont avoir un même rôle syntaxique (ex: adjectif) ou une sémantique proche (ex: des noms de pays). Nous montrerons qu'il est possible d'avoir un espace commun à plusieurs domaines comme des mots et des images (Chapitre 12).

— **Classification de séquences.** Il s'agit ici de classifier des données x_t qui en plus d'avoir une étiquette y_t, possèdent une composante temporelle. Ceci est un aspect du problème intéressant à exploiter du fait des dépendances possibles entre les prédictions à $t-1$, t et $t+1$ par exemple. Des architectures spécifiques de type réseaux de neurones récurrents pour l'apprentissage d'un espace sémantique modélisant le contexte seront présentées ici (Chapitre 10).

1.3 Risque Empirique et Espéré

Commençons d'abord par une formalisation du problème d'apprentissage. L'objectif est de trouver une fonction $f \in \mathcal{F}$ qui va exécuter une tâche particulière. Il faut donc définir l'espace des fonctions $\mathcal{F}$ parmi lequel nous allons chercher la solution optimale à notre problème. Il faut également définir une fonction de coût ou d'objectif $\mathcal{C}$ qui évalue retourne un réél $\mathcal{C}(\mathcal{D}, f) \in \mathbb{R}$ nous permettant d'évaluer la qualité d'une fonction donnée sur un échantillon de données $\mathcal{D}$.

La solution à notre problème est donc la fonction f^* définie par:

$$f^* = \underset{f \in \mathcal{F}}{\arg\min}\, \mathbb{E}_{\mathcal{D}}[\mathcal{C}(\mathcal{D}, f)] \tag{1.1}$$

En supposant qu'on ait accès à une infinité de données, nous serions en mesure de calculer de façon exacte le **risque espéré** $\mathbb{E}_{\mathcal{D}}[\mathcal{C}(\mathcal{D}, f)]$. Malheureusement, l'ensemble de données à notre disposition est de taille finie et on ne peut donc calculer qu'un estimé du risque espéré. L'idée est de minimiser l'objectif sur un ensemble d'entraînement $\mathcal{C}(\mathcal{D}_{\text{train}}, f)$ puis d'approximer le risque espéré sur un ensemble qui n'a pas été vu lors de l'apprentissage, un ensemble de test $\mathcal{D}_{\text{test}}$. On va donc minimiser le **risque empirique** sur l'ensemble d'entraînement:

$$\hat{f}^* = \underset{f \in \mathcal{F}}{\arg\min}\, \mathcal{C}(\mathcal{D}_{\text{train}}, f)] \tag{1.2}$$

Pour une quantité de données limitée, il est possible de trouver une fonction $f \in \mathcal{F}$ qui donne un risque empirique proche de 0 pourvu que $\mathcal{F}$ définisse un espace de fonctions à la complexité assez élevée (il est parfois possible de mesurer cette complexité avec la dimension de Vapnik-Chervonenkis (Vapnik and Chervonenkis, 1971)). Mais ceci ne résulte pas nécessairement en une solution qui généralise bien sur de nouvelles données qui n'ont pas été utilisées au cours de l'entraînement. Ceci est un phénoméne appelé *sur-apprentissage*: on a un risque empirique faible sur l'ensemble d'entraînement et un risque espéré élevé.

Afin de réduire l'écart de performance en généralisation entre la solution du problème de minimisation du risque empirique $\hat{f}^*$ et la solution de l'Eq. 1.2 dénotée f^*, une des alternatives consiste à réduire l'espace $\mathcal{F}$ des fonctions à des fonctions d'une complexité moins élevée. On peut aussi ajouter une pénalité de régularisation sur la complexité de la fonction f contrôlée par un coefficient λ. Une régularisation trop forte peut parfois aboutir à du *sous-apprentissage*. Un exemple typique de sous-apprentissage est "l'apprentissage" d'une fonction constante qui ne s'ajuste pas pour minimiser le risque empirique ou un processus de minimisation par descente de gradient qui diverge suite à l'utilisation d'un pas de gradient trop fort.

En général, un ensemble de données est découpé en 3 ensembles disjoints.

— L'**ensemble d'apprentissage** $\mathcal{D}_{\text{train}}$ pour la minimisation du risque empirique et l'entraînement des paramètres de la fonction f.

— L' **ensemble de validation** $\mathcal{D}_{\text{valid}}$ permet de choisir les hyper-paramtètres de l'algorithme d'apprentissage comme par exemple la pénalité de régularisation λ.

— L'**ensemble de test** $\mathcal{D}_{\text{test}}$ permet d'avoir un approximé du risque espéré.

1.4 Log-vraisemblance

Notre objectif est donc d'avoir le risque espéré le plus faible possible. L'algorithme doit pouvoir généraliser au mieux à de nouveaux exemples. L'hypothèse de base est que les données sont tirées de la même loi pour tous nos ensembles $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{valid}}$ et $\mathcal{D}_{\text{test}}$. En ce sens, les données partagent la même distribution. Ensuite, on suppose généralement que les données sont indépendantes, c'est-à-dire que chaque exemple a été généré par cette distribution indépendamment des autres exemples. Ces deux hypothèses se résument dans l'acronyme i.i.d., indépendantes et identiquement distribuées. L'indépendance se traduit par l'équation suivante:

$$\mathbb{P}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \mathbb{P}(x_i) \tag{1.3}$$

avec $\mathbb{P}$ la distribution à l'origine de notre ensemble de données $\mathcal{D} = x_1, \ldots, x_n$.

Une approche populaire d'apprentissage machine consiste à maximiser la log-vraisemblance sur l'ensemble d'entraînement. Si f_θ est notre modèle, une fonction de densité de probabilité paramétrée par θ, pour approximer $\mathbb{P}$, on ajuste les paramètres θ pour maximiser l'équation suivante:

$$\theta^* = \arg\max_\theta \log f_\theta(x_1, \ldots, x_n) = \log \prod_{i=1}^{n} f_\theta(x_i) = \sum_{i=1}^{n} \log f_\theta(x_i) \tag{1.4}$$

Un produit de probabilités $f_\theta(x)$ est une source d'instabilité numérique. La précision numérique des ordinateurs étant limitée, il est indispensable de rester dans la plage d'encodage définie par nos ordinateurs modernes. Cette décomposition en somme après projection dans l'espace logarithmique permet de remédier à ces problèmes d'instabilité. Maximiser l'équation 1.4 se fait au travers d'un processus d'optimisation.

1.5 Optimisation

L'optimisation qui nous intéresse vise à trouver les paramètres θ qui minimisent (ou maximisent à une inversion de signe près) une fonction donnée $\mathcal{L}(\theta)$. On s'intéresse donc ici à des modèles paramétriques (dépendant de θ). Dans la plupart des cas qui nous intéressent, la fonction à minimiser est différentiable par rapport aux paramètres θ. Cela nous permet de calculer un gradient $\partial\mathcal{L}/\partial\theta$ qui nous donnera une direction à suivre pour minimiser la fonction en mettant à jour les paramètres. Si la forme de la solution n'est pas disponible de manière analytique, on procède par étapes en ajustant les paramètres d'un pas α à chaque étape. Cette méthode s'appelle la descente de gradient:

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial}{\partial\theta}\mathcal{L} \tag{1.5}$$

Le gradient dépend des exemples d'apprentissage et il peut être calculé de difféntes manières. Sur l'ensemble d'apprentissage $\mathcal{D}_{train}$ au complet par exemple, on parle dans ce cas de descente de gradient de type **batch**. Dans le cadre de la maximisation de la log-vraisemblance cela donne:

$$\theta_{t+1} = \theta_t + \alpha \frac{\partial}{\partial\theta} \sum_{x\in\mathcal{D}_{train}} \log f_\theta(x) \tag{1.6}$$

En général, on préfère calculer les gradients soit sur un seul exemple ou un sous-ensemble de taille fixe $\tilde{\mathcal{D}}$ extrait de $\mathcal{D}_{train}$. Le sous-ensemble $\tilde{\mathcal{D}}$ est renouvelé avec de nouveaux exemples à chaque itération du processus d'optimisation. On parle en ce cas de descente de gradient **stochastique** ou **mini-batch**.

$$\theta_{t+1} = \theta_t + \alpha \frac{\partial}{\partial\theta} \sum_{x\in\tilde{\mathcal{D}}_{train}} \log f_\theta(x) \tag{1.7}$$

Lorsque la dérivée seconde $\partial^2\mathcal{L}/\partial\theta^2$ est calculable en pratique, il peut être intéressant d'utiliser la courbure de la fonction de coût pour faire les mises à jour.

L-BFGS (Byrd et al., 1995) par exemple est une méthode populaire. Néanmoins, en raison de son faible coût computationnel, la technique de descente de gradient stochastique reste la plus couramment utilisée pour l'optimisation de réseaux de neurones que nous présenterons à la section suivante.

1.6 Apprentissage Supervisé

Dans le cadre de l'apprentissage supervisé et plus spécifiquement des tâches de classification, nous disposons des entrées $x \in \mathbb{R}^d$ et d'une étiquette $y \in \{1, \ldots, C\}$ qui leur est associé. L'étiquette peut correspondre à la classe d'un chiffre $y \in \{0, \ldots, 9\}$ ou une catégorie d'objet par exemple. Lorsque C est extrêmement grand (de l'ordre du million), nous verrons dans le chapitre 12 qu'il est possible de contourner ce problème.

1.6.1 Réseaux de Neurones Profonds

On restreint ici l'espace des fonctions $\mathcal{F}$ à celui des réseaux de neurones à plusieurs couches plus communément appelés Perceptron Multi-Couches, en anglais *Multi-Layer Perceptron* (MLP)(Rumelhart et al., 1986). Soit $x \in \mathbb{R}^d$ l'entrée fournie au réseau. On fixe $x = h^{(0)}$. La sortie $h^{(i)}(x)$ d'une couche i est définie comme la projection affine $W^{(i)}h^{(i-1)}(x) + b^{(i)}$ des données de la couche inférieure $h^{(i-1)}(x)$ suivie d'une non-linéarité $h^{(i)}(x) = s(W^{(i)}h^{(i-1)}(x) + b^{(i)})$, en général la fonction sigmoïde logistique $s(x) = 1/(1 + e^{-x})$. La fonction sigmoïde logistique est différentiable et permet l'application de l'algorithme de rétropropagation du gradient.

La dernière couche d'un réseau de neurones pour résoudre un problème de classification a dans ce cas la forme d'une régression logistique multinomiale avec autant d'unités que de classes. La sortie d'un réseau à m couches pour C classes est donnée par

$$f(x) = \frac{e^{W^{(m+1)}h^{(m)}(x)+b^{m+1}}}{\sum_{j=1}^{C} e^{W_{j.}^{(m+1)}h^{(m)}(x)+b_j^{(m+1)}}} \in [0,1]^C \tag{1.8}$$

où $W_{j.}^{(m+1)}$ désigne la $j^{\text{ième}}$ ligne de la matrice $W^{(m+1)}$ et $b_j^{(m+1)}$ la $j^{\text{ième}}$ composante du vecteur $b^{(m+1)}$.

Les MLPs s'entraînent par rétropropagation du gradient (Rumelhart et al., 1986): c'est une application astucieuse de la règle de dérivée en chaîne et de la programmation dynamique pour que cela soit effectué de manière efficiente. On procède à la minimisation de la log-vraisemblance négative par descente de gradient pour $i = 1, \ldots, N$:

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial}{\partial \theta} [-\log f_{y^{(i)}}(x^{(i)})] \tag{1.9}$$

avec l'ensemble des paramètres contenus dans $\theta = \{W^{(i)}, b^{(i)}\}_{i=1}^{m+1}$.

1.7 Apprentissage Non Supervisé

Depuis (Hinton et al., 2006b; Bengio et al., 2007a), il a été démontré qu'il peut être avantageux de pré-entraîner un réseau de neurones en utilisant des algorithmes d'extraction de caractéristiques en apprentissage non supervisé. Plutôt que d'initialiser les paramètres de chaque couche $\{W^{(i)}, b^{(i)}\}_{i=1}^{m}$ de façon aléatoire, on les initialise avec les paramètres résultant de l'apprentissage non supervisé. Cela permet dans certains cas d'obtenir de nettes améliorations de performances et d'entraîner des réseaux à plusieurs couches de manière efficace.

Dans ce qui suit, nous présenterons les techniques d'apprentissage non supervisé les plus couramment utilisées pour l'extraction de caractéristiques.

Les algorithmes non supervisés abordés dans cette thèse n'utilisent aucune connaissane a priori du type des données *e.g.* musique, images, données séquentielles, ni leur classe d'appartenance *e.g.* 10 classes de 0 à 9 pour la classification de chiffres.

Il est très difficile de savoir pourquoi un algorithme d'apprentissage non supervisé fonctionne (bien ou mal) sur un certain type de donnée. De manière non exhaustive, voici énumérés quelques-uns des facteurs qui rendent difficile une analyse complète du fonctionnement de ces algorithmes: haute dimensionnalité des données, mauvais conditionnement du problème d'optimisation, processus d'optimisation non convexe, nombreuses paramétrisations possibles des modèles.

Une manière usuelle de sélectionner les hyperparamètres de tels algorithmes passe par une évaluation des performances discriminatives de la représentation apprise.

1.7.1 Analyse en Composantes Principales

Avant d'aborder des méthodes d'apprentissage plus sophistiquées, débutons par la méthode linéaire non supervisée la plus courante et la plus populaire: l'analyse en composantes principales ou en anglais *Principal Component Analysis* **(PCA)** (Pearson, 1901; Hotelling, 1933).

Une PCA avec k composantes principales permet d'obtenir les k composantes orthonormales dans l'espace d'entrée sur lesquelles projeter les données telles que l'on retient le plus de variance dans les données. Ces composantes correspondent aux k premiers vecteurs propres (si on les ordonne par valeurs propres décroissantes) de la matrice de covariance des données. Ces composantes sont ordonnées: la première correspond à la direction qui retient le plus d'information et ainsi de suite par niveau de variance décroissant.

Algorithme Soit $X \in \mathcal{M}_{n\times d}(\mathbb{R})$ la matrice qui contient l'ensemble d'entraînement $\mathcal{D} = \{x^{(i)} \in \mathbb{R}^d\}_{i=1,\ldots,n}$. En premier lieu, on calcule la moyenne empirique $\mu = (1/n)\sum_{i=1}^{n} X_i$ où X_i représente la ligne i de la matrice X *i.e.* l'exemple i. Les données sont centrées $\tilde{X} = X - \mu$ et on calcule la matrice de covariance $C = (1/n)\tilde{X}^T\tilde{X} \in \mathcal{M}_{d\times d}(\mathbb{R})$. La décomposition en valeurs propres de C est ensuite calculée (la partie la plus coûteuse): $C = V^{-1}UV$ où U est une matrice diagonale qui contient l'ensemble des valeurs propres et $V \in \mathcal{M}_{d\times d}$ les vecteurs propres associés (chaque colonne correspond à un vecteur propre). La sortie de la PCA correspond à la nouvelle représentation des données:

$$h(x) = V(x - \mu) \tag{1.10}$$

Si l'on veut avoir une PCA avec décorrélation, en anglais *whitening*, on construit la matrice diagonale U' à partir de U où $U'_{ii} = 1/\sqrt{C_{ii}}$ et la sortie est alors donnée par:

$$h(x) = U'V(x - \mu) \tag{1.11}$$

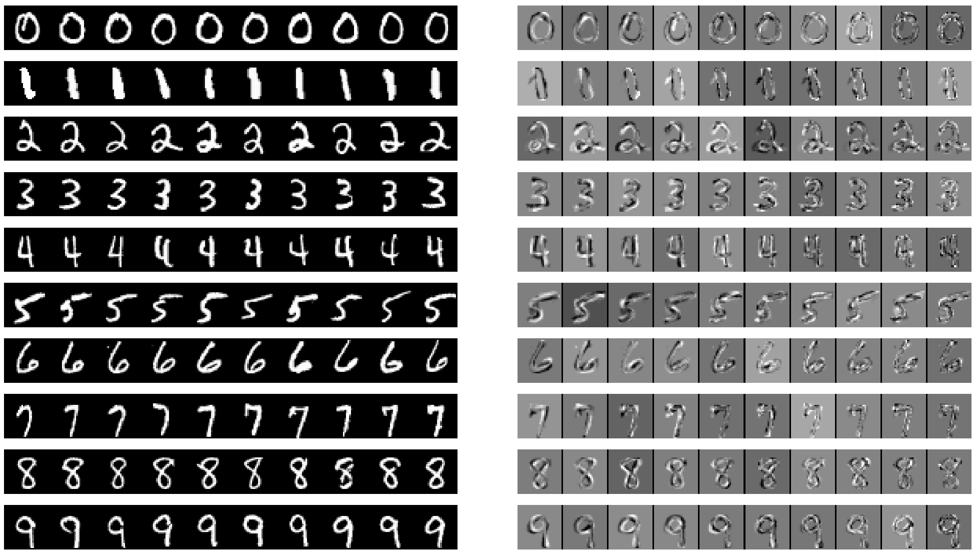


Figure 1.1 – **PCA Locale:** Sur MNIST, pour un exemple de chaque classe (chiffres de 0 à 9) on extrait les 10 plus proches voisins (ppv) intra-classe et on entraîne une PCA sur cet ensemble. Les 10 ppv sont présentés à gauche et les composantes des 10 différentes PCA à droite. Les premières composantes de la PCA contiennent des informations utiles (de possibles transformations) tandis que les dernières n'encodent que du bruit.

Suivant la dimension des données ou le nombre d'exemples d'entraînement, il existe deux algorithmes différents qui permettent d'extraire en $O(\min(n, d)^3)$ (Bishop, 2006). Le cube est dû à l'inversion de matrice.

Hyperparamètres Les hyperparamètres de la PCA comprennent le nombre k de composantes à garder pour la transformation (le nombre de colonnes de V à conserver) et le fait d'effectuer ou non de la décorrélation. Souvent, les premières composantes contiennent une information utile tandis que les dernières composantes représentent du bruit que le processus de décorrélation risque d'amplifier. Notez qu'on peut également appliquer une PCA locale, c'est-à-dire localement sur un sous-ensemble de voisins d'un point (voir Figure 1.1).

1.7.2 Architectures de type Auto-Encodeur

Nous allons maintenant présenter des méthodes non supervisées non linéaires qui peuvent être utilisées par la suite pour l'initialisation de réseaux de neurones. Depuis la formalisation la plus classique de l'auto-encodeur **(AE)** (Gallinari et al., 1987) , nous présenterons le *Denoising Auto-Encoder* **(DAE)** (Vincent et al., 2008, 2010). Ensuite, nous présenterons le *Contractive Auto-Encoder* **(CAE)** (Rifai et al., 2011) et sa version d'ordre supérieur, le *Higher-Order Contractive Auto-Encoder* **(CAE+H)** (Rifai et al., 2011).

L'auto-encodeur classique se décompose en un encodeur et un décodeur. L'encodeur le plus commun effectue une transformation affine des données paramétrée par une matrice de poids $W \in \mathcal{M}_{m \times d}(\mathbb{R})$ et un vecteur de biais $b \in \mathbb{R}^m$ où m désigne le nombre d'unités cachées du modèle. Cette transformation est suivie d'une non-linéarité $s : \mathbb{R}^m \to \mathbb{R}^m$. En général, on utilise la fonction sigmoïde logistique $s(x) = 1/(1 + e^{-x})$ mais la tangente hyperbolique est aussi communément utilisée. On a donc la forme globale suivante:

$$h(x) = s(Wx + b) \tag{1.12}$$

Le décodeur tente de reconstruire l'exemple x à partir de la représentation cachée $h(x)$ et peut partager en la même matrice de poids que l'encodeur mais a un vecteur de biais $b^{'} \in \mathbb{R}^d$ qui lui est propre. Il a alors la forme suivante:

$$r(h(x)) = s(W^T h(x) + b^{'}) \tag{1.13}$$

La non-linéarité $s(.)$ est ici optionnelle et dépend de la nature des données ou de la sortie désirée du décodeur par exemple si elle doit être dans l'intervalle $[0, 1]$ ou non.

Algorithme Tous ces paramètres sont ajustés par optimisation *e.g.* descente de gradient stochastique, visant à réduire l'erreur de reconstruction suivante:

$$\mathcal{J}_{\text{AE}} = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}_{train}} \mathcal{L}(x, r(h(x))) \tag{1.14}$$

La fonction de perte $\mathcal{L}$ utilisée est en général, dans le cas de données binaires (ou modélisant des probabilités), l'entropie croisée négative $\mathcal{L}(x, y) = -\sum_{i=1}^{d} x_i \log y_i + (1-x_i)\log(1-y_i)$ ou l'erreur quadratique $\mathcal{L}(x, y) = \|x-y\|_2^2$. Pour les fins d'initialisation de réseaux de neurones de classification, les performances de l'auto-encodeur classique sont largement dépassées par des versions régularisées de l'auto-encodeur classique telles que les variantes d'auto-encodeur débruitant (DAE) ou contractant

(CAE ou CAE+H) que nous expliquerons dans les sections suivantes.

Hyperparamètres Les hyperparamètres de ce genre d'architecture sont malheureusement nombreux. En voici une liste non exhaustive et des exemples de valeurs typiques à raffiner par une recherche des hyperparamètres optimaux: le nombre d'unités cachées $(1,000)$, la fonction d'activation de l'encodeur ou et du décodeur (fonction sigmoïde), la fonction de perte utilisée (erreur quadratique ou entropie croisée), le pas de gradient $(\{10^{-1},\ldots,10^{-6}\})$.

Une fois une certaine expertise pratique acquise avec l'entraînement de ce genre d'architecture, l'espace des hyperparamètres à explorer qui peut paraître exponentiellement grand au premier abord peut se réduire à quelques valeurs. Une astuce consiste à effectuer un échantillonnage plus intelligent d'hyperparamètres (Bergstra and Bengio, 2012) plutôt qu'une recherche par grille classique.

Auto-Encodeur Débruitant

On présente ici l'auto-encodeur débruitant, en anglais *Denoising Auto-Encoder* (DAE) (Vincent et al., 2008). L'idée du DAE est qu'à partir de l'observation bruitée ou partielle d'une entrée on doit être capable de reconnaître (reconstruire) l'entrée originale. À un très haut niveau, on peut penser qu'en tant qu'êtres humains, nous sommes capables de reconnaître des objets partiellement occultés ou aussi bien à partir d'un son et d'une image, nous sommes en mesure de reconnaître un film déjà vu en ne voyant que certaines séquences.

Plutôt que de minimiser la perte classique (Eq. 1.14), on bruite artificiellement l'entrée originale x et on force l'auto-encodeur à reconstruire l'entrée originale à partir de la version bruitée $\tilde{x}$:

$$\mathcal{J}_{\text{DAE}} = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \mathcal{L}(x, r(h(\tilde{x}))) \tag{1.15}$$

Les processus de corruption communément utilisés sont le bruit de masquage (en anglais *masking noise* ou *dropout noise*) paramétré par p la probabilité de mettre à zéro une composante de l'entrée ou le bruit gaussien paramétré par σ où $\tilde{x} = x+\epsilon$ où $\epsilon \sim \mathcal{N}(0,\sigma)$. D'autres types de bruit peuvent être utilisés (voir Vincent et al. (2010) pour une revue).

Hyperparamètres On peut citer ici en plus des hyperparamètres de l'auto-encodeur classique le type de bruit (gaussien ou *masking*) et le niveau de bruit.

Auto-Encodeur Contractant

On présente ici l'auto-encodeur contractant, en anglais *Contractive Auto-Encoder* (Rifai et al., 2011). Contrairement à la PCA qui extrait les directions de variations **globales** présente dans les exemples d'apprentissage, le CAE apprend des directions de variations **locales**. Il suffit d'ajouter de manière explicite dans la perte minimisée Eq. 1.14 une régularisation qui correspond à la norme du jacobien de l'encodeur $h(x)$ par rapport à l'entrée x sur les points de l'ensemble d'entraînement $\mathcal{D}$:

$$\mathcal{J}_{\text{CAE}} = \mathcal{J}_{\text{AE}} + \lambda \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \| \frac{\partial h}{\partial x}(x) \|_2^2 \tag{1.16}$$

Cette approche suggère aussi la possibilité de visualiser dans l'espace d'entrée quelles sont les directions de variation autour d'un exemple d'apprentissage auxquelles la représentation h est sensible (en regardant les vecteurs propres du jacobien au point d'apprentissage). Cela permet d'évaluer d'un point de vue qualitatif si l'extraction de caractéristiques a convergé vers une solution intéressante ou non (voir Figure 1.2).

Hyperparamètres Contrairement au DAE, le CAE n'a ici qu'un seul hyperparamètre en plus comparé à l'auto-encodeur classique. C'est le coefficient de régularisation λ qui va contrôler le compromis entre erreur de reconstruction et invariance de la représentation.

Auto-Encodeur Contractant d'Ordre Supérieur

Si le fait de pénaliser le premier ordre de la dérivée (Jacobien) de l'encodeur par rapport à l'entrée est bénéfique, il paraît naturel d'ajouter une pénalisation concernant les ordres supérieurs (Hessien, troisième ordre, ...) au CAE original (Eq. 1.18). C'est l'idée présente derrière l'auto-encodeur contractant d'ordre supérieur, en anglais *Higher-order Contractive Auto-Encoder* (CAE+H) (Rifai et al., 2011). Une

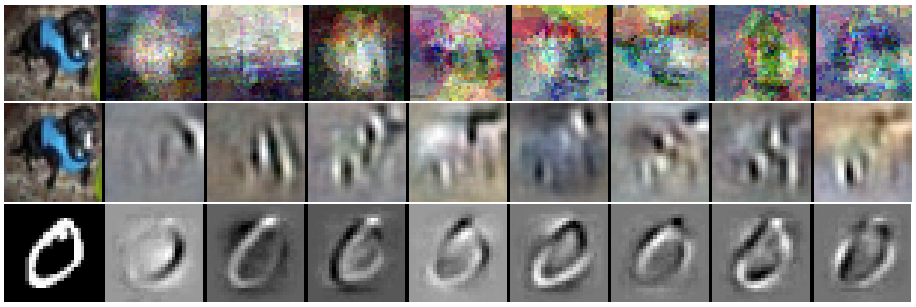


Figure 1.2 – Sur MNIST et sur CIFAR (ensemble de chiffres manuscrits/images RGB), on présente les tangentes apprises par le CAE+H autour d'un point d'apprentissage au milieu pour CIFAR et bas pour MNIST. On peut voir que ces tangentes encodent de l'information sur des transformations plausibles de l'image originale tandis que les composantes principales de la PCA sur CIFAR présentées en haut, n'ont encodé que du bruit. Pour observer les composantes de la PCA sur MNIST, nous référons à la Figure 1.1

fois cette pénalité ajoutée lors de l'entraînement, on pourra constater les effets obtenus en termes de mesures quantitatives de performances de classification (bénéfique ou non) ou de directions de variations obtenues (qualitatif).

Les pertes des Eq. 1.14, 1.15 ou 1.18 sont minimisées par descente de gradient. Il est donc nécessaire de pouvoir calculer les dérivées partielles de la perte $\mathcal{J}$ par rapport aux paramètres θ. Pour commencer, le calcul de la $k^{\text{ième}}$ dérivée de l'encodeur (avec m unités cachées) par rapport à l'entrée (de dimension d) a une complexité algorithmique de $O(md^k)$. Calculer les dérivées partielles de cette dérivée par rapport aux paramètres du modèle a un coût prohibitif. Pour éviter ce calcul, on utilise donc une approximation stochastique de la Hessienne (voir Rifai et al. (2011) pour la preuve):

$$\|\frac{\partial^2 h}{\partial x^2}(x)\|_2^2 = \lim_{\sigma \to 0} \frac{1}{\sigma^2} \int_{\epsilon \rightsquigarrow \mathcal{N}(0,\sigma^2)} \|\frac{\partial h}{\partial x}(x) - \frac{\partial h}{\partial x}(x+\epsilon)\|_2^2 d\epsilon \tag{1.17}$$

Si la variance σ^2 est non nulle, des contributions aux normes des ordres supérieurs apparaissent (Rifai et al., 2011). Ces contributions disparaissent à la limite $\sigma \to 0$. En pratique pour approximer la norme de la Hessienne (Eq. 1.17), on utilise un lot de plusieurs versions bruitées $x + \epsilon$ du même exemple x avec un σ petit mais non négligeable, ce qui donne lieu à l'approximation stochastique de la norme de la Hessienne mais qui comporte aussi d'autres contributions provenant des normes des ordres supérieurs (voir Rifai et al. (2011) pour plus de détails).

L'avantage d'utiliser cette approximation est que l'on peut pénaliser tous les ordres supérieurs à un coût moindre comparé à celui d'une complexité en $O(md^k)$. L'inconvénient est de ne plus savoir exactement quels ordres sont pénalisés et quelle est la contribution de chacun des ordres qui permet d'apprendre de meilleures caractéristiques.

En utilisant cette approximation, on aboutit à la perte du CAE+H avec une taille de lot n_c et $\epsilon_i \rightsquigarrow \mathcal{N}(0, \sigma)$:

$$\mathcal{J}_{\text{CAE+H}} = \mathcal{J}_{\text{CAE}} + \gamma \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \frac{1}{n_c} \sum_{i=1}^{n_c} \| \frac{\partial h}{\partial x}(x) - \frac{\partial h}{\partial x}(x + \epsilon_i) \|_2^2 \tag{1.18}$$

Hyperparamètres Une des difficultés pratiques avec le CAE+H est son nombre d'hyperparamètres. En plus des hyperparamètres du CAE, on a ici le coefficient de régularisation pour les ordres supérieurs γ, le niveau de bruit gaussien σ et la taille du mini-lot n_c utilisés pour l'estimation de la norme de la Hessienne.

Autres Méthodes d'Extraction de Caractéristiques

Il existe de nombreuses autres méthodes d'extraction de caractéristiques qui n'ont pas été détaillées ici. Comme méthodes d'extraction de caractéristiques linéaires, on trouve par exemple l'analyse en composantes indépendantes, en anglais *Independent Component Analysis* (ICA) (Comon, 1994; Hyvärinen et al., 2001). Cette méthode peut être utilisée pour la séparation aveugle de sources.

Deux types de méthodes d'extraction de caractéristiques non linéaire ont aussi fait l'objet de récentes avancées: les méthodes **sparses** (Ranzato et al., 2008; Kavukcuoglu et al., 2008, 2009) qui utilisent une pénalité sur les activations de l'encodeur (afin de le pousser à prendre la valeur 0 - inactif - dans le cas d'une sigmoïde) ou les modèles **génératifs** basés sur des fonctions d'énergie (Ranzato et al., 2007) comme les machines de Boltzmann restreintes, en anglais *Restricted Boltzmann Machine* (RBM) (Tieleman, 2008). Ces méthodes ont été utilisées avec succès pour l'initialisation de réseaux de neurones de type MLP (Hinton, 2006; Ranzato et al., 2008; Kavukcuoglu et al., 2008, 2009) ou réseaux convolutionnels (Kavukcuoglu et al., 2010).

1.7.3 Déplacement sur la Variété

D'après "l'hypothèse de variété", les distributions de données naturelles en haute dimension seraient concentrées le long de variétés de plus faible dimension. En observant les vecteurs propres $V_{.i}(x)$ d'une PCA apprise à partir de l'ensemble $\mathcal{D}_{10-ppv}(x)$ des 10 plus proches voisins d'un point x (Figure 1.1), on constate une similitude avec des potentielles directions de variations autour de ce point central x. Ces directions forment un repère local orthonormé centré en ce point $\mathcal{B}_x = \{V_{.i}(x) \mid U_{i,i} > \epsilon\}$ (voir Notation 1.7.1). À partir de ces directions, on peut former un hyperplan tangent local $\mathcal{H}_x = \{x + v \mid v \in \text{span}(\mathcal{B}_x)\}$ à ce point x (Rifai et al., 2011).

L'hyperplan tangent local $\mathcal{H}_x$ est un nouvel ensemble de points localement proches de la variété le long de laquelle se concentrent les exemples d'apprentissage au voisinage du point x. À partir de cette idée, on peut imaginer naviguer sur la variété de proche en proche (d'un point suffisamment proche d'un autre) en suivant ces directions de variations et en "reprojetant" au besoin sur la variété à l'aide d'un auto-encodeur qui la modélise.

Si on suppose que la variété peut être approximée localement de façon linéaire entre deux points suffisamment proches, cela évite d'essayer de "connecter" deux points x_i et x_j par des directions provenant des hyperplans tangents $\mathcal{H}_{x_i}$ et $\mathcal{H}_{x_j}$. On effectue plutôt le mouvement par une simple interpolation linéaire entre les deux points voisins x_i et x_j (Mesnil et al., 2012).

$$x_{i,j,\alpha} = x_i + \alpha(x_j - x_i) \tag{1.19}$$

avec $\alpha \in]0, 1[$. Il est aussi possible d'effectuer une approximation non linéaire de la variété si cette interpolation a lieu dans l'espace des représentations $h(x)$ et que l'on revient dans l'espace original des x par une simple reconstruction du décodeur. Avec la notation de l'équation 1.13, on obtient:

$$x_{i,j,\alpha} = r(h(x_i) + \alpha(h(x_j) - h(x_i)))) \tag{1.20}$$

Dans le Chapitre 8, on montrera qu'il est plus aisé de se déplacer sur la variété à partir des représentations $h(x)$. Nous montrerons également qu'il est plus facile de mixer (de passer d'une classe à l'autre) lors du processus d'échantillonnage pour les CAEs ou RBMs si l'échantillonnage a lieu dans l'espace des représentations apprises $h(x)$ plutôt que dans l'espace original des données x.

1.7.4 Transfert de Domaine: PCA Transductive et Finetuning

Dans les sections précédentes, nous avons toujours supposé que les données dans $\mathcal{D}_{train}$, $\mathcal{D}_{valid}$ et $\mathcal{D}_{test}$ provenaient de la même distribution. Dans le paradigme du transfert de domaine, cette hypothèse n'est plus valable. Cependant, ce n'est pas pour autant qu'il faut reconstruire les ensembles de données afin de satisfaire cette contrainte de distribution identique. Il est plus intelligent de proposer un apprentissage transductif afin de pouvoir transférer nos connaissances apprises à partir de $\mathcal{D}_{train}$ sur les ensembles $\mathcal{D}_{valid}$ et $\mathcal{D}_{test}$. Le transfert d'apprentissage transductif suppose que la distribution sous-jacente à $\mathcal{D}_{valid}$ et $\mathcal{D}_{test}$ partage de manière limitée ou non le support de la distribution sous-jacente à $\mathcal{D}_{train}$. La notion de support partagé de l'apprentissage transductif suppose que les variations des exemples dans $\mathcal{D}_{valid}$ ou $\mathcal{D}_{test}$ sont un sous-ensemble des variations de $\mathcal{D}_{train}$. Nous pouvons imaginer un exemple comme nous verrons dans le chapitre 4 où $\mathcal{D}_{train}$ contiendrait toutes les images de chiffres $\{0, \ldots, 9\}$ tandis que l'ensemble $\mathcal{D}_{valid}$ contiendrait seulement les chiffres $\{0, 1, 2\}$ et $\mathcal{D}_{test}$ les chiffres $\{5, 6, 7\}$.

Pour une revue complète des différents types de transfert d'apprentissage (inductif, non supervisé et transductif), on réfère le lecteur à la revue de littérature sur ce sujet par (Pan and Yang, 2010).

PCA Transductive

La PCA transductive n'est pas entraînée à partir de $\mathcal{D}_{train}$. Une fois que le processus d'apprentissage non supervisé de $h(x)$ sur l'ensemble $\mathcal{D}_{train}$ est achevé, on entraîne la PCA transductive sur la représentation apprise $h(x)$ avec $x \in \mathcal{D}_{valid}$ ou $x \in \mathcal{D}_{test}$. On entraîne une PCA par ensemble, il y aura donc deux PCA à apprendre dans ce cas. En comparant la notation dans la Section 1.7.1, cela revient à remplacer X par $\{h(x) \mid x \in \mathcal{D}_{valid}\}$ ou $\{h(x) \mid x \in \mathcal{D}_{test}\}$ suivant l'ensemble pour lequel on veut obtenir la représentation.

La PCA transductive a pour objectif de ne retenir que les variations dominantes dans la représentation des ensembles $\mathcal{D}_{valid}$ ou $\mathcal{D}_{test}$. Le classifieur utilisé à partir de la représentation $h(x)$ pourra ainsi ignorer les variations présentes dans $\mathcal{D}_{train}$ sans intérêt pour $\mathcal{D}_{valid}$ ou $\mathcal{D}_{test}$ pour ne se concentrer que sur celles présentes dans les ensembles $\mathcal{D}_{valid}$ ou $\mathcal{D}_{test}$. On peut assimiler ces variations "en trop" à du bruit. La représentation $h(x)$ conserve donc toutes les variations jusqu'à la PCA transductive apprise à partir de $h(x)$ qui elle, ne se concentre que sur les variations spécifiques à l'ensemble de données en question, $\mathcal{D}_{valid}$ ou $\mathcal{D}_{test}$. Cette PCA transductive permet alors un transfert de domaine de $\mathcal{D}_{train}$ vers $\mathcal{D}_{valid}$ ou $\mathcal{D}_{test}$.

Finetuning

Il est possible d'appliquer cette notion d'apprentissage transductif à un réseau de neurones. L'idée est ici d'effectuer un apprentissage non supervisé sur un ensemble $\mathcal{D}_{big-unlabeled}$ pour ensuite procéder à un apprentissage supervisé par **finetuning** sur un ensemble $\mathcal{D}_{small-labeled}$. On effectue alors un transfert de domaine de $\mathcal{D}_{big-unlabeled}$ vers $\mathcal{D}_{small-labeled}$.

En pratique, nous disposons d'une masse de données non étiquetées considérable sur le web. Lorsque notre ensemble de données étiquetées $\mathcal{D}_{small-labeled}$ est de quatité réduite, il peut être intéressant d'apprendre une représentation de façon non supervisée sur un grand ensemble de données non étiquetées $\mathcal{D}_{big-unlabeled}$ partageant les mêmes structures que les données de $\mathcal{D}_{small-labeled}$, par exemple ces données sont toutes des images d'objets. Par la suite cette représentation est utilisée comme point d'initialisation pour un apprentissage supervisé d'un réseau de neurones profond. On peut alors parler de transfert de connaissances apprises sur un grand ensemble de données $\mathcal{D}_{big-unlabeled}$ à un plus petit ensemble $\mathcal{D}_{small-labeled}$. Nous verrons que cela permet d'obtenir des gains significatifs de performance pour la classification d'images de scènes (Chapitre 6).

1.8 Classification de Séquences

Les réseaux de neurones peuvent aussi être utilisés pour la classification de séquences, on utilise en général dans ce cas des réseaux de neurones récurrents. Dans le cadre de l'apprentissage supervisé simple (Section 1.6), nos ensembles $\mathcal{D}_{train}$, $\mathcal{D}_{valid}$ et $\mathcal{D}_{test}$ sont composés des entrées $x \in \mathbb{R}^d$ et une étiquette $y \in \{1, \ldots, C\}$ qui leur est associé. Pour la classification de séquences, les données sont séquentielles et pour une séquence de longueur T, on a $x = \{x_i \in \mathbb{R}^d\}_{i=1}^T$ et $y = \{y_i \in \{1, \ldots, C\}_{i=1}^T\}$.

En traitement du langage naturel par exemple (Chapitre 10), chaque entrée est une séquence de mots et l'étiquette y est constitué d'une séquence de classes associées à chacun des mots (Figure 10.1). À noter que la longueur T de la séquence n'est pas uniforme dans l'ensemble de données et peut varier.

1.8.1 Champs Aléatoires Conditionnels

Une des méthodes de classification de séquences les plus utilisées sont les champs aléatoires conditionnels, en anglais *Conditional Random Fields* (CRF) (Lafferty et al., 2001) et leurs variantes. Un CRF est un modèle graphique non dirigé composé d'un ensemble de sommets $\boldsymbol{X}$ et $\boldsymbol{Y}$ qui correspondent aux variables observées et aux variables de sortie. Ces deux ensembles sont disjoints et le modèle graphique permet d'estimer la probabilité conditionnelle $\mathbb{P}(\boldsymbol{Y}|\boldsymbol{X})$.

La probabilité conditionnelle de modéliser une sortie y conditionnellement à x pour une chaîne linéaire s'écrit sous la forme suivante:

$$f_\theta(y \mid x) = \frac{1}{Z(x)} \prod_{t=1}^{T} \exp H(y_{t-1}, y_t, x_t) \tag{1.21}$$

avec $Z(x)$ un facteur de normalisation et les M fonctions $h_m(y_{t-1}, y_t, x_t)$ "caractéristique" sont définies de la manière suivante:

$$H(y_{t-1}, y_t, x_t) = \sum_{m=1}^{M} \lambda_m h_m(y_{t-1}, y_t, x_t) \tag{1.22}$$

Les fonctions h_m sont fixées au départ et l'apprentissage consiste à ajuster les λ_m. Une fonction caractéristique peut prendre la forme suivante par exemple:

$$h_m(y_{t-1}, y_t, x_t) = \begin{cases} 1 \text{ si } x_t = \text{"september"}, \ y_{t-1} = \text{"chiffre" et } y_t = \text{"mois"} \\ 0 \text{ sinon} \end{cases} \tag{1.23}$$

L'apprentissage s'effectue par l'ajustement des paramètres $\theta = \{\lambda_m\}_{m=1}^M$ pour maximiser la log-vraisemblance sur l'ensemble $\mathcal{D}_{train}$:

$$\mathcal{J}_{\text{CRF}} = \sum_{(x,y)\in\mathcal{D}_{train}} \sum_{t=1}^{T} \log(\frac{1}{Z(x_t)}) + \sum_{m=1}^{M} \lambda_m h_m(y_{t-1}, y_t, x_t) \tag{1.24}$$

L'équation 1.24 est différentiable par rapport à $\theta = \{\lambda_m\}_{m=1}^{M}$ et concave. Il n'y a donc qu'un seul maximum et on peut l'approcher par des méthodes à base de gradient.

1.8.2 Réseaux de Neurones Récurrents

Les réseaux de neurones récurrents (RNN) sont des architectures profondes. Si on compare aux architectures profondes habituelles à m couches, la rétropropagation du gradient au travers des m couches revient à effectuer une rétropropagation dans le temps pour m unités de temps dans le passé.

L'architecture de réseau récurrent de base (Elman, 1990) garde une trace des représentations du passé au travers de ses connexions récurrentes. La représentation $h_t(x)$ peut donc être vue comme un résumé des états précédents $\{h_k(x)\}_{k=0}^{t-1}$.

Il existe plusieurs architectures de réseaux de neurones récurrents possibles (Elman, Jordan, unidirectionnel, bidirectionnel) (Mesnil et al., 2013a), nous décrivons ici la forme la plus simple. Pour un RNN de type Elman pour une séquence d'entrées $\{x_i \in \mathbb{R}^d\}_{i=1}^{T}$ on a la représentation $\{h_i(x) \in \mathbb{R}^{d_h}\}_{i=1}^{T}$ à d_h unités cachées suivante:

$$h_t(x) = s(Ux_t + Vh_{t-1}(x)) \tag{1.25}$$

paramétré par $U \in \mathcal{M}_{d_h \times d}(\mathbb{R})$, $V \in \mathcal{M}_{d_h \times d_h}(\mathbb{R})$ et $h_0 \in \mathbb{R}^{d_h}$.

Pour effectuer la prédiction, on ajoute la couche supérieure suivante similaire à l'équation 1.8:

$$f_t(x) = \frac{e^{h_t(x)+b}}{\sum_{j=1}^{C} e^{W_{j.}h_t(x)+b_j}} \in [0,1]^C \tag{1.26}$$

Pour entraîner le modèle, on effectue une minimisation par descente de gradient de la log-vraisemblance négative pour chaque séquence de $\mathcal{D}_{train}$:

$$\mathcal{J}(\theta) = -\sum_{(x,y)\in\mathcal{D}_{train}} \sum_{t=1}^{T} \log f_t(x_t)_{y_t} \tag{1.27}$$

pour les paramètres $\theta = \{U, V, W, b, h_0\}$. Pour obtenir l'équation 1.27 précédente, l'hypothèse d'indépendance conditionnelle des y_i sachant x a été faite.

2 Espaces Sémantiques Multidomaines

Cette dissertation a pour objet l'apprentissage d'espaces sémantiques pour différents domaines comme par exemple, le langage naturel ou les images. Il est aussi possible d'apprendre un espace sémantique à l'intersection de domaines, on parle en ce cas d'espace sémantique multidomaine (Chapitre 12). Ces espaces $\mathbb{E} \subset \mathbb{R}^d$ sont appelés sémantiques si la représentation $h(x) \in \mathbb{E}$ d'entités x, jugées similaires par un esprit humain, ont leur similitude préservée dans cet espace $\mathbb{E}$. Cette similitude peut se mesurer à l'aide d'une métrique comme la distance euclidienne par exemple.

Dans le chapitre 1, nous avons présenté comment apprendre des représentations caractérisant des espaces sémantiques au travers de l'apprentissage non supervisé. Nous avons aussi observé dans la section 1.7.3 comment se déplacer dans cet espace. Dans ce chapitre, on décrira comment obtenir une représentation vectorielle dans un espace euclidien pour chaque entité d'un même domaine, ou de domaines différents.

Ce court chapitre introduit brièvement les quelques notions essentielles ainsi que quelques approches simples, spécifiques aux espaces sémantiques, qui permettront au lecteur de mieux appréhender les articles qui suivent.

2.1 Espace Sémantique pour le Langage

La représentation habituelle de mots ou de séquences de mots utilisée pour les tâches de traitement du langage naturel est un vecteur de haute dimension avec des bits actifs (ou des comptes) pour les mots ou N-grams présents dans l'entrée. Ces vecteurs contiennent une majorité de zéros et il est possible d'utiliser des structures de données adaptées d'un point de vue computationnel informatique.

Une autre approche pour traiter ce problème consiste à associer une représentation vectorielle $h(\text{mot}) \in \mathbb{E}$ à chacun des M mots du dictionnaire. Cette représentation est contenue dans une matrice $E \in \mathcal{M}_{M \times d}(\mathbb{R})$ où chaque ligne représente un *embedding* de mot. Si on définit une bijection $g(\text{mot}) \in \{0, 1, \ldots, M\}$ qui associe un index i à chaque mot, on a alors notre fonction $h(\text{mot}) = E_{g(\text{mot}).}$ qui retourne la ligne correspondante de la matrice $E_{i,.}$

Mathématiquement, il s'agit d'un produit matriciel *one-hot*("mot")$^T E \in \mathbb{E}$ entre un vecteur *one-hot* (Eq. 2.1) et la matrice des embeddings E. Cette opération étant différentiable, la mise à jour de la matrice des embeddings se fait sans problème au travers de l'algorithme de rétropropagation du gradient, comme une couche additionnelle à notre réseau de neurones.

$$\text{one-hot}(\text{"alice"}) = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \text{L'index du mot "alice" donné par } g \tag{2.1}$$

Il est possible d'aller plus loin. Cette représentation de séquences de mots peut aussi être utilisée à l'intérieur des mots eux-mêmes. Les mots sont des séquences de lettres ou de phonèmes. Si on associe des *embedding* à des 3-Gram de lettres ou de phonèmes, il devient possible d'avoir une représentation des mots à partir de leur structure propre. Cette approche (Shen et al., 2014) a donné de bons résultats en pratique pour la recherche d'information (Information Retrieval).

2.2 Mesure de Similarité

Il est important dans cet espace $\mathbb{E}$ de pouvoir juger de la qualité des représentations apprises. Il n'existe pas encore de mesure consensuelle pour départager une représentation meilleure qu'une autre. On peut en revanche examiner les entités voisines dans l'espace sémantique $\mathbb{E}$ et juger si elles sont similaires. On peut mesurer la similarité entre la représentation de deux entités $e^{(i)} = h(x_i)$ et $e^{(j)} = h(x_j)$ dans $\mathbb{E}$ par un simple produit scalaire:

$$\text{sim}(e^{(i)}, e^{(j)}) =< e^{(i)}, e^{(j)} >= \sum_{k=1}^{d} e_k^{(i)} e_k^{(j)} \tag{2.2}$$

Une autre mesure de similarité qui peut aussi donner d'excellents résultats est la distance cosinus fréquemment utilisée en traitement du langage pour la comparaison de documents textuels.

$$\text{sim}(e^{(i)}, e^{(j)}) = \cos(e^{(i)}, e^{(j)}) = \frac{< e^{(i)}, e^{(j)} >}{\|e^{(i)}\| \|e^{(j)}\|} \tag{2.3}$$

À noter que pour une optimisation par descente de gradient, cette mesure de similarité est plus coûteuse en termes computationnels que le produit scalaire simple (Eq. 2.2).

2.3 Classification en Haute Dimension

Considérons le cas d'une classification classique avec softmax (Eq. 1.8) en négligeant le biais $b^{(m+1)}$. On peut associer l'espace défini par la dernière couche (la couche avant d'appliquer la softmax) à un espace sémantique $\mathbb{E}$. Chaque ligne $W_{j.}^{(m+1)}$ de la matrice $W^{(m+1)}$ du classifieur correspond au centre du prototype associé à la classe j. Pour prédire la classe dominante associée à un exemple x, on calcule la représentation $h^{(m)}(x)$ et la classe j avec la plus grande similarité l'emporte:

$$\arg \max_{j \in \{1,\ldots,C\}} \text{sim}(h^{(m)}(x), W_{j.}^{(m+1)}) \tag{2.4}$$

La softmax est appliquée ensuite pour convertir ces valeurs en probabilités.

Prenons l'exemple de la détection de paraphrase dans un espace sémantique (Dolan et al., 2004). On suppose que notre ensemble d'entraînement $\mathcal{D}_{train}$ est composé de $10,000$ couples de paraphrases, donc $20,000$ phrases au total. Au moment de l'apprentissage, on veut avoir la similarité $\text{sim}(h(x_i), h(x_j))$ entre la représentation $h(x_i)$ et $h(x_j)$ de deux paraphrases x_i et x_j supérieure à la similarité de tous les couples qui ne sont pas des paraphrases. Notre fonction de similarité étant symétrique, au total il est possible de former un ensemble $\mathcal{P}$ de $10,000$ couples positifs et un ensemble $\mathcal{N}$ de $199,970,000$ couples négatifs $(20,000^2/2 - 20000 - 10000)$ Le nombre d'entités C (dans l'exemple précédent $C = 20,000$) est en général très grand.

Au moment de l'apprentissage, calculer la similarité pour tous les couples négatifs devient inefficace et il nécessaire de trouver d'autres stratégies pour contrôler l'explosion du nombre de classes considérées comme négatives. Une possibilité est d'échantillonner un nombre acceptable ($100 \leq C \leq 1,000$) de couples négatifs à chaque mise à jour afin d'obtenir une approximation de la vraie valeur de la softmax. Il est aussi possible de faire de l'échantillonnage de négatif intelligent, c'est-à-dire sélectionner uniquement les couples négatifs qui violent une certaine contrainte et ignorer les autres. Pour un couple de positif (x_i, x_j), cette contrainte pour sélectionner un ensemble de couples négatifs (x_k, x_l) peut prendre la forme d'une violation de marge ϵ:

$$\text{sim}(x_i, x_j) - \text{sim}(x_k, x_l) - \epsilon > 0 \tag{2.5}$$

Il est ensuite plus aisé d'avoir une solution pour un problème à grande échelle, c'est-à-dire pour plusieurs millions d'entités. On sélectionne de façon aléatoire un sous-ensemble $\tilde{\mathcal{N}}_{(x_i,x_j)}$ de couples négatifs associé à chaque mise à jour d'un couple positif $(x_i, x_j) \in \mathcal{P}$. Ce sous-ensemble $\tilde{\mathcal{N}}$ est renouvelé à chaque mise à jour. La fonction à minimiser devient alors:

$$-\sum_{(x_i,x_j)\in\mathcal{P}} \sum_{(x_k,x_l)\in\tilde{\mathcal{N}}_{(x_i,x_j)}} \max(0, \text{sim}(h(x_i), h(x_j)) - \text{sim}(h(x_k), h(x_l)) - \epsilon) \tag{2.6}$$

Nous présenterons cette technique (Weston et al., 2011) plus en détails dans le chapitre 12. Il est possible de s'inspirer de ces idées pour avoir d'autres types de fonctions à minimiser mais le principe reste le même (Shen et al., 2014).

2.4 Espaces Multidomaines

Dans la section précédente 2.3, nous avons observé comment apprendre un espace sémantique pour un seul et unique domaine. Il est possible d'apprendre un espace sémantique commun à l'intersection de différents domaines.

Pour étendre cette idée à deux domaines, par exemple les domaines du langage et des images, il suffit de définir une représentation $h_L(x)$ pour le langage et une représentation $h_I(x)$ pour des images (Chapitre 12). La seule différence par rapport à l'utilisation de couples de phrases présenté dans la section précédente réside dans une représentation h_I additionnelle à apprendre. La fonction à minimiser reste la même:

$$-\sum_{(x_i,x_j)\in\mathcal{P}}\ \sum_{(x_k,x_l)\in\tilde{\mathcal{N}}_{(x_i,x_j)}} \max(0, \mathrm{sim}(h_L(x_i), h_I(x_j)) - \mathrm{sim}(h_L(x_k), h_I(x_l)) - \epsilon) \quad (2.7)$$

Cette approche est bénéfique dans le sens où une prédiction d'image de "requin" à partir d'embeddings de mots même si elle est fausse, donnera une prédiction proche parmi des embeddings voisins qui auront la même sémantique, par exemple "poisson", "thon" (Bengio, 2013).

Nous travaillons actuellement à étendre le concept multidomaine de deux domaines à un nombre de domaines arbitraire. Le nombre de couples positifs croît de façon quadratique avec le nombre de domaines. Une approche avec une croissance linéaire est actuellement à l'étude.

2.5 Apprentissage par Procuration

Il y a plusieurs problèmes qui de par leur formulation, ne peuvent pas être directement résolus du fait d'un manque de données. Par exemple, considérons le problème d'identifier des personnes à partir d'une image de leur visage (Bottou, 2011). Il est impossible de récolter un ensemble de données suffisamment grand, contenant suffisamment de variations de poses et contextes, pour résoudre ce problème pour chaque personne par une approche naïve de classification multiclasse directe. En revanche, il est possible de récolter des données pour un problème légèrement différent: deux visages dans deux images représentent-ils la même personne ? Il est possible de trouver énormément de données pour résoudre ce problème: deux visages dans une même image représentent sûrement deux personnes différentes, et deux visages dans des images successives d'une vidéo sont probablement les mêmes. Ces deux tâches partagent beaucoup de points communs mais la formulation sous la forme du second problème permet sans doute d'apporter une solution effective au premier problème. L'apprentissage d'une solution pour le second problème constitue un **apprentissage par procuration** d'une solution au premier problème.

Dans le chapitre 12, nous proposerons une solution au problème de la classification d'un objet et de parties de cet objet ; par exemple une voiture et son pare-brise, ses roues et ses rétroviseurs. Ce problème manque de données étiquetées mais nous verrons comment utiliser un espace sémantique multidomaine pour effectuer un transfert de l'apprentissage dans le même esprit que l'exemple énoncé ci-dessus. Cet apprentissage par procuration au travers de l'intersection d'un espace sémantique de l'image avec l'espace sémantique du langage nous permettra de générer un ensemble de données effectif de 10 millions d'exemples et ainsi d'obtenir une solution au problème initial.

3 Préambule au Premier Article

Dans ce premier article, nous étudions plusieurs algorithmes d'apprentissage non supervisé de représentation. L'espace de représentation est ici notre espace sémantique. Chaque dimension dans cet espace sémantique encode un degré d'expression d'une combinaison de caractéristiques extraite automatiquement à partir de l'espace d'entrée. Une classification à l'aide d'un hyperplan dans cet espace de représentation sémantique est plus performante que dans l'espace d'entrée original ce qui nous a permis de remporter la compétition "Unsupervised and Transfer Learning Challenge" en 2011.

3.1 Détails de l'article

Unsupervised and Transfer Learning Challenge: a Deep Learning Approach Grégoire Mesnil, Yann Dauphin, Xavier Glorot, Salah Rifai, Yoshua Bengio, Ian Goodfellow, Erick Lavoie, Xavier Muller, Guillaume Desjardins, David Warde-Farley, Pascal Vincent, Aaron Courville and James Bergstra. *Journal of Machine Learning Research: Proceedings of the Unsupervised and Transfer Learning Challenge and workshop*, pages 97 – 110, 2012

Contribution Personnelle Cette compétition faisait partie du travail pratique d'un cours de Yoshua Bengio. Yann Dauphin, Xavier Glorot, Salah Rifai et moi-même avons décidé de former un noyau dur pour remporter la compétition (plus de 800 entrées à nous 4 sur une courte période avec plusieurs centaines de compétiteurs). J'ai proposé plusieurs solutions décisives pour cette victoire comme la PCA transductive, et plusieurs de mes entrées nous ont permis de remporter la première place. Une partie de mon code a été mis à disposition des autres étudiants sous forme de tutoriel afin de leur permettre de participer et d'utiliser notre grappe de calcul. Par la suite, j'ai dirigé la rédaction de l'article en collaboration avec tous les coauteurs et préparé la présentation donnée par Yann Dauphin lors de la conférence ICML 2012. La publication a remporté le PASCAL 2 UTLC best paper award.

3.2 Contexte

À cette époque, l'apprentissage de représentations non supervisé commence à prendre son essor mais de nombreux scientifiques sont encore suspicieux vis-à-vis des architectures profondes à réseaux de neurones. Cette victoire a contribué à montrer que ces techniques pouvaient rivaliser avec d'autres méthodes à l'état de l'art comme les SVMs.

3.3 Contributions

Plusieurs des techniques présentées dans cet article ont par la suite obtenu des publications dans des journaux ou des conférences internationales comme le S3C (Courville et al., 2011) ou le CAE (Rifai et al., 2011). Le pipeline présenté ici a aussi été utilisé pour obtenir une meilleure représentation avec une réduction de dimensionnalité conséquente dans le second article (Chapitre 6) de cette dissertation.

3.4 Récents développements

Avec la récente mise en avant de l'apprentissage supervisé et de ses succès, l'apprentissage non supervisé est un peu mis en retrait même si cela reste le Graal pour de nombreux chercheurs étant donné la quantité massive de donnnées non étiquetées disponible via internet. Cela a surtout permis de fédérer une équipe de recherche avec qui j'ai pu travailler sur d'autres projets. Dans le futur, j'espère pouvoir continuer à perpétuer ces riches collaborations.

4 Unsupervised and Transfer Learning Challenge: a Deep Learning Approach

4.1 Introduction

The objective of machine learning algorithms is to discover statistical structure in data. In particular, *representation-learning algorithms* attempt to transform the raw data into a form from which it is easier to perform supervised learning tasks, such as classification. This is particularly important when the classifier receiving this representation as input is linear and when the number of available labeled examples is small. This is the case here with the Unsupervised and Transfer Learning (UTL) Challenge[1].

Another challenging characteristic of this competition is that the training (development) distribution is typically very different from the test (evaluation) distribution, because it involves a set of classes different from the test classes, i.e., both inputs and labels have a different nature. What makes the task feasible is that these different classes have things in common. The bet we make is that *more abstract features of the data are more likely to be shared among the different classes*, even with classes which are very rare in the training set. Another bet we make with representation-learning algorithms and with Deep Learning algorithms in particular is that the structure of the input distribution $P(X)$ is strongly connected with the structure of the class predictor $P(Y|X)$ for all of the classes Y. It means that representations $h(X)$ of inputs X are useful both to characterize $P(X)$ and to characterize $P(Y|X)$, which we will think of as parametrized through $P(Y|h(X))$. Another interesting feature of this competition is that the input features are anonymous, so that teams are compared based on the strength of their learning algorithms and not based on their ability to engineer hand-crafted features based on task-specific prior knowledge. More material on Deep Learning can be found in a companion paper (Bengio, 2011).

1. `http://www.causality.inf.ethz.ch/unsupervised-learning.php`

The paper is organized as follows. The pipeline going from bottom (raw data) to top (final representation fed to the classifier) is described in 4.2. In addition to the score returned by the competition servers, 4.3 presents other criteria that guided the choice of hyperparameters. 6.5.2 precisely describes the layers we chose to combine for each of the five competition datasets, at the end of the exploration phase that lasted from January 2011 to mid-April 2011.

4.2 Method

We obtain a deep representation by stacking different single-layer blocks, each taken from a small set of possible learning algorithms, but each with its own set of hyper-parameters (the most important of which is often just the dimension of the representation). Whereas the number of possible combinations of layer types and hyper-parameters is exponential as depth increases, we used a greedy layer-wise approach (Bengio et al., 2007a) for building each deep model. Hence, the first layer is trained on the raw input and its hyper-parameters are chosen with respect to the score returned by the competition servers (on the validation set) and different criteria to avoid overfitting to a particular subset of classes (discussed in 4.3). We then fix the i^{th} layer (or keep only a very small number of choices) and search for a good choice of the $i+1^{th}$ layer, pruning and keeping only a few good choices. Depth is thus increased without an explosion in computation until the model does not improve significantly the performance according to our criteria.

The resulting learnt pipeline can be divided in three types of stages: preprocessing, feature extraction and transductive postprocessing.

4.2.1 Preprocessing

Before the feature extraction step, we preprocessed the data using various techniques. Let $\mathcal{D} = \{x^{(j)}\}_{j=1,\dots,n}$ be a training set where $x^{(j)} \in \mathbb{R}^d$.

Standardization One option is to standardize the data. For each feature, we compute its mean $\mu_k = (1/n)\sum_{j=1}^{n} x_k^{(j)}$ and variance σ_k. Then, each transformed feature $\tilde{x}_k^{(j)} = (x_k^{(j)} - \mu_k)/\sigma_k$ has zero mean and unit variance.

Uniformization (t-IDF) Another way to control the range of the input is to uniformize the feature values by restricting their possible values to $[0, 1]$ (and non-parametrically and approximately mapping each feature to a uniform distribution). We rank all the $x_k^{(j)}$ and map them to $[0, 1]$ by dividing the rank by the number of observations sorted. In the case of sparse data, we assigned the same range value (0) for zeros features. One option is to aggregate all the features in these statistics and another is to do it separately for each feature.

Contrast Normalization On datasets which are supposed to correspond to images, each input d-vector is normalized with respect to the values in the given input vector (global contrast normalization). For each sample vector $x^{(j)}$ subtract its mean $\mu^{(j)} = (1/d)\sum_{k=1}^{d} x_k^{(j)}$ and divide by its standard deviation $\sigma^{(j)}$ (also across the elements of the vector). In the case of images, this would discard the average illumination and contrast (scale).

Whitened PCA The Karhulen-Loève transform constantly improved the quality of the representation for each dataset. Assume the training set $\mathcal{D}$ is stored as a matrix $X \in \mathcal{M}_{\mathbb{R}}(n, d)$. First, we compute the empirical mean $\mu = (1/n)\sum_{i=1}^{n} X_{i.}$ where $X_{i.}$ denotes row i of the matrix X, i.e., example i. We center the data $\tilde{X} = X - \mu$ and compute the covariance matrix $C = (1/n)\tilde{X}^T\tilde{X}$. Then, we obtain the eigen-decomposition of the covariance matrix $C = V^{-1}UV$ i.e $U \in \mathbb{R}^d$ contains the eigen-values and $V \in \mathcal{M}_{\mathbb{R}}(d, d)$ the corresponding eigen-vectors (each row corresponds to an eigen-vector). We build a diagonal matrix U' where $U'_{ii} = \sqrt{C_{ii}}$. By the end, the output of the whitened PCA is given by $Y = (X - \mu)VU'$. In our experiments, we used the PCA implementation of the scikits[2] toolbox.

Feature selection In the datasets where the input is sparse, a preprocessing that we found very useful is the following: *only the features active on the training (development)* **and** *test (resp. validation) datasets are retained* for the test set (resp. validation) representations. We removed those whose frequency was low on both datasets (this introduces a new hyper-parameter that is the cut-off threshold, but we only tried a couple of values).

2. http://scikits.appspot.com/

4.2.2 Feature extraction

Feature extraction is the core of our pipeline and has been crucial for getting the first ranks during the challenge. Here we briefly introduce each method that has been used during the competition. See also (Bengio, 2011) along with the citations below for more details.

μ-ss-RBM

The μ-spike and slab Restricted Boltzmann Machine (μ-ssRBM) (Courville et al., 2011) is a recently introduced undirected graphical model that has demonstrated some promise as a model of natural images. The model is characterized by having both a real-valued *slab* vector and a binary *spike* variable associated with each hidden unit. The model possesses some practical properties such as being amenable to block Gibbs sampling as well as being capable of generating similar latent representations of the data to the mean and covariance Restricted Boltzmann Machine (Ranzato and Hinton, 2010).

The μ-ssRBM describes the interaction between three random vectors: the visible vector v representing the observed data, the binary "spike" variables h and the real-valued "slab" variables s. Suppose there are N hidden units and a visible vector of dimension D: $v \in \mathbb{R}^D$. The ith hidden unit ($1 \leq i \leq N$) is associated with a binary *spike* variable: $h_i \in \{0, 1\}$ and a real valued vector $s_i \in \mathbb{R}^K$, pooling over K linear filters. This kind of pooling structure allows the model to *learn* over which filters the model will pool – a useful property in the context of the UTL challenge where we cannot assume a standard "pixel structure" in the input. The μ-ssRBM model is defined via the energy function

$$\begin{aligned} E(v, s, h) = &- \sum_{i=1}^{N} v^T W_i s_i h_i + \frac{1}{2} v^T \left(\Lambda + \sum_{i=1}^{N} \Phi_i h_i \right) v \\ &+ \sum_{i=1}^{N} \frac{1}{2} s_i^T \alpha_i s_i - \sum_{i=1}^{N} \mu_i^T \alpha_i s_i h_i - \sum_{i=1}^{N} b_i h_i + \sum_{i=1}^{N} \mu_i^T \alpha_i \mu_i h_i, \end{aligned}$$

in which W_i refers to the ith weight matrix of size $D \times K$, the b_i are the biases associated with each of the spike variables h_i, and α_i and Λ are diagonal matrices that penalize large values of $\|s_i\|_2^2$ and $\|v\|_2^2$ respectively.

Efficient learning and inference in the μ-ssRBM is rooted in the ability to iteratively sample from the factorial conditionals $P(h \mid v)$, $p(s \mid v, h)$ and $p(v \mid s, h)$ with a Gibbs sampling procedure. For a detailed derivation of these conditionals, we refer the reader to (Courville et al., 2011). In training the μ-ssRBM, we use stochastic maximum likelihood (Tieleman, 2008) to update the model parameters.

Denoising Autoencoder

Traditional autoencoders map an input $x \in \mathbb{R}^{d_x}$ to a hidden representation h (the learnt features) with an affine mapping followed by a non-linearity s (typically a sigmoid): $h = f(x) = s(Wx + b)$. The representation is then mapped back to input space, initially producing a linear reconstruction $r(x) = W'f(x) + b_r$, where W' can be the transpose of W (tied weights) or a different matrix (untied weights). The autoencoder's parameters $\theta = W, b, b_r$ are optimized so that the reconstruction is close to the original input x in the sense of a given loss function $L(r(x), x)$ (the reconstruction error). Common loss functions include squared error $\|r(x) - x\|^2$, squared error after sigmoid $\|s(r(x)) - x\|^2$, and sigmoid cross-entropy $-\sum_i x_i \log s(r_i(x)) + (1 - x_i) \log(1 - s(r_i(x)))$. To encourage robustness of the representation, and avoid trivial useless solutions, a simple and efficient variant was proposed in the form of the Denoising Autoencoders (Vincent et al., 2008, 2010). Instead of being trained to merely reconstruct its inputs, a Denoising Autoencoder is trained to *denoise* artificially corrupted training samples, a much more difficult task, which was shown to force it to extract more useful and meaningful features and capture the structure of the input distribution (Vincent et al., 2010). In practice, instead of presenting the encoder with a clean training sample x, it is given as input a stochastically corrupted version $\tilde{x}$. The objective remains to minimize reconstruction error $L(r(\tilde{x}), x)$ with respect to clean sample x, so that the hidden representation has to help denoise. Common choices for the corruption include additive Gaussian noise, and masking a fraction of the input components at random by setting them to 0 (masking noise).

Contractive Autoencoder

To encourage robustness of the representation $f(x)$ obtained for a training input x, (Rifai et al., 2011) propose to penalize its *sensitivity* to that input, measured as the Frobenius norm of the Jacobian $J_f(x)$ of the non-linear mapping. Formally, if input $x \in \mathbb{R}^{d_x}$ is mapped by an encoding function f to a hidden representation $h \in \mathbb{R}^{d_h}$, this sensitivity penalization term is the sum of squares of all partial derivatives of the extracted features with respect to input dimensions:

$$\|J_f(x)\|_F^2 = \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2 .$$

Penalizing $\|J_f\|_F^2$ encourages the mapping to the feature space to be contractive in the neighborhood of the training data. The *flatness* induced by having small first derivatives will imply an *invariance* or *robustness* of the representation for small variations of the input.

While such a Jacobian term alone would encourage mapping to a useless constant representation, it is counterbalanced in auto-encoder training by the need for the learnt representation to allow a good reconstruction of the training examples.

Rectifiers

Recent works investigated linear rectified activation function variants. (Nair and Hinton, 2010) used Noisy Rectified Linear Units (NReLU) (i.e. $\max(0, x + N(0, \sigma(x)))$ for Restricted Boltzmann Machines. Compared to binary units, they observed significant improvements in term of generalization performance for image classification tasks. Following this line of work, (Glorot et al., 2011a) used the rectifier activation function (i.e. $\max(0, x)$) for deep neural networks and Stacked Denoising Auto-Encoders (SDAE) (Vincent et al., 2008, 2010) and obtained similarly good results.

This non-linearity has various mathematical advantages. First, it naturally creates sparse representations with true zeros which are computationally appealing. In addition, the linearity on the active side of the activation function allows gradient to flow well on the active set of neurons, possibly reducing the vanishing gradients problem.

In a semi-supervised setting similar to that of the Unsupervised and Transfer learning Challenge setup, (Glorot et al., 2011a) obtained state-of-the-art results for a sentiment analysis task (the Amazon 4-task benchmark) for which the bag-of-words input were highly sparse.

But learning such embeddings for huge sparse vectors with the proposed approach is still very expensive. Even though the training cost only scales linearly with the dimension of the input, it can become too expensive when the input becomes very large. Projecting the input vector to its embedding can be made quite efficient by using a sparse matrix-vector product. However, projecting the embedding back to the input space is quite expensive during decoding because one has to compute a reconstruction (and reconstruction error) for all inputs and not just the non-zeros. If the input dimension is 50,000 and the embedding dimension 1,000 then decoding requires 50,000,000 operations. In order to speed-up training for huge sparse input distributions, we use reconstruction sampling (Dauphin et al., 2011). The idea is to reconstruct all the non-zero elements of the input and a small random subset of the zero elements, and to use importance sampling weights to exactly correct the bias thus introduced.

The learning objective is sampled in the following manner:

$$\hat{L}(\mathbf{x}, \mathbf{z}) = \sum_{k}^{d} \frac{\hat{\mathbf{p}}_k}{\mathbf{q}_k} H(\mathbf{x}_k, \mathbf{z}_k)$$

where $\hat{\mathbf{p}} \in \{0, 1\}^{d_x}$ with $\hat{\mathbf{p}} \sim P(\hat{\mathbf{p}}|\mathbf{x})$. The sampling pattern $\hat{\mathbf{p}}$ is resampled for each presentation of the input and it controls which input unit will participate in the training cost for this presentation. The bias introduced by sampling can be corrected by setting the reweighting term $1/\mathbf{q}$ such that $\mathbf{q}_k = E[\hat{\mathbf{p}}_k|k, \mathbf{x}, \tilde{\mathbf{x}}]$.

The optimal sampling probabilities $P(\hat{\mathbf{p}}|\mathbf{x})$ are those that minimize the variance of the estimator $\hat{L}$. (Dauphin et al., 2011) show that reconstructing all non-zeros and a small subset of zeros is a good heuristic. The intuition is that the model is more likely to be wrong on the non-zeros than the zeros. Let $\mathcal{C}(\mathbf{x}, \tilde{\mathbf{x}}) = \{k : \mathbf{x}_k = 1 \text{ or } \tilde{\mathbf{x}}_k = 1\}$. Then bit k is reconstructed with probability

$$P(\hat{\mathbf{p}}_k = 1|\mathbf{x}_k) = \begin{cases} 1 & \text{if } k \in \mathcal{C}(\mathbf{x}, \tilde{\mathbf{x}}) \\ |\mathcal{C}(\mathbf{x}, \tilde{\mathbf{x}})|/d_x & \textit{otherwise} \end{cases} \tag{4.1}$$

(Dauphin et al., 2011) show that the computational speed-up is on the order of ${}^{d_{SMP}}/_{d_x}$ where d_{SMP} is the average number of units that are reconstructed and d_x is the input dimension. Furthermore, reconstruction sampling yields models that converge as fast as the non-sampled networks in terms of gradient steps (but where each step is much faster).

4.2.3 Postprocessing

The competition servers use a Hebbian classifier. Specifically, the discriminant function applied to a test set matrix Z (one row per example) after training the representation on a training set matrix X (one row per example) is given by

$$f(Z) = ZX^T y$$

where $y_i = 1/n_p$ if training example i is positive, or $-1/n_n$ if training example i is negative, where n_p and n_n are the number of positive and negative training examples, respectively. One classifier per class (one against all) is trained.

This classifier does not have any regularization hyperparameters. We were interested in discovering whether some postprocessing of our features could result in Hebbian learning behaving as if it was regularized. It turns out that in a sense, Hebbian learning is already maximally regularized. Fisher discriminant analysis can be solved as a linear regression problem (Bishop, 2006), and the L2 regularized version of this problem yields this discriminant function:

$$g_\lambda(Z) = Z(X^T X + \lambda I)X^T y$$

where λ is the regularization coefficient. Note that

$$\lim_{\lambda \to \infty} \frac{g_\lambda(Z)}{||g_\lambda(Z)||} = \frac{f(Z)}{||f(Z)||}.$$

Since scaling does not affect the final classification decision, Hebbian learning may be seen as maximally regularized Fisher discriminant analysis. It is possible to reduce Hebbian learning's implicit L2 regularization coefficient to some smaller λ by multiplying Z by $(X^T X + \lambda I)^{-1/2})$, but it is not possible to increase it.

Despite this implicit regularization, overfitting is still an important obstacle to good performance in this competition due to the small number of training examples used. We therefore explored other means of avoiding overfitting, such as reducing the number of features and exploring sparse codes that would result in most of the features appearing in the training set being 0. However, the best results and regularization were obtained by a transductive PCA.

Transductive PCA

A Transductive PCA is a PCA transform trained not on the training set but on the test (or validation) set. After training the first k layers of the pipeline on the training set, we trained a PCA on top of layer k, either on the validation set or on the test set (depending on whether we were submitting to the validation set or the test set). Regarding the notation used in 4.2.1, we apply the same transformation with X replaced by the representation on top of layer k of the validation set or the test set i.e $h(X_{valid})$.

This transductive PCA thus only retains variations that are the dominant ones in the test or validation set. It makes sure that the final classifier will ignore the variations present in the training set but irrelevant for the test (or validation) set. In a sense, this is a generalization of the strategy introduced in 4.2.1 of removing features that were not present in the both training and test / validation sets. The lower layers only keep the directions of variation that are dominant in the training set, while the top transductive PCA only keeps those that are significantly present in the validation (or test) set.

We assumed that the validation and test sets contained the same number of classes to validate the number of components on the validation set performance. In general, one needs at least $k-1$ components in order to separate k classes by a set of one-against-all classifiers. Transductive PCA has been decisive for winning the competition as it improved considerably the performance on all the datasets. In some cases, we also used a mixed strategy for the intermediate layers, mixing examples from all three sets.

Other methods

After the feature extraction process, we were able to visualize the data as a three-dimensional scatter plot of the representation learnt. On some datasets, a very clear clustering pattern became visually apparent, though it appeared that several clouds came together in an ambiguous region of the latent space discovered.

In order to attempt to disambiguate this ambiguous region without making hard-threshold decisions, we fit a Gaussian mixture model with the EM algorithm and a small number of Gaussian components chosen by visual inspection of these clouds. We then used the posterior probabilities of the cluster assignments as an alternate encoding.

K-means, by contrast with Gaussian mixture models, makes a hard decision as to cluster assignments. Many researchers were recently impressed when they found out that a certain kind of feature representation (the “triangle code”) based on K-means, combined with specialized pre-processing, yielded state of the art performance on the CIFAR-10 image classification benchmark (Coates et al., 2011). Therefore, we tried K-means with a large number of means and the triangle code as a post-processing step.

In the end, though, none of our selected final entries included a Gaussian mixture model or K-means, as the transductive PCA always worked better as a post-processing layer.

4.3 Criterion

The usual kind of overfitting is due to specializing to particular labeled examples. In this transfer learning context, another kind of overfitting arose: overfitting a representation to particular *classes*. Since the validation set and test set have non-intersecting sets of classes, finding representations that work well on the validation set was not a guarantee for good behavior on the test set, as we learned from our experience with the competition first phase. Note also that the competition was focused on a particular criterion, the Area under the Learning Curve (ALC)[3] which gives much weight to the cases with very few labeled examples (1, 2, or 4, per class, in particular, get almost half of the weight). So the question we investigated in the second and final phase (where some training set labels were revealed) was the following: does the ALC of a representation computed on a particular subset of classes correlate with the ALC of the same representation computed on a different set of classes ?

Overfitting on a specific subset of classes can be observed by training a PCA separately on the training, validation and test sets on **ULE** (this data set corresponds to MNIST digits). The number of components maximizing the ALC will be different, depending on the choice of the subset of classes. Figure 4.1a illustrates

3. http://www.causality.inf.ethz.ch/ul_data/DatasetsUTLChallenge.pdf

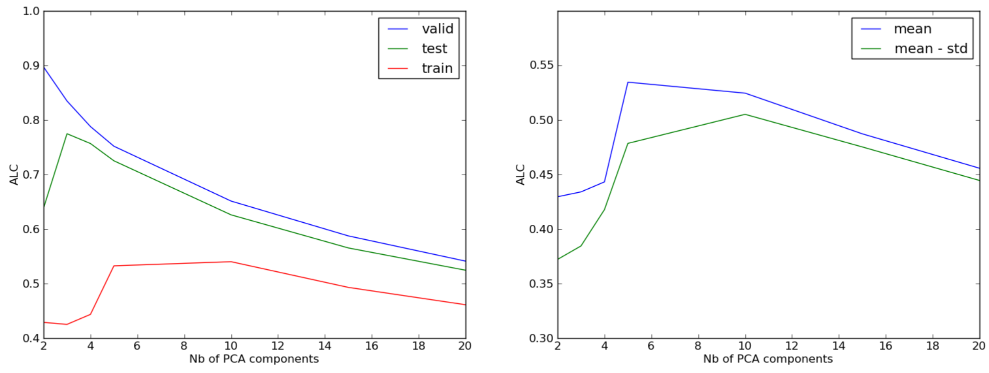
the effect of the number of components retained on the training, validation and test ALCs. While the best number of components on the validation set would be 2, choosing this number of components for the test set significantly degrades the test ALC.

During the first phase, we noticed the absence of correlation between the validation ALC and test ALC computed on the **ULE** dataset. During the second phase, we tried to reproduce the competition setting using the labels available for transfer with the hope of finding a criteria that would guarantee generalization. The ALC was computed on every subset of at least two classes found in the transfer labels and metrics were derived. Those metrics are illustrated in Figure 4.1b. We observed that the standard deviation seems to be inversely proportional to the generalization accuracy, therefore substracting it from the mean ALC ensures that the choice of hyper-parameters is done in a range where the training, validation and test ALCs are correlated. In the case of a PCA, optimizing the $\mu - \sigma$ criteria correctly returns the best number of PCA components, ten, where the training, validation and test ALCs are all correlated.

It appears that this criterion is a simple way to use the small amount of labels given to the competitors for the phase 2. However, this criterion has not been heavily tested during the competition since we always selected our best models with respect to the validation ALC returned by the competition servers. From the phase 1 to the phase 2, we only explored the space of the hyperparameters of our models using a finer grid.

4.4 Results

For each of the five datasets, **AVICENNA**, **HARRY**, **TERRY**, **SYLVESTER** and **RITA**, the strategy retained for the final winning submission on the phase 2 is precisely described. Training such a deep stack of layers from preprocessing to postprocessing takes at most 12 hours for each dataset once you have found the good hyperparameters. All our models are implemented in Theano (Bergstra et al., 2010a), a Python library that allows transparent use of GPUs. During the competition, we used a cluster of GPUs, Nvidia GeForce GTX 580.



(a) ALC on the three sets **(b)** Criterion

Figure 4.1 – ULE Dataset Left: ALC on training, validation and test sets of PCA representations, with respect to the number of principal components retained. Right: Comparison between training, validation and test ALC and the criterion computed from the ALC, obtained on every subset of at least 2 classes present in the transfer labels for different numbers of components of a PCA.

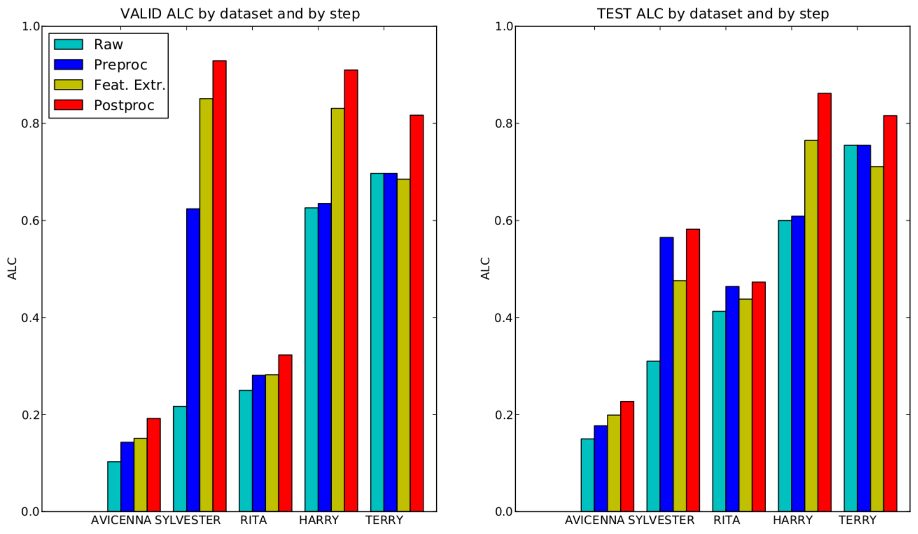


Figure 4.2 – For each data set, we report the Validation and Test ALC after each layer (from raw data to postprocessing). It allows us to see where overfitting arose (**SYLVESTER**) and which of the layers resulted the more important to improve the overall performance.

4.4.1 AVICENNA

Nature of the data It corresponds to arabic manuscripts and consists of $150,205$ training samples of dimension 120.

Best Architecture For preprocessing, we fitted a whitened-PCA on the raw data and kept the first 75 components in order to eliminate the noise from the input distribution. Then, the second layer consisted in a Denoising Autoencoder of 600 hidden units trained with a binomial noise, i.e, each component of the input had a probability $p = 0.3$ of being masked (set to 0). The top layer was a transductive PCA with only the 7 principal components.

Results This strategy ranked first with a validation and final ALC score of 0.1932 and 0.2273 respectively. Training a contractive auto-encoder gives similar results on the validation set i.e a validation and final ALC score of 0.1930 and 0.1973 respectively.

4.4.2 HARRY

Nature of the data It corresponds to human actions and consists of $69,652$ training samples of dimension $5,000$, which are sparse: only 2% of the components are typically non-zero.

Best Architecture For the first layer, we uniformized the non-zero feature values (aggregating all features) across the concatenation of the training, validation and test sets. For the second layer, we trained on the union of the 3 sets a Denoising Auto-Encoder with rectifier units and reconstruction sampling. We used the binomial masking noise ($p = 0.5$) as corruption process, the logistic sigmoid as reconstruction activation function and the cross entropy as reconstruction error criterion. The size of the hidden layer was 5000 and we added an L_1 penalty on the activation values to encourage sparsity of the representation. For the third layer, we applied a transductive PCA and kept 3 components.

Results We obtained the best validation ALC score of the competition. This was also the case for the final evaluation score with an ALC score of 0.861933, whereas

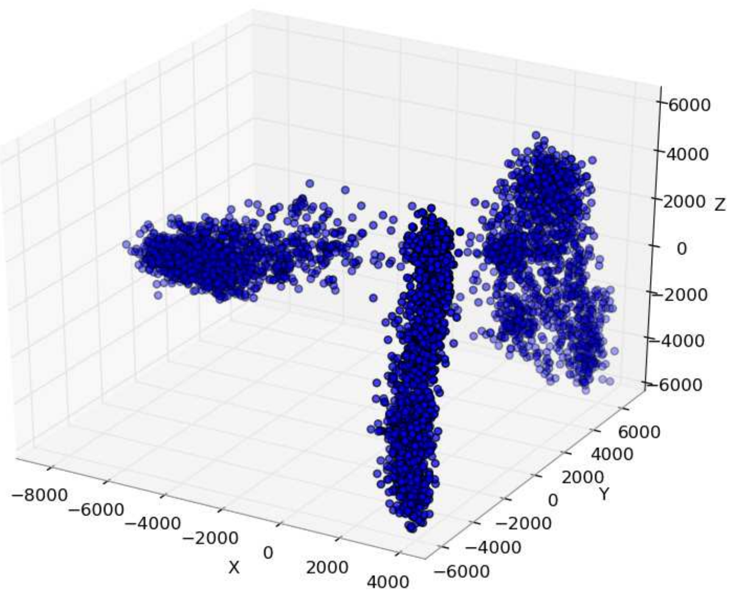


Figure 4.3 – **HARRY** evaluation set after the transductive PCA, the data is nicely clustered, suggesting that the learned preprocessing has discovered the underlying class structure.

the second best obtained 0.754497. Figure 4.3 shows the final data representation we obtained for the test (evaluation) set.

4.4.3 TERRY

Nature of the data This is a natural language processing (NLP) task, with 217,034 training samples of dimension 41,236, and a high level of sparsity: only 1% of the components are non-zero in average.

Best Architecture A setup similar to **HARRY** has been used for **TERRY**. For the first layer, we kept only the features that were active on both training and validation sets (and similarly with the test set, for preparing the test set representations). Then, we divided the non-zero feature values by their standard deviation across the concatenation of the training, validation and test set. For the second layer, we trained on the three sets a Denoising Auto-Encoder with rectifier units and reconstruction sampling. We used binomial masking noise ($p = 0.5$) as corruption process, the logistic sigmoid as reconstruction activation function and the squared error as reconstruction error criterion. The size of the hidden layer was

5000 and we added an L_1 penalty on the activation values to encourage sparsity of the representation. For the third layer, we applied a transductive PCA and kept the leading 4 components.

Results We ranked second on this dataset with a validation and final score of 0.816752 and 0.816009.

4.4.4 SYLVESTER

Nature of the data It corresponds to ecology data and consists of $572,820$ training samples of dimension 100.

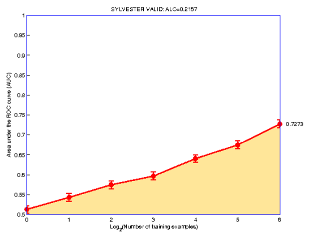
Best Architecture For the first layer, we extracted the meaningful features and discarded the apparent noise dimensions using PCA. We used the first 8 principal dimensions as the feature representation produced by the layer because it gave the best performance on the validation set. We also whitened this new feature representation by dividing each dimension by its corresponding singular value (square root of the eigenvalue of the covariance matrix, or corresponding standard deviation of the component). Whitening gives each dimension equal importance both for the classifier and subsequent feature extractors. For the second and third layers, we used a Contractive Auto-Encoder (CAE). We have selected a layer size of 6 based on validation ALC. For the fourth layer, we again apply a transductive PCA.

Figure 4.4 shows the evolution of the ALC curve for each layer of the hierarchy. Note that at each layer, we only use the top-level features as the representation.

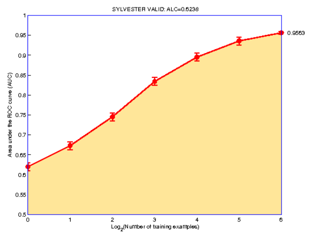
Results This yielded an ALC of 0.85109 for the validation set and 0.476341 for the test set. The difference in ALC may be explained by the fact that Sylvester is the only dataset where the test set contains more classes than the validation set and, and thus our assumpptions of equal number of classes might have hurt test performance here.

4.4.5 RITA

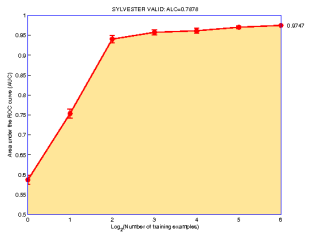
Nature of the data It corresponds to the CIFAR RGB image dataset and consists of $111,808$ training samples of dimension $7,200$.



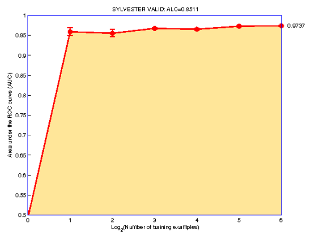
(a) Raw Data



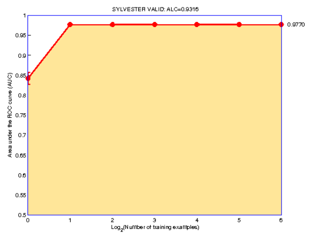
(b) 1st layer



(c) 2nd layer



(d) 3rd layer



(e) t-PCA

Figure 4.4 – Validation performance increases with the depth of our feature hierarchy for the **SYLVESTER** dataset. ALC: Raw Data (0.2167), 1st Layer (0.6238), 2nd Layer (0.7878), 3rd Layer (0.8511), t-PCA(0.9316)

Best Architecture The μ-ssRBM was initially developed as a model for natural images. As such, it was a natural fit for the **RITA** dataset. Their ability to learn the pooling structure was also a clear advantage, since the max-pooling strategy typically used in vision tasks with convolutional networks (LeCun et al., 1998a) could no longer be employed due to the obfuscated nature of the dataset.

For pre-processing, each image has been contrast-normalized. Then, we reduced the dimensionality of the training dataset by learning on the first $1,000$ principal components. For feature extraction, we chose the number of hidden units to be large enough (1000) while still being computationally efficient on GPUs. The learning rate of 10^{-3} and number of training updates ($110,000$ updates with minibatches of size 16) are chosen such that hidden units have sparse activations through pools of size 9, hovering around 10-25%. The post-processing was consistent with the other datasets: we used the transductive PCA method using only the first 4 principal components.

Results This yielded an ALC score of 0.286 and 0.437 for the validation and final test sets respectively. We also tried to stack 3 layers of contractive auto-encoders directly on the raw data and it achieved a valid ALC of 0.3268. As it appeared actually transductive, we prefered to keep the μ-ssRBM in our competition entries because it was trained on the whole training set.

4.5 Conclusion

The competition setting with different class labels in the validation and the test sets was quite unusual. The similarity between two classes must be sufficient for transfer learning to be possible. More formal assessments of class similarity might be useful in such settings. It is not obvious that the similarity between the subsets of classes chosen for the different datasets in the context of the competition is sufficient for an effective generalization, neither that the similarity between the subsets of classes found in the transfer labels is representative of the similarity between the classes found in the training, validation and test datasets. Finally, for assessing transfer across classes properly would require a larger number of classes. In a future competition, we suggest that both similarity and representativeness (including number of classes) should be ensured in a more formal or empirical way.

On all five tasks, we have found the idea of stacking different layer-wise representation-learning algorithms to work very well. One surprise was the effectiveness of PCA both as a first layer and a last layer, in a transductive setting. As core feature-learning blocks, the contractive auto-encoder, the denoising auto-encoder and spike-and-slab RBM worked best for us on the dense datasets, while the sparse rectifier denoising auto-encoder worked best on the sparse datasets.

Acknowledgements

The authors acknowledge the support of the following agencies for research funding and computing support: NSERC, RQCHP, CIFAR. This is also funded in part by the French ANR Project ASAP ANR-09-EMER-001.

5 Préambule au Deuxième Article

Ce deuxième article considère différentes techniques d'apprentissage non supervisé appliquées de manière séquentielle. Nous étudions plusieurs stratégies pour faire face à un problème d'entrée à dimensionnalité élevée. Une différence majeure avec le premier article réside dans l'utilisation de la structure des données d'entrée contrairement au cas précédent où aucune information a priori ne pouvait être utilisée lors du processus d'apprentissage. Chaque dimension de l'espace d'entrée correspondant à un détecteur d'objet particulier, pour une pose, une position et une échelle données. Notre approche obtient une amélioration des performances par rapport au système original et montre une capacité à apprendre une sémantique dans l'apparition jointe d'objets intéressante pour la classification de scènes dans des images.

5.1 Détails de l'article

Unsupervised Learning of Semantics of Object Detections for Scene Categorization Grégoire Mesnil, Salah Rifai, Antoine Bordes, Xavier Glorot, Yoshua Bengio and Pascal Vincent. *Advances in Intelligent Systems, Computing* Springer, Vol. 318, Maria De Marsico and Ana Fred (Eds): Pattern Recognition Applications and Methods, 2015

Contribution Personnelle Le début de ce travail a été effectué en collaboration avec Antoine Bordes, Xavier Glorot et Salah Rifai. L'idée initiale était d'utiliser le contexte dans des images (présence d'objets) pour améliorer la catégorisation de scènes. Je me suis chargé de préparer les représentations d'Object Bank pour tous nos ensembles de données et j'ai effectué l'intégralité des expériences excepté celles relatives aux CAEs réalisées par Salah Rifai. J'ai ensuite adapté le pipeline de la compétition afin de réduire la dimension d'entrée et permettre un apprentissage non supervisé. Après plusieurs discussions fructueuses lors d'une présentation à un workshop de NIPS en 2011 (Mesnil et al., 2011), l'idée s'est améliorée et notre publication (Mesnil et al., 2013) a été acceptée à ICPRAM. Après l'oral, l'article a été selectionné pour publication dans un recueil contenant les meilleures publications présentées à cette conférence (Mesnil et al., 2015). J'ai rédigé l'article en collaboration avec les coauteurs.

5.2 Contexte

À cette époque les réseaux de neurones convolutionnels n'ont pas encore remporté la compétition ImageNet (Krizhevsky et al., 2012a) et les détecteurs d'objet à l'état de l'art restent ceux utilisés dans Object Bank (Felzenszwalb et al., 2010). Pour la classification de scènes, Object Bank est encore la méthode à l'état de l'art. C'est pourquoi nous avons décidé d'utiliser les représentations d'Object Bank comme support à notre travail afin d'explorer si une combinaison des détections d'objets de manière plus abstraite permettait un gain en performance.

5.3 Contributions

Malgre ses très bonnes performances à l'époque, un des inconvénients d'Object Bank reste la dimensionnalité très élevée de la représentation. Dans notre travail, nous avons exploré différentes techniques pour combiner les détections d'objets afin de permettre aux chercheurs intéressés d'utiliser une représentation d'Object Bank plus compacte et plus performante.

5.4 Récents développements

Il est certain qu'utiliser des détecteurs d'objets à base de réseaux de neurones convolutionnels améliorerait nettement les performances de notre approche (Zhou et al., 2014). Au moment du développement de ces idées, notre méthode souffrait principalement du manque de précision des détecteurs d'objets.

6 Unsupervised Learning of Semantics of Object Detections for Scene Categorization

6.1 Introduction

Automatic scene categorization is crucial for many applications such as content-based image indexing (Smeulders et al., 2000) or image understanding. This is defined as the task of assigning images to predefined categories ("office", "sailing", "mountain", etc.). Classifying scene is complicated because of the large variability of quality, subject and conditions of natural images which lead to many ambiguities w.r.t. the corresponding scene label.

Standard methods build an intermediate representation before classifying scenes by considering the image as a whole (Torralba, 2003; Vogel and Schiele, 2004; Fei-Fei and Perona, 2005; Oliva and Torralba, 2006). In particular, many such approaches rely on power spectral information, such as magnitude of spatial frequencies (Oliva and Torralba, 2006) or local texture descriptors (Fei-Fei and Perona, 2005). They have shown to perform well in cases where there are large numbers of scene categories.

Another line of work conveys promising potential in scene categorization. First applied to object recognition (Farhadi et al., 2009a), attribute-based methods have now proved to be effective for dealing with complex scenes. These models define high-level representations by combining semantic lower-level elements, e.g., detection of object parts. A precursor of this tendency for scenes was an adaptation of pLSA (Hofmann, 2001) to deal with "visual words" proposed by (Bosch et al., 2006). An extension of this idea consists in modeling an image based on its content i.e. its objects (Espinace et al., 2010; Li-Jia Li and Fei-Fei, 2010a). Hence, the Object-Bank (OB) project (Li-Jia Li and Fei-Fei, 2010b) aims at building high-dimensional over-complete representations of scenes (of dimension $44,604$) by combining the outputs of many object detectors (177) taken at various poses, scales and positions in the original image (leading to 252 attributes per detector). Experimental results indicate that this approach is effective since simple classifiers such as Support Vector Machines trained on their representations achieve state-of-the-art performance. However, this approach suffers from two flaws: (1) curse of dimensionality (very large number of features) and (2) individual object detectors have a poor precision (30% at most). To solve (1), the original paper proposes to use structured norms and group sparsity to make best use of the large input. Our work studies new ways to combine the very rich information provided by these multiple detectors, dealing with the uncertainty of the detections. A method designed to select and combine the most informative attributes would be able to carefully manage redundancy, noise and structure in the data, leading to better scene categorization performance.

Hence, in the following, we propose a sequential 2-steps strategy for combining the feature representations provided by the OB object detectors on which the linear SVM classifier is destined to be trained for categorizing scenes. The first step adapts Principal Components Analysis (PCA) to this particular setting: we show that it is crucial to take into account the structure of the data in order for PCA to perform well. The second one is based on *Deep Learning*. Deep Learning has emerged recently (see (Bengio, 2009a) for a review) and is based on neural network algorithms able to discover data representations in an unsupervised fashion (Hinton et al., 2006b; Bengio et al., 2007b; Ranzato et al., 2007; Kavukcuoglu et al., 2009; Jarrett et al., 2009). We propose to use this ability to combine multiple detector features. Hence, we present a model trained using Contractive Auto-Encoders (Rifai et al., 2011), which have already proved to be efficient on many image tasks and has contributed to winning a transfer learning challenge (Mesnil et al., 2012).

We validate the quality of our models in an extensive set of experiments in which several setups of the sequential feature extraction process are evaluated on benchmarks for scene classification (Lazebnik et al., 2006; Li and Fei-Fei, 2007; Quattoni and Torralba, 2009; Xiao et al., 2010). We show that our best results substantially outperform the original methods developed on top of OB features, while producing representations of much lower dimension. The performance gap is usually large, indicating that advanced combinations are highly beneficial. We show that our method based on dimensionality reduction followed by deep learning offers a flexibility which makes it able to benefit from semi-supervised and transfer learning.

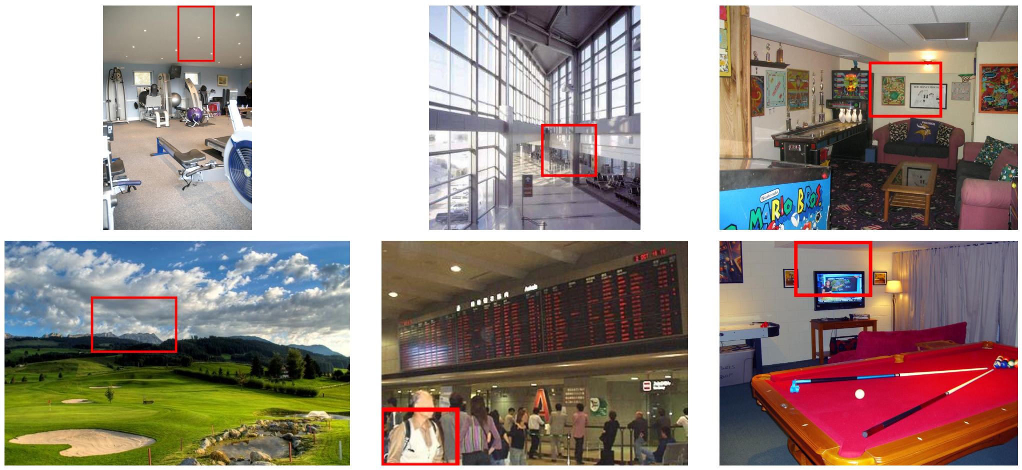


Figure 6.1 – *Left:* Cloud *Middle:* Man *Right:* Television. *Top:* False Detections *Bottom:* True Detections. Images from **SUN** (Xiao et al., 2010) for which we compute the OB representation and display the bounding box around the average position of various objects detectors. For instance, the *television* detector can be viewed either as a *television* detector or a *rectangle* shape detector i.e. high-order statistical properties of the image.

6.2 Scene Categorization with Object-Bank

Let us begin by introducing the approach of the OB project (Li-Jia Li and Fei-Fei, 2010a). First, the 177 most useful (or frequent) objects were selected from popular image datasets such as LabelMe (Russell et al., 2008), ImageNet (Deng et al., 2009a) and Flickr. For each of these 177 objects, a specific detector, existing in the literature (Felzenszwalb et al., 2008; Hoiem et al., 2005), was trained. Every detector is composed of 2 *root filters* depending on the pose, each one coming with its own deformable pattern of parts, e.g., there is one root filter for the front-view of a bike and one for the side-view. These $354 = 177 \times 2$ part-based filters (each composed by a root and its parts) are used to produce features of natural images. For a given image, a filter is convolved at 6 different scales. At each scale, the max-response among $21 = 1+4+16$ positions (whole image, quadrants, quadrants within each quadrant) is kept, producing a response map of dimension $126 = 6 \times 21$. All 2×177 maps are finally concatenated to produce an over-complete representation $x \in \mathbb{R}^{44,604}$ of the original image.

In the original OB paper (Li-Jia Li and Fei-Fei, 2010a), classifiers for scene categorization are learned directly on these feature vectors of dimension $44,604$.

More precisely, C classifiers (Linear SVM or Logistic Regression) are trained in a 1-versus-all setting in order to predict the correct scene category $y_{\text{category}}(x)$ among C different categories. Various strategies using structured sparsity with combinations of ℓ_1/ℓ_2 norms have been proposed to handle the very large input.

6.3 Unsupervised Feature Learning

The approach of OB for the task of scene categorization, based on specific object detectors, is appealing since it works well in practice. This suggests that a scene is better recognized by first identifying basic objects and then exploiting the underlying semantics in the dependencies between the corresponding detectors.

However, it appears that none of the individual object detectors reaches a recognition precision of more than 30%. Hence, one may question whether the ideal view that inspired this approach (and expressed above) is indeed the reason of OB's success. Alternatively, one may hypothesize that the $44,604$ OB features are more useful for scene categorization because they represent high level statistical properties of images than because they precisely report the absence/presence of objects – see Figure 6.1. OB tried structured sparsity to handle this feature selection but there may be other ways – simpler or not.

This paper investigates several ways of learning higher-level features **on top of** the high dimensional representation provided by OB, expecting that capturing further structure may improve categorization performance. Our approach employs *unsupervised feature learning/extraction* algorithms, i.e. generic feature extraction methods which were not developed specifically for images. We will consider both standard Principal Component Analysis and Contractive Auto-Encoders (Rifai et al., 2011). The latter is a recent machine learning method which has proved to be a robust feature extraction tool.

6.3.1 Principal Component Analysis

Principal Component Analysis (PCA) (Pearson, 1901; Hotelling, 1933) is the most prevalent technique for linear dimensionality reduction. A PCA with k components finds the k orthonormal directions of projection in input space that retain most of the *variance* of the training data. These correspond to the eigenvectors associated with the leading eigenvalues of the training data's covariance matrix. Principal components are ordered, so that the first corresponds to the direction along which the data varies the most (largest eigenvalue), etc. . .

Since we will consider an auto-encoder variant (presented next), we should mention here a well-known result: a linear auto-encoder with k hidden units, trained to minimize squared reconstruction error, will learn projection directions that span the same *subspace* as a k component PCA (Baldi and Hornik, 1989). However the regularized non-linear auto-encoder variant that we consider below is capable of extracting qualitatively different, and usually more useful, nonlinear features.

6.3.2 Contractive Auto-Encoders

Contractive Auto-Encoders (CAEs) (Rifai et al., 2011) are among the latest developments in a line of machine learning research on nonlinear feature learning methods, that started with the success of Restricted Boltzmann Machines (Hinton et al., 2006b) for pre-training deep networks, and was followed by other variants of auto-encoders such as sparse (Ranzato et al., 2007; Kavukcuoglu et al., 2009; Goodfellow et al., 2009) and denoising auto-encoders (Vincent et al., 2008). It was selected here mainly due to its practical ease of use and recent empirical successes.

Unlike PCA that decomposes the input space into leading *global* directions of variations, the CAE learns features that capture local directions of variation (in some regions of input space). This is achieved by penalizing the norm of the Jacobian of a latent representation $h(x)$ with respect to its input x at training samples. Rifai et al. (Rifai et al., 2011) show that the resulting features provide a local coordinate system for a low dimensional manifold of the input space. This corresponds to an atlas of charts, each corresponding to a different region in input space, associated with a different set of active latent features. One can think about this as being similar to a mixture of PCAs, each computed on a different set of training samples that were grouped together using a similarity criterion (and corresponding to a different input region), but without using an independent parametrization for each component of the mixture, i.e., allowing to generalize across the charts, and away from the training examples.

In the following, we summarize the formulation of the CAE as a regularized extension of a basic Auto-Encoder (AE). In our experiments, the parametrization of this AE consists in a non-linear encoder or latent representation h of m hidden units with a linear decoder or reconstruction g towards an input space of dimension d.

Formally, the latent variables are parametrized by:

$$h(x) = s(Wx + b_h), \tag{6.1}$$

where s is the element-wise logistic sigmoid $s(z) = \frac{1}{1+e^{-z}}$, $W \in \mathcal{M}_{m\times d}(\mathbb{R})$ and $b_h \in \mathbb{R}^m$ are the parameters to be learned during training. Conversely, the units of the decoder are linear projections of $h(x)$ back into the input space:

$$g(h(x)) = W^T h(x). \tag{6.2}$$

Using mean squared error as the reconstruction objective and the L2-norm of the Jacobian of h with respect to x as regularization, training is carried out by minimizing the following criterion by stochastic gradient descent:

$$\mathcal{J}_{\text{CAE}}(\Theta) = \sum_{x\in\mathcal{D}} \|x - g(h(x))\|^2 + \lambda \sum_{i=1}^{m}\sum_{j=1}^{d} \left|\frac{\partial h_i}{\partial x_j}(x)\right|^2, \tag{6.3}$$

where $\Theta = \{W, b_h\}$, $\mathcal{D} = \{x^{(i)}\}_{i=1,\dots,n}$ corresponds to a set of n training samples $x \in \mathbb{R}^d$ and λ is a hyper-parameter controlling the level of contraction of h. A notable difference between CAEs and PCA is that features extracted by CAEs are non-linear w.r.t. the inputs, so that multiple layers of CAEs can be usefully composed (stacked), whereas stacking linear PCAs is pointless.

6.4 Extracting Better Features with Advanced Combination Strategies

In this work, we study two different sub-structures of OB. We consider the *pose* response defined by the output of only one part-based filter at all positions and

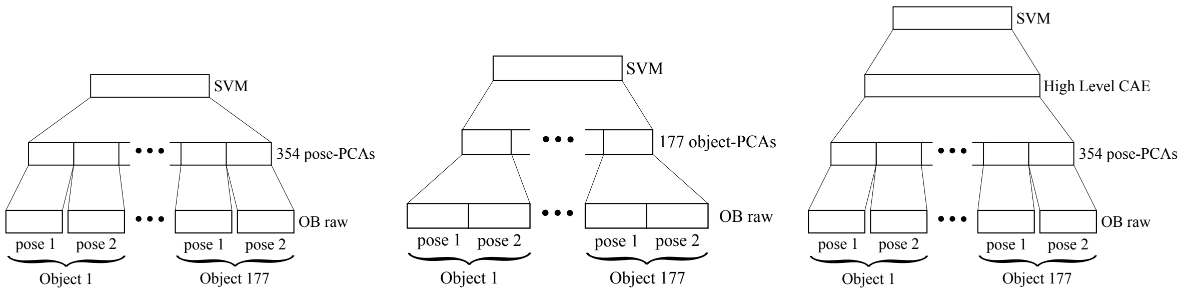


Figure 6.2 – Different Combination Strategies (a) and (b) *pose* and *object* PCAs (c) high-level CAE: *pose*-PCA as dimensionality reduction technique in the first layer and a CAE stacked on top. We denote it high-level because it can learn *context information* i.e. plausible joint appearance of different objects.

scales, and the *object* response which is the concatenation of all *pose* responses associated to an object. Combination strategies are depicted in Figure 6.2.

6.4.1 Simplistic Strategies: Mean and Max Pooling

The idea of pooling responses at different locations or poses has been successfully used in Convolutional Neural Networks such as LeNet-5 (LeCun et al., 1999) and other visual processing (Serre et al., 2005) architectures inspired by the visual cortex.

Here, we pool the 252 responses of each object detector into one component (using the mean or max operator) leading to a representation of size $177 = 44604/252$. It corresponds to the mean/max over the object responses at different scales and locations. One may view the object max responses as features encoding absence/presence of objects while discarding all the information about the detector's positions.

6.4.2 Combination Strategies with PCA

PCA is a standard method for extracting features from high dimensional input, so it is a good starting point. However, as we find in our experiments, exploiting the particular structure of the data, e.g., according to poses, scales, and locations, can yield to improved results.

Whole PCA. An ordinary PCA is trained on the raw output of OB ($x \in \mathbb{R}^{44{,}604}$) without looking for any structure. Given the high-dimensionality of OB's representation, we used the Randomized PCA algorithm of the scikits toolbox[1].

Pose-PCA. Each of the two *poses* associated with each *object* detector is considered independently. This results in $354 = 2 \times 177$ different PCAs, which are trained on *pose* outputs ($x \in \mathbb{R}^{126}$) – see Figure 6.2.

Object-PCA. Only each *object* response ($x \in \mathbb{R}^{252}$) is considered separately, therefore 177 PCAs are trained in total. It allows the model to capture variations among all *pose* responses at various scales and positions – see Figure 6.2.

Note that, in all cases, whitening the PCA (i.e. dividing each eigenvector's response by the corresponding squared root eigenvalue) performs very poorly. For post-processing, the PCA outputs $\tilde{x}$ are always normalized: $\tilde{x} \leftarrow (\tilde{x} - \mu)/\sigma$ according to mean μ and the deviation σ of the whole, per *object* or per *pose* PCA outputs. Thereby, we ensure contributions from all *objects* or *poses* to be in the same range. The number of components in all cases has been selected according to the classification accuracy estimated by 5-fold cross-validation.

6.4.3 Improving upon PCA with CAE

Due to hardware limitations and high-dimensional input, we could not train a CAE on the whole OB output ("whole CAE"). However, we address this problem with the sequential feature extraction steps below.

To overcome the tractability problem that forbids a CAE to be trained on the whole OB output, we preprocess it by using the *pose*-PCAs as a dimensionality reduction method. We keep only the 5 first components of each *pose*. Given this low-dimensional representation (of dimension $1,770$), we are able to train a CAE – see Figure 6.2. The CAE has a global view of all object detectors and can thus learn to capture *context information*, defined by the joint appearance of combinations of various objects. Moreover, instead of using an SVM on top of the learned representations, we can use a Multi-Layer Perceptron whose weights would be initialized

1. Available from *http://scikits.appspot.com/*

by those of this CAE. This setting is where the CAE has shown to perform best in practice (Rifai et al., 2011).

6.5 Experiments

6.5.1 Datasets

We evaluate our approach on 3 scene datasets, cluttered indoor images (MIT Indoor Scene), natural scenes (15-Scenes), and event/activity images (UIUC-Sports). Images from a large scale scene recognition dataset (SUN-397 database) have also been used for unsupervised learning.

— **MIT Indoor** is composed of 67 categories and, following (Li-Jia Li and Fei-Fei, 2010a; Quattoni and Torralba, 2009), we used 80 images from each category for training and 20 for testing.

— **15-Scenes** is a dataset of 15 natural scene classes. According to (Lazebnik et al., 2006), we used 100 images per class for training and the rest for testing.

— **UIUC-Sports** contains 8 event classes. We randomly chose 70 / 60 images for our training / test set respectively, following the setting of (Li-Jia Li and Fei-Fei, 2010a; Li and Fei-Fei, 2007).

— **SUN-397** contains a full variety of 397 well sampled scene categories (100 samples per class) composed of $108,754$ images in total.

6.5.2 Tasks

We consider 3 different tasks to evaluate and compare the considered combination strategies. In particular, various supervision settings for learning the CAE are explored. Indeed, a great advantage of this kind of method is that it can make use of vast quantities of unlabeled examples to improve its representations. We thus illustrate this by proposing experiments in which the CAE has been trained in supervised or in semi-supervised way and also in a transfer context.

	MIT *(plain)*	MIT+SUN *(semi-supervised)*
object-MAX + SVM	24.3%	–
object-MEAN + SVM	41.0%	–
whole-PCA + SVM	40.2%	–
object-PCA + SVM	42.6%	46.1%
pose-PCA + SVM	40.1%	46.0%
pose-PCA + MLP	42.9%	46.3%
pose-PCA + CAE (MLP)	**44.0**%	**49.1**%
Object Bank + SVM	37.6%	–
Object Bank + rbf-SVM	37.7%	–
DPM + Gist + SP	43.1%	–
Improvement w.r.t. Object Bank	+6.4%	+11.5%

Table 6.1 – **MIT Indoor.** Results are reported on the official split (Quattoni and Torralba, 2009) for all combination strategies described in Section 6.4. Only the unsupervised feature learning strategies (PCA and CAE based) *can* benefit from the addition of unlabeled scenes from SUN. Object Bank + SVM refers to the original system (Li-Jia Li and Fei-Fei, 2010a) and DPM + Gist + SP (Pandey and Lazebnik, 2011) corresponds to the state-of-the-art method on MIT Indoor.

MIT Indoor (plain). Only the official training set of the MIT Indoor scene dataset (5, 360 images) is used for unsupervised feature learning. Each representation is evaluated by training a linear SVM on top of the learned features.

MIT+SUN (semi-supervised). This task, like the previous one, uses the official train/test split of the MIT Indoor scene dataset for its supervised training and evaluation of scene categorization performance. For the initial unsupervised feature extraction however, we augmented the MIT Indoor training set with the whole dataset of images from SUN-397 (108, 754 images). This yields a total of 123, 034 images for unsupervised feature learning and corresponds to a *semi-supervised* setting. Our motivation for adding scene images from SUN, besides increasing the number of training samples, is that on MIT Indoor, which contains only indoor scenes, OB detectors specialized on outdoor objects would likely be mostly inactive (as a sailboat detector applied on indoor scenes) and irrelevant, introducing an harmful noise in the unsupervised feature learning. As SUN is composed of a wide range of indoor and outdoor scene images, its addition to MIT Indoor ensures that

	UIUC-Sports	15-SCENES
object-MAX + SVM	$67.23 \pm 1.29\%$	$71.08 \pm 0.57\%$
object-MEAN + SVM	$81.88 \pm 1.16\%$	$83.17 \pm 0.53\%$
object-PCA + SVM	$83.90 \pm 1.67\%$	$85.58 \pm 0.48\%$
pose-PCA + SVM	$83.81 \pm 2.22\%$	$85.69 \pm 0.39\%$
pose-PCA + MLP	$84.29 \pm 2.23\%$	$84.93 \pm 0.39\%$
pose-PCA + CAE (MLP)	$\mathbf{85.13 \pm 1.07\%}$	$\mathbf{86.44 \pm 0.21\%}$
Object Bank + SVM	78.90%	80.98%
Object Bank + rbf-SVM	$78.56 \pm 1.50\%$	$83.71 \pm 0.64\%$
Improvement w.r.t. OB	+6.23%	+5.46%

Table 6.2 – **UIUC Sports and 15-Scenes** Results are reported for 10 random splits (available at www.anonymous.org) and compared to the original OB results (Li-Jia Li and Fei-Fei, 2010a) - Object Bank + SVM - on one single split.

each detector meaningfully covers its whole range of activity (having a ”balanced” number of positives/negatives detections through the training set) and the feature extraction methods can be efficiently trained to capture it.

One may object that training on additional images does not provide a fair comparison w.r.t. the original OB method. Nevertheless, we recall that (1) the supervised classifiers do not benefit from these additional examples and (2) object detectors which are the core of OB representations (and all detector-based approaches) have also obviously been trained on *additional* images.

UIUC-Sports and 15-Scenes (transfer). We would also like to evaluate the discriminative power of the various representations learned on the MIT+SUN dataset, but on new scene images and categories that were *not* part of the MIT+SUN dataset. This might be useful in case other researchers would like to use our compact representation on a different set of images. Using the representation output by the feature extractors learned with MIT+SUN, we train and evaluate classifiers for scene categorization on images from UIUC-Sports and 15-Scenes (not used during unsupervised training). This corresponds to a *transfer learning* setting for the feature extractors.

Object-Bank	Pooling	*whole*-PCA	*object*-PCA	*pose*-PCA	*pose*-PCA+CAE
$44,604$	177	$1,300$	$2,655$	$1,770$	$1,000$

Table 6.3 – Dimensionality Reduction. Dimension of representations obtained on MIT Indoor. The *pose*-PCA+CAE produces a compact and powerful combination.

6.5.3 SVMs on Features Learned with each Strategy

In order to evaluate the quality of the features generated by each strategy, a linear SVM is trained on the features extracted by each combination method. We used LibLinear (Fan et al., 2008) as SVM solver and chose the best C according to 5-fold cross-validation scheme. We compare accuracies obtained by features provided by all considered combination methods against the original OB performances (Li-Jia Li and Fei-Fei, 2010a). Results obtained with SVM classifiers on all MIT-related tasks are displayed in Table 6.1 and those concerning UIUC and 15-scenes in Table 6.2.

The simplistic strategy *object* mean-pooling performs surprisingly well on all datasets and tasks whereas *object* max-pooling obtained the worst results. It suggests that taking the mean response of an object detector across various scales and positions is actually meaningful compared to consider presence/absence of objects as max-pooling does.

On MIT and MIT+SUN, *object* or *pose* PCAs reach almost the same range of performance slightly above the current state-of-the-art performances (Pandey and Lazebnik, 2011), except for whole-PCA which performs poorly: one must consider the structure of OB to combine features efficiently. In the experiments, keeping the 10 (resp. 15) first principal components gave us the best results for pose-PCA (resp. object-PCA).

Besides, Table 6.3 shows that both PCAs and PCA+CAE allow a huge reduction of the dimension of the OB feature representation.

Results obtained for the UIUC-Sports and 15-Scenes transfer learning tasks are displayed in Table 6.2. Representations learned on MIT+SUN generalize quite well and can be easily used for other datasets even if images from those datasets have not been seen at all during unsupervised learning.

6.5.4 Deep Learning with Fine Tuning

Previous work (Larochelle et al., 2009) on Deep Learning generally showed that the features learned through unsupervised learning could be improved upon by fine-tuning them through a supervised training stage. In this stage (which follows the unsupervised pre-training stage), the features and the classifier on top of them are together considered to be a supervised neural network, a Multi-Layer Perception (MLP) whose hidden layer is the output of the trained features. Hence we apply this strategy to the *pose* PCA+CAE architecture, keeping the PCA transformation fixed but fine-tuning the CAE and the MLP altogether. These results are given at the bottom of tables 6.1 and 6.2. The MLP are trained with early stopping on a validation set (taken from the original training set) for 50 epochs.

This yields 44.0% test accuracy on plain MIT and 49.1% on MIT+SUN: this allows to obtain state-of-the-art performance, with or without semi-supervised training of the CAEs, even if these additional examples are highly beneficial. As a check, we also evaluate the effect of the unsupervised pre-training stage by completely skipping it and only training a regular supervised MLP of 1000 hidden units on top of the PCA output, yielding a worse test accuracy of 42.9% on MIT and 46.3% on MIT+SUN. This improvement with fine-tuning on labeled data is a great advantage for CAE compared to PCA. Fine-tuning is also beneficial on UIUC-Sports and 15-Scenes. On both datasets, this leads to an improvement of +6% and +5% w.r.t the original system.

Finally, we trained a non-linear SVM (with rbf kernel) to verify whether this gap in performances was simply due to the replacement of a linear classifier (SVM) by a non-linear one (MLP) or to the detectors' outputs combination. The poor results of the rbf-SVM (see tables 6.1 and 6.2) suggests that the careful combination strategies are essential to reach good performance.

6.5.5 Use of External Semantic Information for Re-Ranking

WordNet's (Miller, 1995) semantic structure provide an easy way to measure word similarities. We assume that closely related objects detectors (according to WordNet) should fire together and could be grouped in order to build semantically meaningful features. E.g. by grouping the output of *ship*, *sea* and *sun* into a single feature, the combination's output might be useful for classifying the "sailing" scene

Context Semantics learned by the CAE
sailboat, rock, tree, coral, blind
roller coaster, building, rail, keyboard, bridge
sailboat, autobus, bus stop, truck, ship
curtain, bookshelf, door, closet, rack
soil, seashore, rock, mountain, duck
attire, horse, bride, groom, bouquet
bookshelf, curtain, faucet, screen, cabinet
desktop computer, printer, wireless, computer screen

Table 6.4 – Context semantics: Names of the detectors corresponding to the highest weights of 8 hidden units of the CAE. These hidden units will fire when those objects will be detected altogether.

category.

In our experiments, we used the lesk distance in WordNet to extract the neighbors of each detector's name. Some examples are depicted in Table 6.5. Afterwards, given the score $s(x) \in \mathbb{R}^{177}$ obtained with the mean-pooling strategy from the original OB representation $x \in \mathbb{R}^{44,604}$, we performed the following Re-Ranking operation:

$$s_i^{'}(x) = \sum_{j=1}^{177} s_j(x)\gamma^{R(i,j)} \text{ for } i = 1, \ldots, 177 \tag{6.4}$$

where $\gamma \in [0, 1]$ is a decay hyper-parameter tuned on a validation set. $R(i, j)$ corresponds to the rank of the object j among the neighbors of object i according to the lesk metric ($R(i, i) = 0$). Results are presented in Table 6.6. The relatively small improvement brought by WordNet illustrates the fact that the poor intrinsic quality of the object detectors prevents any use of external semantic resource to improve their combination.

6.6 Discussion

In this work, we add one or more levels of trained representations on top of the layer of object and part detectors (OB features) that have constituted the basis

Rank	**bus**	**lion**	**laptop**
1.	car	tree	baggage
2.	ship	dog	desktop computer
3.	truck	bird	computer
4.	aircraft	horse	bed
5.	train	computer	door

Table 6.5 – **WordNet semantics**: Names of the detectors and their top-ranked neighbors according to the lesk distance computed from WordNet.

object-MEAN+SVM	MIT *(plain)*
w/o Re-Ranking	41.03%
with Re-Ranking	41.52%

Table 6.6 – **Re-Ranking**: Results are reported on the official split (Quattoni and Torralba, 2009). Object-mean+SVM refers to the mean-pooling strategy with and w/o the Re-Ranking transformation.

of very promising trend of approach for scene classification (Li-Jia Li and Fei-Fei, 2010a). These higher-level representations are mostly trained in an unsupervised way, following the trend of so-called Deep Learning (Hinton et al., 2006b; Bengio, 2009a; Jarrett et al., 2009), but can be fine-tuned using the supervised detection objective.

These learned representations capture statistical dependencies in the co-occurrence of detections the object detectors from (Li-Jia Li and Fei-Fei, 2010a). In fact, one can see in Table 6.4 plausible contexts of joint appearance of several objects learned by the CAE. These detectors, which can be quite imperfect when seen as actual detectors, contain a lot of information when combined altogether. However, the uncertainty of detectors makes it hard to combine using external semantic sources such as WordNet. As reported in Table 6.6, we observe a slight improvement (+0.5%) using our Re-Ranking strategy and lesk words' similarities. The extraction of those *context semantics* with unsupervised feature-learning algorithms has empirically shown better performances but these semantics are inherent to the detectors outputs and can not be easily combined with any known predefined semantic system such as the one defined in WordNet.

In particular, we find that Contractive Auto-Encoder (Rifai et al., 2011) can substantially improve performance on top of *pose* PCAs as a way to extract non-linear dependencies between these lower-level OB detectors (especially when fine-tuned). They also improve greatly upon the use of the detectors as inputs to an SVM or a logistic regression (which were, with structured regularization, the original methods used by OB).

This trained post-processing allows us to reach the state-of-the-art on MIT Indoor and UIUC (85.13% against 85.30% obtained by LScSPM (Gao et al., 2010)) while being competitive on 15-scenes (86.44% also versus 89.70% LScSPM). On these last two datasets, we reach the best performance for methods only relying on object/part detectors. Compared to other kinds of methods, we are limited by the accuracy of those detectors (only trained on HOG features), whereas competitive methods can make use of other descriptors such as SIFT (Gao et al., 2010), known to achieve excellent performance in image recognition.

Besides its good accuracies, it is worth noting that the feature representation obtained by the *pose* PCA+CAE is also very compact, allowing a 97% reduction compared to the original data (see Table 6.3). Handling a dense input of dimension $44,604$ is not a common thing. By providing this compact representation, we think that researchers will be able to use the rich information provided by OB in the same way they use low-level image descriptors such as SIFT.

As future work, we are planning other ways of combining OB features e.g. considering the output of all detectors at a given scale and position and combine them afterwards in a hierarchical manner. This would be a kind of dual view of the OB features. Other plausible departures could take into account the topology (e.g. spatial structure) of the pattern of detections, rather than treat the response at each location and scale as an attribute and the set of attributes as unordered. This could be done in the same spirit as in Convolutional Networks (LeCun et al., 1999), aggregating the responses for various objects detectors/locations/scales in a way that takes explicitly into account the object category, location and scale of each response, similarly to the way filter outputs at neighboring locations are pooled in each layer of a Convolutional Network.

6.7 Acknowledgments

We would like to thank Gloria Zen for her helpful comments. This work was supported by NSERC, CIFAR, the Canada Research Chairs, Compute Canada and by the French ANR Project ASAP ANR-09-EMER-001. Codes for the experiments have been implemented using Theano (Bergstra et al., 2010a) Machine Learning library.

7 Préambule au Troisième Article

Les deux premiers articles résument nos approches pour des problématiques différentes: des entrées où aucun a priori n'est disponible ou au contraire, des données dont la structure peut être utilisée à notre avantage. Alors que les précédents travaux focalisent sur l'aspect technique de différents algorithmes d'apprentissage, ce troisième article s'intéresse à nos intuitions concernant la structure de l'espace sémantique de représentation appris. En particulier, nous nous intéressons à la capacité de visiter des classes différentes lors du processus d'échantillonnage à partir de l'espace de représentation sémantique.

7.1 Détails de l'article

Better Mixing via Deep Representations Yoshua Bengio [1], Grégoire Mesnil[1], Yann Dauphin and Salah Rifai. *International Conference on Machine Learning* 2013

Contribution Personnelle Le point de départ de cette idée a été présenté à Snowbird à un workshop (Mesnil et al., 2012). Cela consistait à montrer sur support vidéo comment il était possible de se déplacer d'un exemple à un autre dans l'espace des représentations tout en restant sur la variété des exemples d'apprentissage. Cela a renforcé les intuitions de Yoshua concernant un problème de mixing entre différents modes et pourquoi il était plus aisé de visiter toutes les classes lors du processus d'échantillonnage dans l'espace abstrait des représentations. À partir de modèles de RBMs et de CAEs entraînés par Salah Rifai et Yann Dauphin, j'ai réalisé et conçu l'intégralité des expériences visant à vérifier les hypothèses de Yoshua, qui

1. indique une contribution similaire

lui a rédigé l'article. Il a été décidé ensemble que nous avions une contribution égale pour cet article.

7.2 Contexte

On a toujours peu d'intuitions sur ce que les représentations apprises par les réseaux de neurones contiennent exactement ou sur la structure particulière de l'espace des représentations. Cette publication visait à présenter et valider nos intuitions:

Hypothèse 1: Profondeur et Mixage durant l'échantillonnage.
Une architecture profonde apprend un espace sémantique dans lequel les chaînes de Markov mixent plus rapidement entre différentes classes.

Hypothèse 2: L'espace sémantique appris rend la variété initiale localement linéaire.
La variété des exemples d'apprentissage complexe et non linéaire dans l'espace initial est rendue localement linéaire dans l'espace sémantique défini par les couches abstraites. Le volume de l'espace sémantique est occupé de façon plus uniforme que l'espace d'entrée.

7.3 Contributions

Ce travail a été cité par plusieurs articles inspirés par cette hypothèse de remplissage de l'espace de représentation et de structure de la variété des exemples d'apprentissage.

7.4 Récents développements

Ces idées sont toujours d'actualité et restent dans l'esprit des chercheurs pour guider leurs intuitions. Le concept de naviguer sur la variété dans l'espace des représentations a notamment été repris par Ian Goodfellow[2] pour présenter la variété des chiffres appris par les Adversarial Networks (Goodfellow et al., 2014).

2. https://plus.google.com/+IanGoodfellow/posts/SJxfk4SeNi7

8 Better Mixing via Deep Representations

8.1 Introduction and Background

Deep learning algorithms attempt to discover multiple levels of representation of the given data (see (Bengio, 2009a) for a review), with higher levels of representation defined hierarchically in terms of lower level ones. The central motivation is that higher-level representations can potentially capture relevant higher-level abstractions. Mathematical results in the case of specific function families have shown that choosing a sufficient depth of representation can yield exponential benefits, in terms of size of the model, to represent some functions (Håstad, 1986; Håstad and Goldmann, 1991; Bengio et al., 2006; Bengio and LeCun, 2007; Bengio and Delalleau, 2011). The intuition behind these theoretical advantages is that lower-level features or latent variables can be *re-used* in many ways to construct higher-level ones, and the potential gain becomes exponential with respect to depth of the circuit that relates lower-level features and higher-level ones (thanks to the exponential number of paths in between). The ability of deep learning algorithms to construct abstract features or latent variables on top of the observed variables relies on this idea of re-use, which brings with it not only computational but also statistical advantages in terms of *sharing of statistical strength*, e.g., as already exploited in multi-task learning (Caruana, 1995; Baxter, 1997; Collobert and Weston, 2008) and learning algorithms involving *parameter sharing* (Lang and Hinton, 1988; LeCun, 1989).

There is another – less commonly discussed – motivation for deep representations, introduced in Bengio (2009a): the idea that they may help to *disentangle* the underlying factors of variation. Clearly, if we had learning algorithms that could do a good job of discovering and separating out the underlying causes and factors of variation present in the data, it would make further processing (typically, taking decisions) much easier. One could even say that the ultimate goal of AI research is to build machines that can understand the world around us, i.e., disentangle the factors and causes it involves, so progress in that direction seems important. If learned representations do a good job of disentangling the underlying factors of variation, earning (on top of these representations, e.g., towards specific tasks of interest) becomes substantially easier because disentangling counters the effects of the curse of dimensionality. With good disentangling, there is no need for further learning, only good inference. Several observations suggest that some deep learning algorithms indeed help to disentangle the underlying factors of variation (Goodfellow et al., 2009; Glorot et al., 2011b). However, it is not clear why, and to what extent in general (if any), different deep learning algorithms may sometimes help this disentangling.

Many deep learning algorithms are based on some form of unsupervised learning, hence capturing salient structure in the data distribution. Whereas deep learning algorithms have mostly been used to learn features and exploit them for classification or regression tasks, their unsupervised nature also means that in several cases they can be used to generate samples. In general the associated sampling algorithms involve a Markov Chain and MCMC techniques, and these can notoriously suffer from a fundamental problem of *mixing between modes*: it is difficult for the Markov chain to jump from one mode of the distribution to another, when these are separated by large low-density regions, a common situation in real-world data, and under the *manifold hypothesis* (Cayton, 2005; Narayanan and Mitter, 2010). This hypothesis states that natural classes present in the data are associated with low-dimensional regions in input space (manifolds) near which the distribution concentrates, and that different class manifolds are well-separated by regions of very low density. Slow mixing between modes means that consecutive samples tend to be correlated (belong to the same mode) and that it takes many consecutive sampling steps to go from one mode to another and even more to cover all of them, i.e., to obtain a large enough representative set of samples (e.g. to compute an expected value under the target distribution). This happens because these jumps through the empty low-density void between modes are unlikely and rare events. When a learner has a poor model of the data, e.g., in the initial stages of learning, the model tends to correspond to a smoother and higher-entropy (closer to uniform) distribution, putting mass in larger volumes of input space, and in particular, between the modes (or manifolds). This can be visualized in generated samples of images, that look more blurred and noisy. Keep in mind that MCMCs tend to make moves to *nearby probable configurations*. Mixing between modes is therefore initially easy for such poor models. However, as the model improves and its corresponding distribution sharpens near where the data concentrate, mixing between modes becomes considerably slower. Making one unlikely move (i.e., to a low-probability configuration) may be possible, but making N such moves becomes exponentially unlikely in N, as illustrated in Figure 8.1. Since sampling is an integral part of many learning algorithms (e.g., to estimate the log-likelihood gradient), slower mixing between modes then means slower or poorer learning, and one may even suspect that learning stalls at some point because of the limitations of the sampling algorithm. To improve mixing between modes, a powerful idea that has been explored recently for deep learning algorithms (Desjardins et al., 2010; Cho et al., 2010; Salakhutdinov, 2010b,a) is *tempering* (Neal, 1994). The idea is to use smoother densities (associated with *higher temperature* in a Boltzmann Machine or Markov Random Field formulation) to make quick but approximate jumps between modes, but use the sharp "correct" model to actually generate the samples of interest around these modes, and allow samples to be exchanged between the different levels of temperature.

Here we want to discuss another possibly related idea, and claim that *mixing between modes is easier when sampling at the higher levels of representation*. The objective is not to propose a new sampling algorithm or a new learning algorithm, but rather to investigate this hypothesis through experiments using existing deep learning algorithms. The objective is to further our understanding of this hypothesis through more specific hypotheses aiming to explain why this would happen, using further experimental validation to test these more specific hypotheses. The idea that deeper generative models produce not only better features for classification but also better quality samples (in the sense of better corresponding to the target distribution being learned) is not novel and several observations support this hypothesis already, some quantitatively (Salakhutdinov and Hinton, 2009), some more qualitative (Hinton et al., 2006b). The specific contributions of this paper is to focus on why the samples may be better, and in particular, why the chains may converge faster, based on the previously introduced idea that deeper representations can do a better job of disentangling the underlying factors of representation.

8.2 Hypotheses

We first clarify the first hypothesis to be tested here.

Hypothesis H1: Depth vs Better Mixing Between Modes. A successfully trained deeper architecture has the potential to yield representation spaces in which Markov chains mix faster between modes.

If experiments validate that hypothesis, the most important next question is: why ? The main explanation we conjecture is formalized in the following hypothesis.

Hypothesis H2: Depth vs Disentangling. Part of the explanation of **H1** is that deeper representations can better disentangle the underlying factors of variation.

Why would that help to explain **H1** ? Imagine an abstract (high-level) representation for object image data in which one of the factors is the “reverse video bit”, which inverts black and white, e.g., flipping that bit replaces intensity $x \in [0, 1]$ by $1-x$. With the default value of 0, the foreground object is dark and the background is light. Clearly, flipping that bit does not change most of the other semantic characteristics of the image, which could be represented in other high-level features. However, for every image-level mode, there would be a reverse-video counterpart mode in which that bit is flipped: these two modes would be separated by vast “empty” (low density) regions in input space, making it very unlikely for any Markov chain in input space (e.g. Gibbs sampling in an RBM) to jump from one of these two modes to another, because that would require most of the input pixels or hidden units of the RBM to simultaneously flip their value. Instead, if we consider the high-level representation which has a “reverse video” bit, flipping only that bit would be a very likely event under most Markov chain transition probabilities, since that flip would be a small change preserving high probability. As another example, imagine that some of the bits of the high-level representation identify the category of the object in the image, independently of pose, illumination, background, etc. Then simply flipping one of these object-class bits would also drastically change the raw pixel-space image, while keeping likelihood high. Jumping from an object-class mode to another would therefore be easy with a Markov chain in representation-space, whereas it would be much less likely in raw pixel-space.

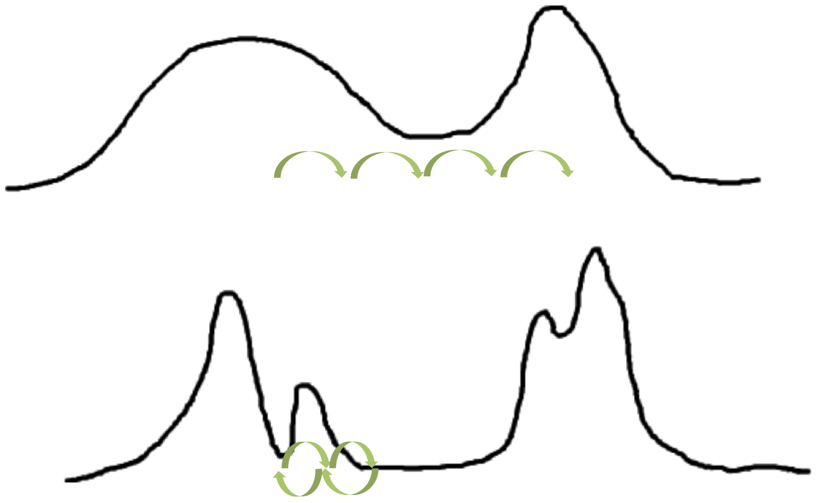


Figure 8.1 – Top: early during training, MCMC mixes easily between modes because the estimated distribution has high entropy and puts enough mass everywhere for small-steps movements (MCMC) to go from mode to mode. Bottom: later on, training relying on good mixing can stall because estimated modes are separated by vast low-density deserts.

Another point worth discussing (and which should be considered in future work) in **H2** is the notion of *degree* of disentangling. Although it is somewhat clear what a completely disentangled representation would look like, deep learning algorithms are unlikely to do a perfect job of disentangling, and current algorithms do it in stages, with more abstract features being extracted at higher levels. Better disentangling would mean that *some* of the learned features have a higher mutual information with *some* of the known factors. One would expect at the same time that the features that are highly predictive of one factor be less so of other factors, i.e., that they specialize to one or a few of the factors, becoming *invariant* to others. Please note here the difference between the objective of learning disentangled representations and the objective of learning invariant features (i.e., invariant to some specific factors of variation which are considered to be like nuisance parameters). In the latter case, one has to know ahead of time what the nuisance factors are (what is noise and what is signal ?). In the former, it is not needed: we only seek to separate out the factors from each other. Some features should be sensitive to one factor and invariant to the others.

Let us now consider additional hypotheses that specialize **H2**.

Hypothesis H3: Disentangling Unfolds and Expands. Part of the explanation of **H2** is that more disentangled representations tend to **(a)** unfold the manifolds near which raw data concentrates, as well as **(b)** expand the relative volume occupied by high-probability points near these manifolds.

H3(a) says is that disentangling has the effect that the projection of high-density manifolds in the high-level representation space have a smoother density and are easier to model than the corresponding high-density manifolds in raw input space. Let us again use an object recognition analogy. If we have perfectly disentangled object identity, pose and illumination, the high-density manifold associated with the distribution of features in high-level representation-space is flat: we can interpolate between some training examples (i.e. likely configurations) and yet stay in a high-probability region. For example, we can imagine that interpolating between two images of the same object at different poses (lighting, position, etc.) in a high-level representation-space would yield images of the object at intermediate poses (i.e., corresponding to likely natural images), whereas interpolating in pixel space would give a superposition of the two original images (i.e., unlike any natural image). If interpolating between high-probability examples (i.e. within their convex set) gives high-probability examples, then it means that the distribution is more uniform (fills the space) within that convex set, which is what **H3(b)** is saying. In addition, a good high-level representation does not need to allocate as much real estate (sets of values) for unlikely configurations. This is already what most unsupervised learning algorithms tend to do. For example, dimensionality reduction methods such as the PCA tend to define representations where most configurations are likely (but these only occupy a subspace of the possible raw-space configurations). Also, in clustering algorithms such as k-means, the training criterion is best minimized when clusters are approximately equally-weighted, i.e., the average posterior distribution over cluster identity is approximately uniform. Something similar is observed in the brain where different areas of somatosensory cortex correspond to different body parts, and the size of these areas adaptively depends (Flor, 2003) on usage of these (i.e., more frequent events are represented more finely and less frequent ones are represented more coarsely). Again, keep in mind that the actual representations learned by deep learning algorithms are not perfect, but what we will be looking for here is whether deeper representations correspond to more unfolded manifolds and to more locally uniform distributions, with high-probability points occupying an overall greater volume (compared to the available volume).

8.3 Representation-Learning Algorithms

The learning algorithms used in this paper to explore the preceding hypotheses are the Deep Belief Network or DBN (Hinton et al., 2006b), trained by stacking Restricted Boltzmann Machines or RBMs, and the Contractive Auto-Encoder or CAE (Rifai et al., 2011), for which a sampling algorithm was recently proposed (Rifai et al., 2012). In the experiments, the distribution under consideration is the asymptotic distribution associated with the stochastic process used to generate samples. In the case of DBNs it clearly corresponds to the analytically defined distribution associated with the formula for the DBN probability. The Markov transition operator for DBNs is the one associated with Gibbs sampling (in the top RBM) (Hinton et al., 2006b). The Markov transition operator for stacked CAEs has been spelled out in Rifai et al. (2012) and linked to Langevin MCMC in Alain and Bengio (2012).

Each layer of the DBN is trained as an RBM, and a 1-layer DBN is just an RBM. An RBM defines a joint distribution between a hidden layer h and a visible layer v. Gibbs sampling at the top level of the DBN is used to obtain samples from the model: the sampled top-level representations are stochastically projected down to lower levels through the conditional distributions $P(v|h)$ defined in each RBM. To avoid unnecessary additional noise, and like previous authors have done, at the last stage of this process (i.e. to obtain the raw-input level samples), only the mean-field values of the visible units are used, i.e., $E[v|h]$. In the experiments on face data (where grey levels matter a lot), a Gaussian RBM is used at the lowest level.

An auto-encoder (LeCun, 1987; Hinton and Zemel, 1994) is parametrized through an encoder function f mapping input-space vector x to representation-space vector h, and a decoder function g mapping representation-space vector h to input-space reconstruction r. The experiments with the CAE are with $h = f(x) = \text{sigmoid}(Wx + b)$ and $r = g(h) = \text{sigmoid}(W^T h + c)$. The CAE is a regularized auto-encoder with tied weights (input to hidden weights are the transpose of hidden to reconstruction weights). Let $J = \frac{\partial f(x)}{\partial x}$ the Jacobian matrix of the encoder function. The CAE is trained to minimize a cross-entropy reconstruction loss plus a contractive regularization penalty $\alpha||J||_F^2$ (the sum of the squared elements of the Jacobian matrix). Like RBMs, CAE layers can be stacked to form deeper models, and one can either view them as deep auto-encoders (by composing the encoders together and the decoders together) or like in a DBN, as a top-level generative model (from which one can sample) coupled with encoding and decoding functions into and from the top level (by composing the lower-level encoders and decoders). A sampling algorithm was recently proposed for CAEs (Rifai et al., 2012), alternating between projecting through the auto-encoder (i.e. performing a reconstruction) and adding Gaussian noise $JJ^T\epsilon$ in the directions of variation captured by the auto-encoder.

8.4 Experiments

The experiments have been performed on the MNIST digits dataset (LeCun et al., 1998b) and the Toronto Face Database (Susskind et al., 2010), TFD. The former has been heavily used to evaluate many deep learning algorithms, while the latter is interesting because of the manifold structure it displays, and for which the main control factors (such as emotion and person identity) are known.

We have varied the number of hidden units at each layer independently for shallow and deep networks, and the models that gave best validation performance for each depth are shown. The qualitative aspects of the results were insensitive to layer size. The results reported are for DBNs with 768-1024-1024 layer sizes (28×28 input) on MNIST, and 2304-512-1024 on TFD (48×48 input). The CAEs have sizes 768-1000-1000 and 2304-1000-1000 on MNIST and TFD respectively.

8.4.1 Sampling at Different Depths

Better Samples at Higher Levels

To test **H1**, we first plot sequences of samples at various depths. One can verify in Fig. 8.2 that samples obtained at deeper layers are visually more likely and mix faster between modes.

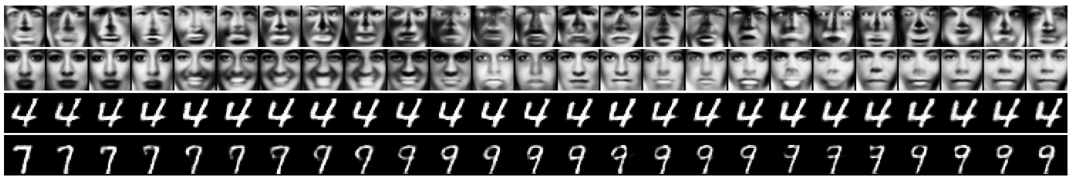


Figure 8.2 – Sequences of 25 samples generated with a CAE on TFD (rows 1 and 2, respectively for 1 or 2 hidden layers) and with an RBM on MNIST (rows 3 and 4, respectively for 1 or 2 hidden layers). On TFD, the second layer clearly allows to get quickly from woman samples (left) to man samples (right) passing by various facial expressions whereas the single hidden layer model shows poor samples. Bottom rows: On MNIST, the single-layer model gets stuck near the same mode while the second layer allows to mix among classes.

In addition, we measure the quality of the obtained samples, using a procedure for the comparison of sample generators described in Breuleux et al. (2011). Note that the mixing properties measured here are a consequence of the underlying models as well as of the chosen sampling procedures. For this reason, we have chosen to monitor the *quality of the samples* with respect to the original data generating distribution that was used to train the model. The procedure of Breuleux et al. (2011) measures the log-likelihood of a test set under the density computed from a Parzen window density estimator built on generated samples ($10,000$ samples here). Log-likelihoods for different models are presented in Table 8.1 (rightmost columns). Those results also suggest that the quality of the samples is higher if the Markov chain process used for sampling takes place in the upper layers.

This observation agrees with **H3(b)** that moving in higher-level representation spaces where the manifold has been expanded provides higher quality samples than moving in the raw input space where it may be hard to stay in high density regions.

Visualizing Representation-Space by Interpolating Between Neighbors

According to **H3(a)**, deeper layers tend to locally unfold the manifold near high-densities regions of the input space, while according to **H3(b)** there should be more relative volume taken by plausible configurations in representation-space. Both of these would imply that convex combinations of neighboring examples in representation-space correspond to more likely input configurations. Indeed, interpolating between points on a flat manifold should stay on the manifold. Furthermore, when interpolating between examples of different classes (i.e., different modes), **H3(b)** would suggest that most of the points in between (on the linear interpolation line) should correspond to plausible samples, which would not be the case in input space. In Fig. 8.3, we interpolate linearly between neighbors in representation space and visualize in the input space the interpolated points obtained at various depths. One can see that interpolating at deeper levels gives visually more plausible samples.

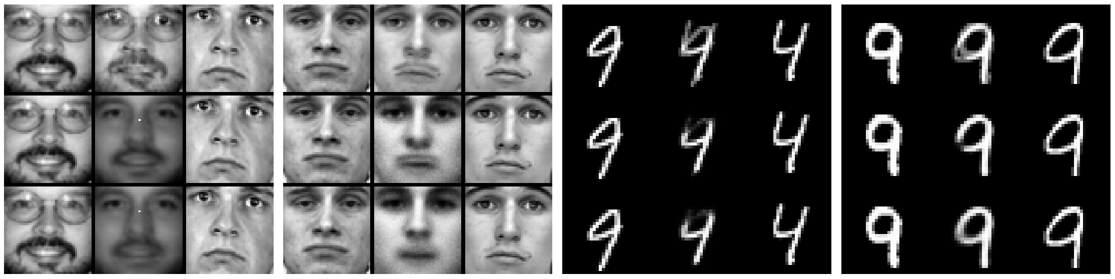
8.4.2 Measuring Mixing Between Modes by Counting Number of Classes Visited

We evaluate here the ability of mixing among various classes. We consider sequences of length 10, 20 or 100 and compute histograms of number of different classes visited in a sequence, for the two different depths and learners, on TFD. Since classes typically are in different modes (manifolds), counting how many different classes are visited in an MCMC run tells us how quickly the chain mixes between modes. We have chosen this particular method to monitor mixing modes because it focuses more directly on visits to modes, instead of the traditional autocorrelation of the samples (which measures how fast the samples change). Fig. 8.4(c,f) show that the deeper architectures visit more classes and the CAE mixes faster than the DBN.

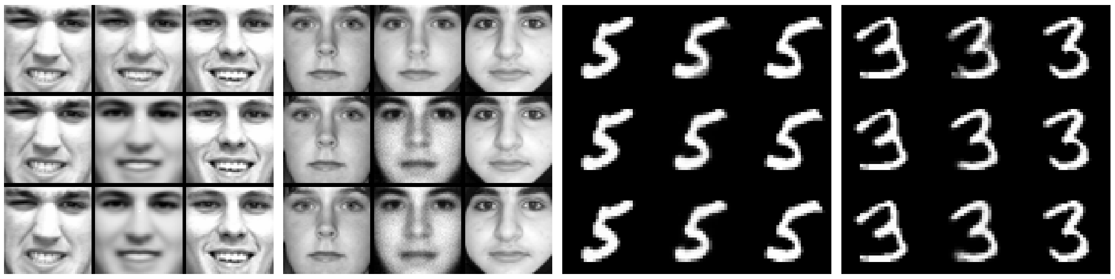
8.4.3 Occupying More Volume Around Data Points

In these experiments (Fig. 8.4 (a,b,d,e)) we estimate the quality of samples whose representation is in the neighborhood of training examples, at different levels of representation. In the first setting (Fig. 8.4 (a,b)), the samples are interpolated at the midpoint between an example and its k-th nearest neighbor, with k on the x-axis. In the second case (Fig. 8.4 (d,e)), isotropic noise is added around an example, with noise standard deviation on the x-axis. In both cases, 500 samples are generated for each data point plotted on the figures, with the y-axis being the log-likelihood introduced earlier, i.e., estimating the quality of the samples. We find that on higher-level representations of both the CAE and DBN, a much larger proportion of the local volume is occupied by likely configurations, i.e., closer to the input-space manifold near which the actual data-generating distribution concentrates. Whereas the first experiment shows that this is true in the convex set between neighbors at different distances (i.e., in the directions of the manifold), the second shows that this is also true in random directions locally (but of course likelihoods

(a) Interpolating between an example and its 200-th nearest neighbor (see caption below).



(b) Interpolating between an example and its nearest neighbor.



(c) Sequences of points interpolated at different depths

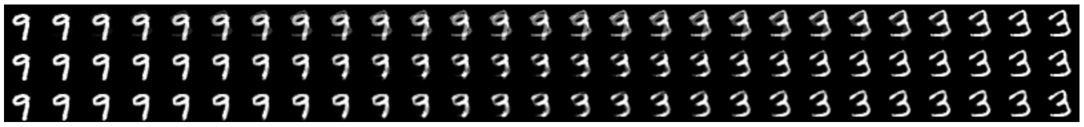


Figure 8.3 – Linear interpolation between a data sample and the 200-th (a) and 1st (b) nearest neighbor, using representations at various depths (top row=input space, middle row=1st layer, bottom row=2nd layer). In each 3×3 block the left and right columns are test examples while the middle column is the image obtained by interpolation, based on different levels of representation. Interpolating at higher levels clearly gives more plausible samples. Especially in the raw input space (e.g., (a), 2nd block), one can see two mouths overlapping while only one mouth appears for the interpolated point at the 2nd layer. Interpolating with the 1-nearest neighbor does not show any difference between the levels because the nearest neighbors are close enough for a linear interpolation to be meaningful, while interpolaing with the 200-th nearest neighbors shows the failure of interpolation in raw input space but successful interpolation in deeper levels. In (c), we interpolate between samples of *different classes*, at different depths (top=raw input, middle=1st layer, bottom=2nd layer). Note how in lower levels one has to go through unplausible patterns, whereas in the deeper layers one almost jumps from a high-density region of one class to another (of the other class).

are also worse there). The first result therefore agrees with **H3(a)** (unfolding) and **H3(b)** (volume expansion), while the second result mostly confirms **H3(b)**.

	Classification				**Log-likelihood**	
	MNIST		**TFD**		**MNIST**	**TFD**
	SVM	MLP+	SVM	MLP+		
raw	8.34%	-	33.48 ± 2.14 %	-	-	-
CAE-1	1.97%	1.14%	25.44 ± 2.45%	24.12 ± 1.87 %	67.69 ± 2.87	591.90 ± 12.27
CAE-2	1.73%	0.81%	24.76 ± 2.46%	23.73 ± 1.62%	121.17 ± 1.59	2110.09 ± 49.15
DBN-1	1.62%	1.21%	26.85 ± 1.62%	28.14± 1.40	-243.91 ± 54.11	604 ± 14.67
DBN-2	1.33%	0.99%	26.54 ± 1.91%	27.79 ± 2.34	137.89 ± 2.11	1908.80 ± 65.94

Table 8.1 – Left: Classification rates of various classifiers (SVM, MLP+) using representations at various depth (with CAE or DBN) learned on the MNIST and TFD datasets. The DBN 0.99% error on MNIST has been obtained with a 3-layer DBN and the 0.81% error with the Manifold tangent Classifier (Rifai et al., 2011) that is based on a CAE-2 and discriminant fine-tuning. MLP+ uses discriminant fine-tuning. Right: Log-likelihoods from Parzen-Windows density estimators based on 10,000 samples generated by each model. This quantitatively confirms that the samples generated from deeper levels are of higher quality, in the sense of better covering the zones where test examples are found.

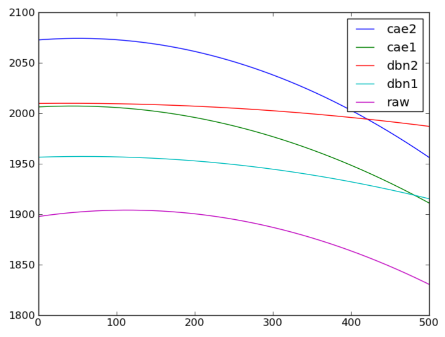
8.4.4 Discriminative Ability vs Volume Expansion

Hypothesis **H3** could arguably correspond to worse discriminative power [1]: if on the higher-level representations the different classes are "closer" to each other (making it easier to mix between them), would it not mean that they are more confusable ? We first confirm with the tested models (as a sanity check) that the deeper level features are conducive to better classification performance, in spite of their better generative abilities and better mixing between modes.

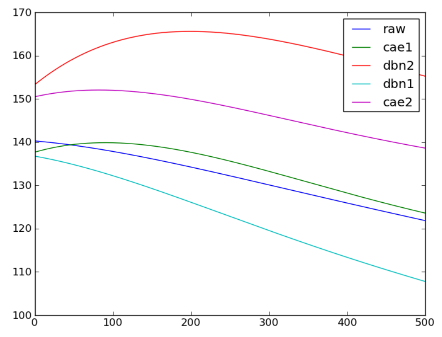
We train a linear SVM on the concatenation of the raw input with the upper layers representations (which worked better than using only the top layer, a setup already used successfully when there is no supervised fine-tuning (Lee et al., 2009)). Results presented in Table 8.1 show that the representation is more linearly separable if one increases the depth of the architecture and the information added by each layer is helpful for classification. Also, fine-tuning a MLP initialized with those weights is still the best way to reach state-of-the-art performances.

To explain the good discriminant abilities of the deeper layers (either when concatenated with lower layers or when fine-tuned discriminatively) in spite of the

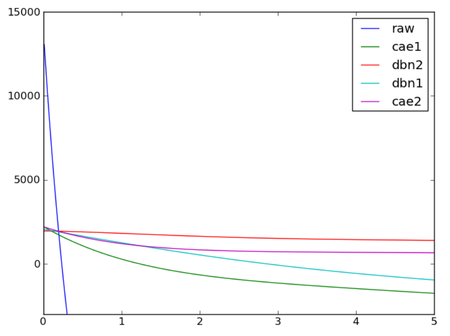
1. as pointed out by Aaron Courville, personal communication



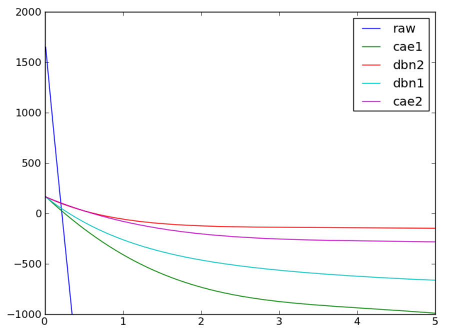
(a) TFD



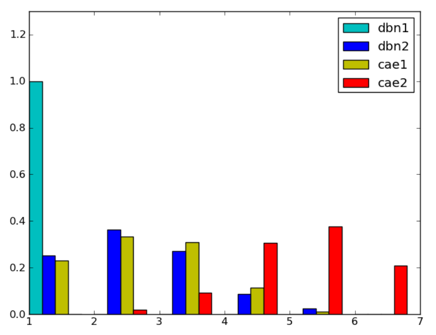
(b) MNIST



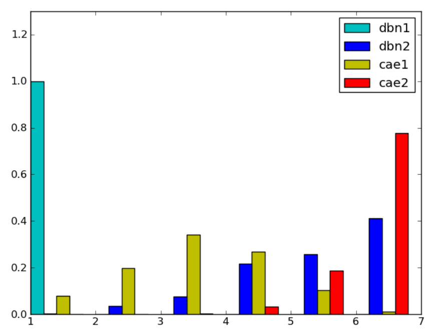
(c) TFD



(d) MNIST



(e) Mixing - 10 samples



(f) Mixing - 20/100 samples

Figure 8.4 – (a) (b) Local Convex Hull - Log-density computed w.r.t. linearly interpolated samples between an example and its k-NNs, for k (x-axis) between 1 and 500. The manifold thus seem generally more unfolded (flatter) in deeper levels, especially against raw input space, as interpolating between far points yields higher density under deeper representations. (c) (d) Local Convex Ball - Log-density of samples generated by adding Gaussian noise to representation at different levels ($\sigma \in [0.01, 5]$, the x-axis): More volume is occupied by good samples on deeper layers. (e) (f) Mode Mixing Histograms - distribution (y-axis) of number of classes visited (x-axis) for different models. (e) with 10 samples. (f) with 20 samples for CAE, 100 samples with DBN. Deeper models mix much better.

better mixing observed, we conjecture the help of a better disentangling of the underlying factors of variation, and in particular of the class factors. This would mean that the manifolds associated with different classes are more unfolded (as assumed by **H3(a)**) and possibly that different hidden units specialize more to specific classes than they would on lower layers. Hence the unfolding (**H3(a)**) and disentangling (**H1**) hypotheses reconcile better discriminative ability with expanded volume of good samples (**H3(b)**).

8.5 Conclusion

The following hypotheses were tested: (1) deeper representations can yield better samples and better mixing between modes ; (2) this is due to better disentangling ; (3) this is associated with unfolding of the manifold where data concentrate along with expansion of the volume good samples take in representation-space. The experimental results were in agreement with these hypotheses. They showed better samples and better mixing on higher levels, better samples obtained when interpolating between examples at higher levels, and better samples obtained when adding isotropic noise at higher levels. We also considered the potential conflict between the third hypothesis and better discrimination (confirmed on the models tested) and explained it away as a consequence of the second hypothesis.

This could be immediate good news for applications requiring to generate MCMC samples: by transporting the problem to deeper representations, better and faster results could be obtained. Future work should also investigate the link between better mixing and the process of training deep learners themselves, when they depend on an MCMC to estimate the log-likelihood gradient. One interesting direction is to investigate the link between tempering and the better mixing chains obtained from deeper layers.

8.6 Acknowledgements

The authors thank A. Courville and P. Vincent for fruitful discussions, as well as NSERC, Canada Research Chairs, CIFAR, Compute Canada and by the French ANR Project ASAP ANR-09-EMER-001. Codes for the experiments have been implemented using the Theano (Bergstra et al., 2010a) Machine Learning libraries.

9 Préambule au Quatrième Article

Dans ce quatrième article, un espace sémantique est appris pour une tâche de langage à l'aide de réseaux de neurones récurrents. Cet espace sémantique est utilisé comme espace d'entrée pour le réseau récurrent et c'est le processus d'apprentissage lui-même qui va ajuster la structure de l'espace sémantique. Au terme de l'apprentissage, des mots aux sens similaires ou de même classe sont alors proches au sens d'une distance euclidienne dans cet espace sémantique d'entrée.

9.1 Détails de l'article

Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu and Geoffrey Zweig. *IEEE Transactions on Audio, Signal and Language Processing* 2015.

Contribution personnelle Cet article est issu de la recherche effectuée lors de deux stages à Microsoft Research à Redmond aux États-Unis en 2012 et 2013. Cette recherche a donné lieu à une première publication (Mesnil et al., 2013b) que j'ai présentée à Lyon. Après avoir rendu le code disponible[1] pour permettre aux chercheurs de reproduire les résultats de nos expériences, un tutoriel a été réalisé dans le cadre des Deep Learning Tutorials[2].

Ensuite, un travail de fond visant à rassembler tout le travail à l'état de l'art concernant les Réseaux de Neurones Récurrents sur cette tâche a été coordonné en collaboration avec Xiaodong He, chercheur à Microsoft. J'ai rédigé la première version de cette publication et réalisé l'intégralité des expériences excepté un résulat

1. https://github.com/mesnilgr/is13
2. http://deeplearning.net/tutorial/rnnslu.html

utilisant les dropout fourni par Yann Dauphin et les expériences de CRFs récurrents effectuées par Kaisheng Yao.

9.2 Contexte

Au commencement de ce travail de recherche, les CRFs sont toujours considérés comme l'état de l'art dans le domaine de la compréhension du langage parlé. Nous avons pu démontrer que les réseaux de neurones récurrents combinés à plusieurs techniques comme les embeddings de mots et les fenêtres de contexte permettaient d'améliorer les performances instaurant ainsi un nouvel état de l'art.

9.3 Contributions

Ces architectures sont actuellement utilisées en tant que produit chez Microsoft. Un brevet a découlé de ces travaux. Tous les détails concernant cette publication permettent de reproduire les résultats de nos travaux avec l'aide du tutoriel et du code mis à disposition.

9.4 Récents développements

Des architectures similaires à celles présentées dans cette publication ont récemment fait l'objet d'améliorations de performances significatives dans la Traduction Machine (Sutskever et al., 2014). Aussi, le tutoriel réalisé pour les Deep Learning Tutorial sera présenté lors d'un atelier sur Theano à San Francisco en janvier 2015.

10 Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding

10.1 Introduction

The term ”spoken language understanding” (SLU) refers to the targeted understanding of human speech directed at machines (Tur and Mori, 2011). The goal of such ”targeted” understanding is to convert the recognition of user input, S_i, into a task-specific semantic representation of the user's intention, U_i at each turn. The dialog manager then interprets U_i and decides on the most appropriate system action, A_i, exploiting semantic context, user specific meta-information, such as geo-location and personal preferences, and other contextual information.

The semantic parsing of input utterances in SLU typically consists of three tasks: domain detection, intent determination, and slot filling. Originating from call routing systems, the domain detection and intent determination tasks are typically treated as semantic utterance classification problems (Schapire and Singer, 2000; Haffner et al., 2003; Yaman et al., 2008; Sarikaya et al., 2011). Slot filling is typically treated as a sequence classification problem in which contiguous sequences of words are assigned semantic class labels. (Wang et al., 2005; Wang, Deng, and Acero, Wang et al.; Pieraccini et al., 1992; Wang and Acero, 2006; He and Young, 2003; Raymond and Riccardi, 2007; Viterbi, 1967; Zue and Glass, 2000).

In this paper, following the success of deep learning methods for semantic utterance classification such as domain detection (Sarikaya et al., 2011) and intent determination (Deng et al., 2012; He, 2012; Dauphin et al., 2013), we focus on applying deep learning methods for slot filling. Standard approaches to solving the slot filling problem include generative models, such as HMM/CFG composite models (Pieraccini et al., 1992; Wang et al., 2005; Macherey et al., 2001), hidden vector state (HVS) model (He and Young, 2003), and discriminative or conditional models such as conditional random fields (CRFs) (Lafferty et al., 2001; Wang, Deng, and Acero, Wang et al.; Wang and Acero, 2006; Raymond and Riccardi, 2007; Viterbi, 1967; Liu et al., 2012; Jeong and Lee, 2007) and support vector machines (SVMs) (Kudo and Matsumoto, 2001). Despite many years of research, the slot filling task in SLU is still a challenging problem, and this has motivated the recent application of a number of very successful continuous-space, neural net, and deep learning approaches, e.g. (Deng et al., 2012; Mesnil et al., 2013a; Yao et al., 2013; Sarikaya et al., 2011; Yao et al., 2014).

In light of the recent success of these methods, especially the success of RNNs in language modeling (Mikolov et al., 2011, 2013) and in some preliminary SLU experiments (Mesnil et al., 2013a; Yao et al., 2013; Sarikaya et al., 2011; Yao et al., 2014), in this paper we carry out an in-depth investigation of RNNs for the slot filling task of SLU. In this work, we implemented and compared several important RNN architectures, including the Elman-type networks (Elman, 1990), Jordan-type networks (Jorda, 1997) and their variations. To make the results easy to reproduce and rigorously comparable, we implemented these models using the common Theano neural network toolkit (Bergstra et al., 2010b) and evaluated them on the standard ATIS (Airline Travel Information Systems) benchmark. We also compared our results to a baseline using conditional random fields (CRF). Our results show that on the ATIS task, both Elman-type networks and Jordan-type networks outperform the CRF baseline substantially, and a bi-directional Jordan-type network that takes into account both past and future dependencies among slots works best.

In the next section, we formally define the semantic utterance classification problem along with the slot filling task and present the related work. In Section 10.3, we propose a brief review of deep learning for slot filling. Section 10.4 more specifically describes our approach of RNN architectures for slot filling. We describe sequence level optimization and decoding methods in Section 10.5. Experimental results are summarized and discussed in section 12.6.

10.2 Slot Filling in Spoken Language Understanding

A major task in spoken language understanding in goal-oriented human-machine conversational understanding systems is to automatically extract semantic concepts,

Sentence	show	flights	from	Boston	to	New	York	today
Slots/Concepts	O	O	O	B-dept	O	B-arr	I-arr	B-date
Named Entity	O	O	O	B-city	O	B-city	I-city	O
Intent	Find_Flight							
Domain	Airline Travel							

Table 10.1 – ATIS utterance example IOB representation

or to fill in a set of arguments or ”slots” embedded in a semantic frame, in order to achieve a goal in a human-machine dialogue.

An example sentence is provided in Table 10.1, with domain, intent, and slot/-concept annotations illustrated, along with typical domain-independent named entities. This example follows the popular in/out/begin (IOB) representation, where *Bostoi* and *New York* are the departure and arrival cities specified as the slot values in the user's utterance, respectively.

While the concept of using semantic frames (templates) is motivated by the case frames of the artificial intelligence area, the slots are very specific to the target domain and finding values of properties from automatically recognized spoken utterances may suffer from automatic speech recognition errors and poor modeling of natural language variability in expressing the same concept. For these reasons, spoken language understanding researchers employed statistical methods. These approaches include generative models such as hidden Markov models, discriminative classification methods such as CRFs, knowledge-based methods, and probabilistic context free grammars. A detailed survey of these earlier approaches can be found in (Wang, Deng, and Acero, Wang et al.).

For the slot filling task, the input is the sentence consisting of a sequence of words, L, and the output is a sequence of slot/concept IDs, S, one for each word. In the statistical SLU systems, the task is often formalized as a pattern recognition problem: Given the word sequence L, the goal of SLU is to find the semantic representation of the slot sequence S that has the maximum *a posteriori* probability $P(S|L)$.

In the generative model framework, the Bayes rule is applied:

$$\hat{S} = \arg\max_S P(S|L) = \arg\max_S P(L|S)P(S) \tag{10.1}$$

The objective function of a generative model is then to maximize the joint probability $P(L|S)P(S) = P(L, S)$ given a training sample of L, and its semantic annotation, S.

The first generative model, used by both the AT&T CHRONUS system (Pieraccini et al., 1992) and the BBN Hidden Understanding Model (HUM) (Miller et al., 1994), assumes a deterministic one-to-one correspondence between model states and the segments, i.e., there is only one segment per state, and the order of the segments follows that of the states.

As another extension, in the Hidden Vector State model the states in the Markov chain representation encode all the structure information about the tree using stacks, so the semantic tree structure (excluding words) can be reconstructed from the hidden vector state sequence. The model imposes a hard limit on the maximum depth of the stack, so the number of the states becomes finite, and the prior model becomes the Markov chain in an HMM (He and Young, 2003).

Recently, discriminative methods have become more popular. One of the most successful approaches for slot filling is the conditional random field (CRF) (Lafferty et al., 2001) and its variants. Given the input word sequence $L_1^N = l_1, \ldots, l_N$, the linear-chain CRF models the conditional probability of a concept/slot sequence $S_1^N = s_1, \ldots, s_N$ as follows:

$$P(S_1^N|L_1^N) = \frac{1}{Z}\prod_{t=1}^{N} \exp H(s_{t-1}, s_t, l_{t-d}^{t+d}) \tag{10.2}$$

where

$$H(s_{t-1}, s_t, l_{t-d}^{t+d}) = \sum_{m=1}^{M} \lambda_m h_m(s_{t-1}, s_t, l_{t-d}^{t+d}) \tag{10.3}$$

and $h_m(s_{t-1}, s_t, l_{t-d}^{t+d})$ are features extracted from the current and previous states s_t and s_{t-1}, plus a window of words around the current word l_t, with a window size of $2d + 1$.

CRFs have first been used for slot filling by (He and Young, 2003). CRF models have been shown to outperform conventional generative models. Other discriminative methods such as the semantic tuple classifier based on SVMs (Young, 2012) has the same main idea of semantic classification trees as used by the Chanel system (Kuhn and Mori, 1995), where local probability functions are used, i.e., each phrase is separately considered to be a slot given features. More formally,

$$P(S_1^N|L_1^N) = \prod_{t=1}^{N} P(s_t|s_1^{t-1}, L_1^N) \tag{10.4}$$

These methods treat the classification algorithm as a black box implementation of linear or log-linear approaches but require good feature engineering. As discussed in (Bengio, 2009b; Deng et al., 2012), one promising direction with deep learning architectures is integrating both feature design and classification into the learning procedure.

10.3 Deep Learning Review

In comparison to the above described techniques, deep learning uses many layers of neural networks (Bengio, 2009b). It has made strong impacts on applications ranging from automatic speech recognition (Dahl et al., 2012) to image recognition (Krizhevsky et al., 2012b).

A distinguishing feature of NLP applications of deep learning is that inputs are symbols from a large vocabulary, which led the initial work on neural language modeling (Bengio et al., 2000) to suggest map words to a learned distributed representation either in the input or output layers (or both), with those embeddings learned jointly with the task. Following this principle, a variety of neural net architectures and training approaches have been successfully applied (Huang et al., 2013; Deng et al., 2012; Collobert et al., 2011a; Mikolov et al., 2011, 2013; He, 2012; Mikolov and Zweig, 2012; Shen et al., 2014; Socher et al., 2012; Yih et al., 2014; Yu et al., 2013). Particularly, RNNs (Mikolov et al., 2011, 2013; Mikolov and Zweig, 2012) are also widely used in NLP. One can represent an input symbol as a one-hot vector, i.e., containing zeros except for one component equal to one, and this weight vector is considered as a low-dimensional continuous valued vector representation of the original input, called word embedding. Critically, in this vector space, similar words that have occurred syntactically and semantically tend to be placed by the learning procedure close to each other, and relationships between words are preserved. Thus, adjusting the model parameters to increase the objective function for a training example which involves a particular word tends to improve performances for similar words in similar context, thereby greatly improving generalization and addressing the curse-of-dimensionality obstacle faced with traditional n-gram non-parametric models (Bengio et al., 2000).

One way of building a deep model for slot filling is to stack several neural network layers on top of each other. This approach was taken in (Deoras and Sarikaya, 2013), which used deep belief networks (DBNs), and showed superior results to a CRF baseline on ATIS. The DBNs were built with a stack of Restricted Boltzmann Machines (RBMs) (Hinton et al., 2006a). The RBM layers were pre-trained to initialize the weights. Then the well-known back-propagation algorithm was used to fine-tune the weights of the deep network in a discriminative fashion. Once the individual local models are trained, Viterbi decoding is carried out to find the best slot sequence given the sequence of words.

In contrast to using DBNs, we propose recurrent neural networks (RNNs). The basic RNNs used in language modeling read an input word and predict the next word. For SLU, these models are modified to take a word and possibly other features as input, and to output a slot value for each word. We will describe RNNs in detail in the following section.

10.4 Recurrent Neural Networks for Slot-Filling

We provide here a description of the RNN models used for the slot filling task.

10.4.1 Words Embeddings

The main input to a RNN is a one-hot representation of the next input word. The first-layer weight matrix defines a vector of weights for each word, whose dimensionality is equal to the size of the hidden layer (Fig. 10.1) - typically a few hundred. This provides a continuous-space representation for each word. These neural word embeddings (Bengio et al., 2000) may be trained a-priori on external data such as the Wikipedia, with a variety of models ranging from shallow neural networks (Schwenk and Gauvain, 2005) to convolutional neural networks (Collobert et al., 2011a) and RNNs (Mikolov et al., 2011). Such word embeddings actually

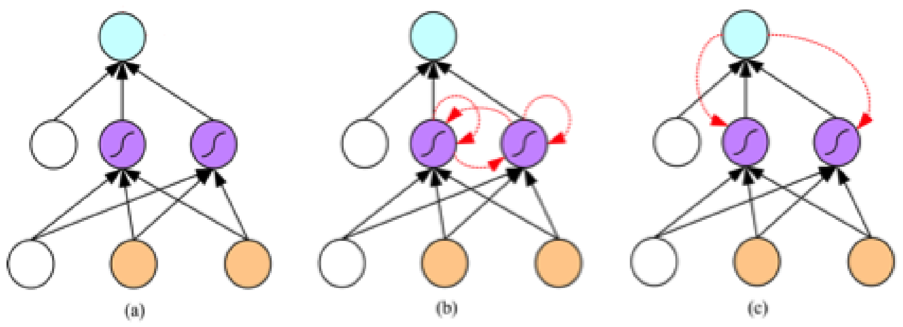


Figure 10.1 – Three different types of neural networks (a) Feed-forward NN ; (b) Elman-RNN ; (c) Jordan-RNN

present interesting properties (Mikolov et al., 2013) and tend to cluster (Collobert et al., 2011a) when their semantics are similar.

While (Mesnil et al., 2013a; Yao et al., 2013) suggest initializing the embedding vectors with unsupervised learned features and then fine-tune it on the task of interest, we found that directly learning the embedding vectors initialized from random values led to the same performance on the ATIS dataset, when using the SENNA[1] word embeddings. While this behavior seems very specific to ATIS, we considered extensive experiments about different unsupervised initialization techniques out of the scope of this paper. Word embeddings were initialized randomly in our experiments.

10.4.2 Context Word Window

Before considering any temporal feedback, one can start with a context word window as input for the model. It allows one to capture short-term temporal dependencies given the words surrounding the word of interest. Given d_e the dimension of the word embedding and $|V|$ the size of the vocabulary, we construct the d-context word window as the ordered concatenation of $2d+1$ word embedding vectors, i.e. d previous word followed by the word of interest and d next words, with the following dot product:

$$C_d(l_{i-d}^{i+d}) = \tilde{E}\tilde{l}_{i-d}^{i+d} \in \mathbb{R}^{d_e(2d+1)} \tag{10.5}$$

where $\tilde{E}$ corresponds to the embedding matrix $E \in \mathcal{M}_{d_e \times |V|}(\mathbb{R})$ replicated ver-

1. http://ml.nec-labs.com/senna/

$$l(t) = [\text{flights}, \textbf{from}, \text{Boston}\ ]$$
$$\text{"from"} \rightarrow l_{\text{from}} \in \mathbb{R}^{d_e}$$
$$l(t) \rightarrow C_3(t) = [l_{\text{flights}}, l_{\textbf{from}}, l_{\text{Boston}}]$$

Table 10.2 – Context window mapping example

tically $2d+1$ times and $\tilde{l}_{i-d}^{i+d} = [\tilde{l}_{i-d}, \ldots, \tilde{l}_i, \ldots, \tilde{l}_{i+d}]^T \in \mathbb{R}^{|V|(2d+1)}$ corresponds to the concatenation of one-hot word index vectors $\tilde{l}_i$.

$$\tilde{l}_i(\text{"flight"}) = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \text{The index of word "flight" in the vocabulary} \tag{10.6}$$

In this window approach, one might wonder how to build a d-context window for the first/last words of the sentence. We work around this border effect problem by padding the beginning and the end of sentences d times with a special token. In Table 10.2, we depict an example of building a context window of size 3 around the word "from":

In this example, $l(t)$ is a 3-word context window around the t-th word "from". l_{from} corresponds to the appropriate line in the embedding matrix E mapping the word "from" to its word embedding. Finally, $C_3(t)$ gives the ordered concatenated word embeddings vector for the sequence of words in $l(t)$.

10.4.3 Elman, Jordan and Hybrid architectures

As in (Mesnil et al., 2013a), we describe here the two most common RNN architectures in the literature: the Elman (Elman, 1990) and Jordan (Jorda, 1997) models. The architectures of these models are illustrated in Fig. 10.1.

In contrast with classic feed-forward neural networks, the Elman neural network keeps track of the previous hidden layer states through its recurrent connections. Hence, the hidden layer at time t can be viewed as a state summarizing past inputs along with the current input. Mathematically, Elman dynamics with d_h hidden nodes at each of the H hidden layers are depicted below:

$$h^{(1)}(t) = f(U^{(1)} C_d(l_{t-d}^{t+d}) + U'^{(1)} h^{(1)}(t-1)) \tag{10.7}$$

$$h^{(n+1)}(t) = f(U^{(n+1)} h^{(n)}(t) + U'^{(n+1)} h^{(n+1)}(t-1)) \tag{10.8}$$

where we used the non-linear sigmoid function applied element wise for the hidden layer $f(x) = 1/(1 + \exp -x)$ and $h^{(i)}(0) \in \mathbb{R}^{d_h}$ are parameter vectors to be learned. The superscript denotes the depth of the hidden layers and U' represents the recurrent weights connection. The posterior probabilities of the classifier for each class are then given by the softmax function applied to the hidden state:

$$P(y(t) = i | l_0^{t+d}) = \frac{\exp \sum_{j=1}^{d_h} V_{i,j} h_j^{(H)}(t)}{\sum_{i=1}^{N} \exp \sum_{j=1}^{d_h} V_{i,j} h_j^{(H)}(t)} \tag{10.9}$$

Where V correspond to the weights of the softmax top layer.

The learning part then consists of tuning the parameters $\theta = \{E, h^{(1)}(0), U^{(1)}, U'^{(1)}, \dots, h^{(H)}(0), U^{(H)}, U'^{(H)}, V\}$ of the RNN with N output classes. Precisely, the matrix shapes are $U^{(1)} \in \mathcal{M}_{d_h \times d_e(2d+1)}(\mathbb{R})$; $U'^{(1)}, \dots, U^{(H)}, U'^{(H)} \in \mathcal{M}_{d_h \times d_h}(\mathbb{R})$ and $V \in \mathcal{M}_{N \times d_h}(\mathbb{R})$. For training, we use stochastic gradient descent, with the parameters being updated after computing the gradient for each one of the sentences in our training set $\mathcal{D}$, towards minimizing the negative log-likelihood. Note that a sentence is considered as a tuple of words and a tuple of slots:

$$\mathcal{L}(\theta) = \sum_{(S,W) \in \mathcal{D}} \sum_{t=1}^{T} \log P_\theta(s_t | l_0^{t+d}) \tag{10.10}$$

Note that the length T of each sentence can vary among the training samples and the context word window size d is a hyper-parameter.

The Jordan RNN is similar to the Elman-type network except that the recurrent connections take their input from the output posterior probabilities:

$$h(t) = f(U C_d(l_{t-d}^{t+d}) + U' P(y(t-1))) \tag{10.11}$$

where $U' \in \mathcal{M}_{d_h \times N}(\mathbb{R})$ and $P(y(0)) \in \mathbb{R}^N$ are additional parameters to tune. As pointed out in (Mesnil et al., 2013a), three different options can be considered for the feedback connections: (a) $P(y(t-1))$, (b) a one-hot vector with an active bit for $\arg\max_i P_i(y(t-1))$ or even (c) the ground truth label for training. Empirically (Mesnil et al., 2013a), none of these options significantly outperformed all others.

In this work, we focused on the Elman-type, Jordan-type and hybrid versions of RNNs. The hybrid version corresponds to a combination of the recurrences from the Jordan and the Elman models:

$$h(t) = f(UC_d(l_{t-d}^{t+d}) + U^{'}P(y(t-1)) + U^{*}h(t-1)) \tag{10.12}$$

10.4.4 Forward, Backward and Bidirectionnal variants

In slot filling, useful information can be extracted from the future and we do not necessarily have to process the sequence online in a single forward pass. It is also possible to take into account future information with a single backward pass but still, this approach uses only partial information available. A more appealing model would consider both past and future information at the same time: it corresponds to the bi-directional Elman (Schuster and Paliwal, 1997; Graves et al., 2013) or Jordan (Mesnil et al., 2013a) RNN.

We describe the bidirectional variant only for the first layer since it is straightforward to build upper layers as we did previously for the Elman RNN. First, we define the forward $\overrightarrow{h}(t)$ and the backward $\overleftarrow{h}(t)$ hidden layers:

$$\overrightarrow{h}(t) = f(\overrightarrow{U}C_d(l_{t-d}^{t+d}) + \overrightarrow{U}'\overrightarrow{h}(t-1)) \tag{10.13}$$

$$\overleftarrow{h}(t) = f(\overleftarrow{U}C_d(l_{t-d}^{t+d}) + \overleftarrow{U}'\overleftarrow{h}(t-1)) \tag{10.14}$$

where $\overrightarrow{U}$ corresponds to the weights for the forward pass and $\overleftarrow{U}$ for the backward pass. The superscript $U^{'}$ corresponds to the recurrent weights. The bidirectional hidden layer $\overleftrightarrow{h}(t)$ then takes as input the forward and backward hidden layers:

$$\overleftrightarrow{h}(t) = f(BC_d(l_{t-d}^{t+d}) + B^{'}\overrightarrow{h}(t-1) + B^{*}\overleftarrow{h}(t+1)) \tag{10.15}$$

where B are the weights for the context window input, $B^{'}$ projects the forward pass hidden layer of the previous time step (past), and B^{*} the backward hidden layer of the next time step (future).

10.5 Sequence Level Optimization and Decoding

The previous architectures are optimized based on a tag-by-tag likelihood as opposed to a sequence-level objective function. In common with Maximum Entropy Markov Model (MEMM) (McCallum et al., 2000) models, the RNNs produce a sequence of locally-normalized output distributions, one for each word position. Thus, it can suffer from the same label bias (Lafferty et al., 2001) problem. To ameliorate these problems, we propose two methods: Viterbi decoding with slot language models and recurrent CRF.

10.5.1 Slot Langage Models

As just mentioned, one advantage of CRF models over RNN models is that it is performing global sequence optimization using tag level features. In order to approximate this behavior, and optimize the sentence level tag sequence, we explicitly applied the Viterbi (Viterbi, 1967) algorithm. To this end, a second order Markov model has been formed, using the slot tags, $s_i \in S$ as states, where the state transition probabilities, $P_{(LM)}(s_i|s_j)$ are obtained using a trigram tag language model (LM). The tag level posterior probabilities obtained from the RNN were used when computing the state observation likelihoods.

$$\begin{aligned}\hat{S} &= \arg\max_S P(S|L) \\ &= \arg\max_S P_{(LM)}(S)^\alpha \times P(L|S) \\ &= \arg\max_S P_{(LM)}(S)^\alpha \times \prod_t \frac{P_{\text{RecNN}}(s_t|l_t)}{P(s_t)}\end{aligned}$$

As is often done in the speech community, when combining probabilistic models of different types, it is advantageous to weight the contributions of the language and observation models differently. We do so by introducing a tunable model combination weight, α, whose value is optimized on held-out data. For computation, we used the SRILM toolkit[2].

2. http://www.speech.sri.com/projects/srilm/

10.5.2 Recurrent CRF

The second scheme uses the objective function of a CRF, and trains RNN parameters according to this objective function. In this scheme, the whole set of model parameters, including transition probabilities and RNN parameters, are jointly trained, taking advantage of the sequence-level discrimination ability of the CRF and the feature learning ability of the RNN. Because the second scheme is a CRF with features generated from an RNN, we call it a recurrent conditional random field (R-CRF) (Yao et al., 2014, 2013). The R-CRF differs from previous works that use CRFs with feed-forward neural networks (Peng et al., 2009; Yu et al., 2010) and convolutional neural networks (Xu and Sarikaya, 2013), in that the R-CRF uses RNNs for feature extraction – using RNNs is motivated by its strong performances on natural language processing tasks. The R-CRF also differs from works in sequence training of DNN/HMM hybrid systems (Vesely et al., 2013; Kingsbury et al., 2012; Su et al., 2013) for speech recognition, which use DNNs and HMMs, in that R-CRF uses the CRF objective and RNNs.

The R-CRF objective function is the same as Eq. 10.2 defined for the CRF, except that its features are from the RNN. That is, the features $h_m(s_{t-1}, s_t, l_0^{t+d})$ in the CRF objective function Eq. 10.3 now consist of transition feature $h_m(s_{t-1}, s_t)$ and tag-specific feature $h_m(s_t, l_{t-d}^{t+d})$ from the RNN. Note that since features are extracted from an RNN, they are sensitive to inputs back to time t=0. Eq. 10.3 is re-written as follows:

$$
\begin{aligned}
H(s_{t-1}, s_t, l_{t-d}^{t+d}) &= \sum_{m=1}^{M} \lambda_m h_m(s_{t-1}, s_t, l_0^{t+d}) \\
&= \sum_{p=1}^{P} \lambda_p h_p(s_{t-1}, s_t) + \sum_{q=1}^{Q} \lambda_q h_q(s_t, l_0^{t+d})
\end{aligned}
$$

In a CRF, $h_m(s_{t-1}, s_t, l_{t-d}^{t+d})$ is fixed and is usually a binary value of one or zero, so the only parameters to learn are the weights λ_m. In contrast, the R-CRF uses RNNs to output $h_m(s_t, l_0^{t+d})$, which itself can be tuned by exploiting error back-propagation to obtain gradients. To avoid the label-bias problem (Lafferty et al., 2001) that motivated CRFs, the R-CRF uses un-normalized scores from the activations before the softmax layer as features $h_m(s_t, l_0^{t+d})$. In the future, we would like to investigate using activations from other layers of RNNs.

The R-CRF has additional transition features to estimate. The transition features are actually the transition probabilities between tags. Therefore the size of this feature set is $O(N^2)$ with N the number of slots. The number of RNN parameters is $O(NH + H^2 + HV)$. Usually the relation among vocabulary size V, hidden layer size H and slot number N is $V >> H > N$. Therefore, the number of additional transition features is small in comparison.

10.6 Experimental Results

In this section we present our experimental results for the slot filling task using the proposed approaches.

10.6.1 Datasets

We used the ATIS corpus as used extensively by the SLU community, e.g. (Tur and Mori, 2011; Wang, Deng, and Acero, Wang et al.; Tur et al., 2011, 2010). The original training data include 4,978 utterances selected from Class A (context independent) training data in the ATIS-2 and ATIS-3 corpora. In this work, we randomly sampled 20% of the original training data as the held-out validation set, and used the left 80% data as the model training set. The test set contains 893 utterances from the ATIS-3 Nov93 and Dec94 datasets. This dataset has 128 unique tags, as created by (Raymond and Riccardi, 2007) from the original annotations. In our first set of experiments on several training methods and different directional architectures, we only used lexical features in the experiments. Then, in order to compare with other results, we incorporated additional features in the RNN architecture.

In our experiments, we preprocessed the data as in (Yao et al., 2013). Note that authors in (Deng et al., 2012; Mesnil et al., 2013a; Deoras and Sarikaya, 2013; Tur et al., 2011, 2010) used a different preprocessing technique, and hence their results are not directly comparable. However, the best numbers reported on ATIS by (Deoras and Sarikaya, 2013) are 95.3% F1-score on manual transcriptions with DBNs, using word and named entity features (in comparison to their CRF baseline of 94.4%).

As additional sets of experiments, we report results on two other custom datasets focusing on movies (He, 2012) and entertainment. Each word has been manually assigned a slot using the IOB schema as described earlier.

10.6.2 Baseline and Models

On these datasets, Conditional Random Fields (CRF) are commonly used as a baseline (Wang, Deng, and Acero, Wang et al.). The input of the CRF corresponds to a binary encoding of N-grams inside a context window. For all datasets, we carefully tuned the regularization parameters of the CRF and the size of the context window using 5-fold cross-validation. Meanwhile, we also trained a feed-forward network (FFN) for slot filling, with the architecture shown in Fig. 10.1(a). The size of the context window for FFN is tuned using 5-fold cross-validation.

10.6.3 RNN versus Baselines and Stochastic training versus Sentence mini-batch updates

Different ways of training the models were tested. In our experiments, the stochastic version considered a single (word, label) couple at a time for each update while the sentence mini-batch processed the whole sentence before updating the parameters. Due to modern computing architectures, performing updates after each example considerably increases training time. A way to process many examples in a shorter amount of time and exploit inherent parallelism and cache mechanisms of modern computers relies on updating parameters after examining a whole mini-batch of sentences.

First, we ran 200 experiments with random sampling (Bergstra and Bengio, 2012) of the hyper-parameters. The sampling choices for each hyper-parameter were for the depth, $H \in \{1, 2\}$, the context size, $d \in \{3, 5, \ldots, 17\}$, the embedding dimension, $d_e \in \{50, 100\}$ and 3 different random seed values. The learning rate was sampled from a uniform distribution in the range $[0.05, 0.1]$. The embedding matrix and the weight matrices were initialized from the uniform distribution in the range $[-1, 1]$. We performed early-stopping over 100 epochs, keeping the parameters

F1-score %	Elman	Jordan	Hybrid
RNN	94.98	94.29	95.06
FFN	93.32		
CRF	92.94		

Table 10.3 – Test set F1-score of the different models after 200 runs of random sampling of the hyper-parameters. All models are trained using the stochastic gradient approach.

that gave the best performance on the held-out validation set measured after each training epoch (pass on the training set).

The F1-measure on the test set of each method was computed after the hyper-parameter search. Results are reported in Table 10.3. All the RNN variants and the FFN model outperform the CRF baseline. And all the RNN variants outperform the FFN model, too.

Then, given the best hyper-parameters found previously on the validation set, we report the average, minimum, maximum and variance of the test set accuracy over 50 additional runs by varying only the random seed. In our case, the random initialization seed impacted the way we initialized the parameters and how we shuffled the samples at each epoch. Note that for the Hybrid RNN and stochastic updates, the score obtained during hyper-parameters search corresponds to the max of the validation set score over different random seeds. The results are presented in Table 10.4. The observed variances from the mean are in the range of 0.3%, which is consistent with the 0.6% reported in (Yao et al., 2013) with the 95% significance level based on the binomial test. We also observe that stochastic (STO) performs better than sentence mini-batches (MB) on average. In a large-scale setting, it is always more beneficial to perform sentence mini-batches as it reduces the training complexity. On our small ATIS benchmark, it took about the same number of epochs for convergence for both training schemes STO and MB, but each epoch took longer with STO.

10.6.4 Local Context Window and Bi-Directional Models

The slot-filling task is an off-line task, i.e., we have access to the whole sentence at prediction time. It should be beneficial to take advantage of all future and past available information at any time step. One way to do it consists of using bidirectional models to encode the future and past information in the input. The

F1-score %		Elman	Jordan	Hybrid
STO	Min	93.23	92.91	94.19
	Max	95.04	94.31	95.06
	Avg	94.44 ± 0.41	93.81 ± 0.32	94.61 ± 0.18
MB	Min	92.80	93.17	93.06
	Max	94.42	94.15	94.21
	Avg	93.58 ± 0.30	93.72 ± 0.24	93.66 ± 0.30

Table 10.4 – Measurement of the impact of using different ways of training the models and random seed on the performance.

bidirectional approach relies on the capacity of the network to summarize the past and future history through its hidden state. Here, we compare the bidirectional approach with the local context window where the future and past information is fed as input to the model. Therefore, rather than considering a single word here, the context window allows us to encode the future and past information in the input.

We ran a set of experiments for different architectures with different context-window sizes and no local context window and compare the results to a CRF using either unigram or N-grams. Results are summarized in Table 10.5. Note that the CRF using no context window (e.g., using unigram features only) performs significantly worse than the CRF using a context window (e.g., using up to 9-gram features).

The absence of a context window affects the performance of the Elman RNN (−1.83%), and it considerably damages the accuracy of the Jordan RNN (−29.00%). We believe this is because the output layer is much more constrained than the hidden layer, thus making less information available through recurrence. The softmax layer defines a probability and all its components sum to 1. The components are tied together, limiting their degree of freedom. In a classic hidden layer, none of the component is tied to the others, giving the Elman hidden layer a bit more power of expression than the Jordan softmax layer. A context window provides further improvements, while the bidirectional architecture does not benefit any of the models.

F1-score	Elman	Jordan	Hybrid	CRF
Single, w/o context	93.15	65.23	93.32	69.68
BiDir, w/o context	93.46	90.31	93.16	
Single, context	94.98(9)	94.29(9)	95.06(7)	92.94(9)
Bidir, context	94.73(5)	94.03(9)	94.15(7)	

Table 10.5 – F1-score of single and Bi-Directional models with or w/o context windows. We report the best context window size hyper-parameter as the number in the round brackets.

F1-score	Elman	Jordan	Hybrid	CRF
Word	94.98	94.29	95.06	92.94
Word+NE	96.24	95.25	95.85	95.16

Table 10.6 – Performance with Named Entity features.

10.6.5 Incorporating Additional Features

Most of the time, additional information such as look-up tables or clustering of words into categories is available. At some point, in order to obtain the best performance, we want to integrate this information in the RNN architecture. At the model level, we concatenated the Named Entity (NE) information feature as a one-hot vector feeding both to the context window input and the softmax layer (Mikolov and Zweig, 2012).

For the ATIS dataset, we used the gazetteers of flight related entities, such as airline or airport names as named entities. In Table 10.6, we can observe that it yields significant performance gains for all methods, RNN and CRF included.

10.6.6 ASR setting

In order to show the robustness of the RNN approaches, we have also performed experiments using the automatic speech recognition (ASR) outputs of the test set. The input for SLU is the recognition hypothesis from a generic dictation ASR system and has a word error rate (WER) of 13.8%. While this is significantly higher than the best reported performances of about 5% WER (Yaman et al., 2008), this provides a more challenging and realistic framework. Note that the model trained with manual transcriptions is kept the same.

Table 10.7 presents these results. As seen, the performance drops significantly for all cases, though RNN models continue to outperform the CRF baseline. We

F1-score	Elman	Jordan	Hybrid	CRF
Word	94.98	94.29	95.06	92.94
ASR	85.05	85.02	84.76	81.15

Table 10.7 – Comparison between manually labeled word and ASR output.

F1-score	Elman	Jordan	Hybrid
ATIS Word	94.98	94.29	95.06
ATIS Word+Viterbi	94.99(+0.01)	94.25(−0.04)	94.77(−0.29)
ATIS Word/CRF	92.94		
ATIS ASR	85.05	85.02	84.76
ATIS ASR+Viterbi	86.16(+1.11)	85.21(+0.19)	85.36(+0.6)
ATIS ASR/CRF	81.15		
Entertainment	88.67	88.70	89.04
Entertainment+Viterbi	90.19(+1.42)	90.62(+1.92)	90.01(+0.97)
Entertainment+Viterbi+Dropout	-	91.14(+2.44)	-
Entertainment /CRF	90.64		

Table 10.8 – Comparison with Viterbi decoding with different methods on several datasets

also notice that under the ASR condition, all three types of RNN perform similar to each other.

10.6.7 Entertainment dataset

As an additional experiment, we ran our best models on a custom dataset from the entertainment domain. Table 10.8 shows these results. For this dataset, the CRF outperformed RNN approaches. There are two reasons for this:

— The ATIS and Entertainment datasets are semantically very different. While the main task in ATIS is disambiguating between a departure and an arrival city/date, for the entertainment domain, the main challenge is detecting longer phrases such as movie names.

— While RNNs are powerful, the tag classification is still local, and the overall sentence tag sequence is not optimized directly as with CRFs.

However, as we shall cover in the next sections, the performance of the RNN approach can be improved using three techniques: Viterbi decoding, Dropout regularization, and fusion with the CRF framework.

10.6.8 Slot Language Models and Decoding

Using the Viterbi algorithm with the output probabilities of the RNN boosts the performance of the network in the Entertainment domain, while on ATIS, the improvement is much less significant. This shows the importance of modeling the slot dependencies explicitly and demonstrates the power of dynamic programing.

10.6.9 Dropout regularization

While deep networks have more capacity to represent functions than CRFs, they might suffer from overfitting. Dropout (Krizhevsky et al., 2012b) is a powerful way to regularize deep neural networks. It is implemented by randomly setting some of the hidden units to zero with probability p during training, then dividing the parameters by $1/p$ during testing. In fact, this is an efficient and approximate way of training an exponential number of networks that share parameters and then averaging their answer, much like an ensemble. We have found it further improves the performance on the Entertainment dataset, and beats the CRF by 0.5% as seen in Table 10.8 (i.e., 91.14% vs. 90.64%).

10.6.10 R-CRF results

We now compare the RNN and R-CRF models on the ATIS, Movies and Entertainment datasets. For this comparison, we have implemented the models with C code rather than Theano. On the ATIS data, the training features include word and named-entity information as described in (Tur et al., 2011), which aligns to the "Word+NE" line in Table 10.6. Note that performances between RNNs in Theano and C implementations are slightly different on ATIS. The C implementation of RNNs obtained 96.29% F1 score and Theano obtained 96.24% F1 score. We used a context window of 3 for bag-of-word feature (Yao et al., 2013). In this experiment, the RNN and R-CRF both are of the Elman type and use a 100-dimension hidden layer. On the Movies data, there are four types of features. The n-gram features are unigrams and bi-grams appeared in the training data. The regular expression features are those tokens, such as zip code and addresses, that can be defined in regular expressions. The dictionary features include domain-general knowledge sources such as US cities and domain-specific knowledge sources such as hotel names, restaurant

F1-score	CRF	RNN	R-CRF
ATIS Word+NE	95.16	96.29	96.46
Movies	75.50	78.20	82.21
Entertainment	90.64	88.11	88.50

Table 10.9 – Comparison with R-CRF and RNN on ATIS, Movies, and Entertainment datasets.

names, etc. The context-free-grammar features are those tokens that are hard to be defined in a regular expression but have context free generation rules such as time and date. Both RNNs and CRFs are optimal for the respective systems on the ATIS and Movies domains. On the Entertainment dataset, both RNN and R-CRF used 400 hidden layer dimension and momentum of 0.6. Features include a context window of 3 as a bag-of-words. The learning rate for RNNs is 0.1 and for R-CRFs it is 0.001.

As shown in Table 10.9, the RNNs outperform CRFs on ATIS and Movies datasets. Using the R-CRF produces an improved F1 score on ATIS. The improvement is particularly significant on Movies data, because of the strong dependencies between labels. For instance, a movie name has many words and each of them has to have the same label of "movie_name". Therefore, it is beneficial to incorporate dependencies between labels, and train at the sequence level. On the Entertainment dataset, the RNN and R-CRF did not perform as well as the CRF. However, results confirm that the R-CRF improves over a basic RNN.

10.7 Conclusions

We have proposed the use of recurrent neural networks for the SLU slot filling task, and performed a careful comparison of the standard RNN architectures, as well as hybrid, bi-directional, and CRF extensions. Similar to the previous work on application of deep learning methods for intent determination and domain detection, we find that these models have competitive performances and have improved performances over the use of CRF models. The new models set a new state-of-the-art in this area. Investigation of deep learning techniques for more complex SLU tasks, for example ones that involve hierarchical semantic frames, is part of future work.

11 Préambule au Cinquième Article

L'article précédent utilisait un espace sémantique en entrée d'un réseau de neurones. Dans cette dernière contribution, nous utilisons le réseau de neurones pour effectuer une projection dans un espace sémantique de sortie. C'est ensuite la similarité au sens d'un produit scalaire dans cet espace sémantique de sortie qui détermine la décision du classifieur. Nous présentons aussi comment apprendre un espace sémantique commun à plusieurs domaines, dans notre cas des mots et des images.

11.1 Détails de l'article

Learning Semantic Representations of Objects and their Parts Grégoire Mesnil, Antoine Bordes, Jason Weston, Gal Chechik and Yoshua Bengio, *Machine Learning Journal: Special Issue on Learning Semantics*, 2013

Contribution personnelle Le projet a débuté suite à une visite d'Antoine Bordes dans les bureaux de Google. Il m'a ensuite proposé de prendre part au projet. L'idée originale est d'Antoine, Jason Weston s'est ensuite chargé de nous fournir des représentations d'images utilisées en industrie par Google à cette époque. J'ai réalisé l'intégralité des expériences et participé à la rédaction avec la collaboration des coauteurs. La partie expérience s'est révélée être très intéressante car elle consistait à effectuer une tâche d'apprentissage avec un très grand nombre de données, plusieurs millions de couples. Nous remercions les reviewers anonymes du Machine Learning Journal qui ont aussi contribué à la qualité de l'article au travers de leurs questionnements.

11.2 Contexte

Au début de ces recherches, l'idée de partager des espaces sémantiques entre plusieurs domaines est assez neuve (Weston et al., 2011). Cependant, l'idée d'utiliser des embeddings pour modéliser le langage a déjà fait son chemin (Bengio et al., 2003), ces méthodes obtiennent même d'excellentes performances sur diverses tâches de traitement du langage naturel (Collobert et al., 2011b). L'originalité de notre travail est d'effectuer un transfert d'apprentissage au travers d'un espace sémantique commun à plusieurs domaines.

11.3 Contributions

J'ai eu la chance de pouvoir présenter cet article oralement à l'UTC de Compiègne devant l'équipe recherche ainsi qu'au GDR Information Signal Image Vision à Paris devant 70 personnes. Nous espérons que l'idée d'utiliser des espaces sémantiques pour effectuer du transfert d'apprentissage viendra enrichir le paysage de la recherche actuelle.

11.4 Récents développements

Sur Google Images, on peut observer de la recherche augmentée dans le système en production. Étant donné un mot-clé initial, le système va suggérer d'autres mots-clés de recherche sémantiquement reliés au mot-clé initial. Par exemple, une recherche d'image pour le mot-clé ”voiture” va ensuite suggérer d'autres mots-clés comme ”voiture sport”, ”f1”, ”voiture jaune”. Ces prémisses de recherche augmentée restent limitées au niveau du langage et n'affectent pas l'ensemble des images retournées par le système. Dans notre travail, nous proposons d'avoir un espace sémantique commun au langage et aux images qui permette de suggérer des images en plus des mots-clés.

Parallèlement, le système Devise (Bengio, 2013) à base d'espace sémantique est actuellement utilisé par Google pour son moteur de recherche d'images.

12 Learning Semantic Representations of Objects and their Parts

12.1 Introduction

Images and language are two complementary representations of information, and learning to translate between the two is of great interest. In one direction, the task of image annotation maps images to words, and in the other direction, the task of image retrieval maps words to images. Both the semantics of words and the semantics of images play a key role in these two tasks, since an accurate mapping is required to retrieve images and text that are semantically similar. At the same time, there also exist complex semantic relations within each modality that are important to model. For example, between-objects relations include the relation *X is a part of Y* and *X is an instance of Y*, and similar relations exist in the semantics of text terms. We wish to develop models that learn these types of semantic relations across and within modalities simultaneously.

Automatic tools provided by machine learning can be designed to capture the semantics described above. However, in real applications both the dictionary of possible words and the set of possible objects are large and learning their semantics requires a large amount of training data. Indeed, the performance of machine learning models highly depends on the quality and size of their training data sets, so there is a clear incentive to design methods able to handle the huge resources now available.

In this work, we develop a machine learning method to learn the semantics of words, objects and parts of objects that is efficient enough to be trained on large scale datasets. The method works by learning a latent semantic representation for each possible word or phrase, and each object and part. Each semantic concept has a vectorial representation in a low dimensional embedding space, the dimensions of which are learnt from data. In the low dimensional semantic embedding space we want to capture similarities between words, objects and part of objects, e.g. so that objects are related to their particular labels and parts in the space. To do this we employ a loss function that optimizes the ranking of words given an object, the loss function tries to get the correct assignment near the top of the ranked list.

The task we consider is of automatically labeling images with the objects in the scene as well as the parts of those objects. Importantly, we are interested in object parts that can be viewed as objects by themselves, like the flash bulb of a camera, or a wheel of a car. This is different from the numerous unnamed parts of which every object consists of. Identifying such "parts that are objects", can be particularly useful in image retrieval and browsing, where a large amount of such precisely labeled data could potentially lead to a compelling advance. For example, this could allow to automatically extend image queries via semantic part-based bridges: when looking for images of a particular object (say a "car"), one could also be presented images of related object parts (such as "wheel" or "windshield"). This would seamlessly enhance the browsing experience. Unfortunately, although some limited datasets with this type of labeling are available (Yao et al., 2007; Russell et al., 2008), collecting such deeply annotated data is costly and time consuming, Moreover, although some image annotation tasks can benefit from collecting indirect data, like the case of users clicking on images in search engines or accompanying text surrounding images, there are not any large scale applications that provide such evidence for object parts in images that we are aware of.

In this work we hence also propose a training method to tackle this problem of labeling objects and their parts in images without requiring any precisely labeled data. Our approach relies on a proxy supervision which bypasses the problem of precise annotation by using part-based semantic information among labels. Our model is composed of two ranking components: one for ranking labels according to an image and one for ranking labels according to another label. The first component can be seen as a standard image annotation model whereas the second one learns to give high ranks to pairs of labels for which one is the part of the other. The two components are trained jointly using combined data built from two sources (ImageNet and WordNet).

This paper builds upon previous work published by (Weston et al., 2011). However, the previous work has only focused on the standard image annotation setting, whereas the present version proposes jointly learning object and part representations. Hence, many new elements are provided including the word, object and part joint model, its training scheme, the proxy supervision setting, the ImageNet+WordNet dataset and all experimental results on it and on the LHI data set (Yao et al., 2007).

The paper is organized as follows. Section 12.2 describes learning joint latent semantic models for words and objects. Section 12.3 presents our full image annotation model for learning latent semantic models over words, objects and their parts. Section 12.4 then describes how to train both types of model. In particular, the form of the loss for supervised learning is first described, and then we introduce our proxy supervision setup for training objects and their parts when supervision is limited, and describe our corresponding custom data set. Section 12.5 describes related work on image annotation and part-based approaches. An empirical evaluation on our custom test set and on precisely labeled images from the LHI data set is given in Section 12.6. Finally, Section 12.7 concludes.

12.2 Latent Semantic Word and Object Models

Before we describe the model that learns the semantics of words, objects and their parts we start with a simpler model of representing only words and objects, previously described in (Weston et al., 2011). In this case, a large amount of supervised data can be obtained, and we hence detail a supervised training criterion for learning the appropriate ranking function. We will explain below how this setup is extended to learning about objects and their parts.

The following model learns a single latent semantic feature representation where both objects in images and word annotations are represented. The mapping functions for the two views are different, but are learnt jointly to optimize the supervised loss of interest for our final task (here, we concentrate on the task of annotating images). The method is described pictorially in Figure 12.1.

Notation summary

— $\mathcal{L}$ is a set depicted by the K words of the dictionary corresponding to the image annotations.

— $\mathcal{I}$ is the raw pixel space of images (no constraints on the size of the images).

— $\Phi_{\mathcal{I}} : \mathcal{I} \rightarrow \mathbb{R}^N$ maps an image to its sparse representation.

— $\Phi_{\mathcal{L}} : \mathcal{L} \rightarrow \mathbb{N}$ maps an annotation to its index in the dictionary.

— $E_{\mathcal{I}} : \mathcal{I} \rightarrow \mathbb{R}^D$ *embeds* an image to the semantic space.

— $E_{\mathcal{L}} : \mathcal{L} \rightarrow \mathbb{R}^D$ *embeds* an annotation to the same semantic space. In some cases, the semantic space for annotations is different than the images semantic space (see Section 12.6.2 about Unshared models).

— $f_I : \mathcal{I} \times \mathcal{L} \rightarrow \mathbb{R}$ returns a score defining the similarity between an annotation and an image.

Given an image I_o containing an object we wish to learn a function $E_{\mathcal{I}} : \mathcal{I} \rightarrow \mathbb{R}^D$ that *embeds* the inferred object I_o into a low-dimensional semantic space (where D is typically 50-100 dimensions and $E_{\mathcal{I}}(I_o)$ is the representation of I_o in the semantic space). Simultaneously, given an annotation of an object L_o, we wish to also learn a function $E_{\mathcal{L}} : \mathcal{L} \rightarrow \mathbb{R}^D$ that represents the annotation in the same semantic space. Then, our overall model takes the form:

$$f_I(I_o, L_o) = S(E_{\mathcal{I}}(I_o), E_{\mathcal{L}}(L_o)),$$

where $S : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ is a measure of similarity in the semantic embedding space *e.g.* a dot product $S(x, y) = x^\top y$.

For the feature representation of images, we first employ a fixed mapping $\Phi_{\mathcal{I}}(\cdot)$ that transforms the pixel representation into an N-dimensional vector which in this paper is a high-dimensional and sparse feature representation, as is also commonly performed in other works. Then, we transform this intermediate representation to the D-dimensional semantic space, via a linear map using a $D \times N$ matrix V of parameters:

$$E_{\mathcal{I}}(I_o) = V \Phi_{\mathcal{I}}(I_o).$$

In our model there is a dictionary with K possible labels that can be embedded using $E_{\mathcal{L}}(\cdot)$. Following other works dealing with embedding representations for text (Bengio, 2008; Weston, Bengio, and Usunier, Weston et al.) for each label we learn a D dimensional vector that will represent the label, resulting in a $D \times K$ dimensional matrix W of parameters to learn for the K labels:

$$E_{\mathcal{L}}(L_o) = W_{\Phi_{\mathcal{L}}(L_o)},$$

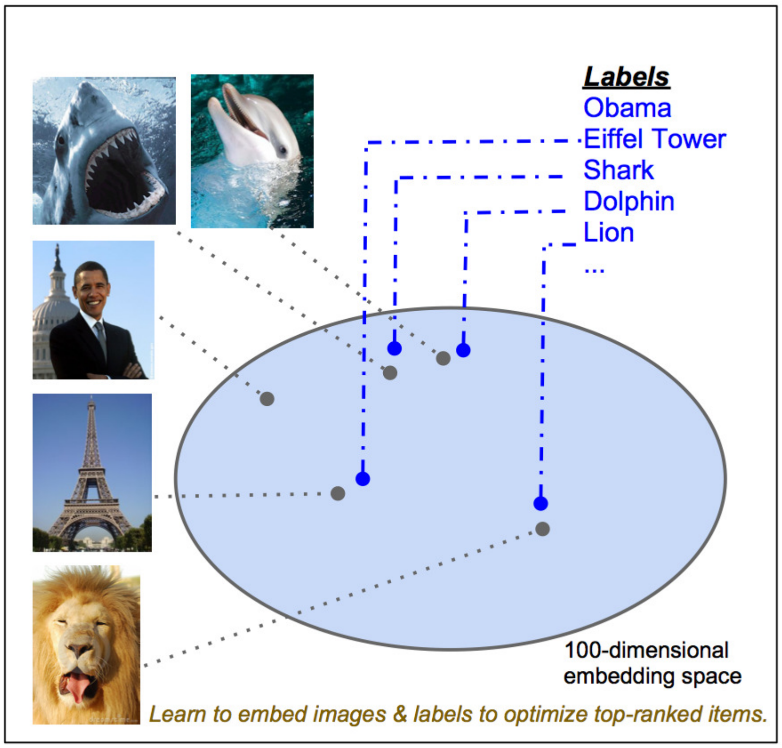

Labels
Obama
Eiffel Tower
Shark
Dolphin
Lion
...
100-dimensional embedding space
Learn to embed images & labels to optimize top-ranked items.

Figure 12.1 – Learning Latent Semantic Word and Object Models.

where $\Phi_{\mathcal{L}}(\cdot)$ maps from the particular label to its index in the dictionary, and thus retrieves the relevant column of a $D \times K$ matrix W. (Note that a standard matrix product with a one-hot vector would perform the same operation.)

Finally, for S we choose the dot product similarity in the semantic space, resulting in the final model:

$$f_I(I_o, L_o) = (V\Phi_{\mathcal{I}}(I_o))^\top W_{\Phi_{\mathcal{L}}(L_o)},$$

Our goal is to rank the candidate annotations of a given image such that the highest ranked annotations describe best the semantic content of the image. We will describe the training procedure in Section 12.4.

12.3 Latent Semantic Word, Object and Part Models

We want our model to simultaneously learn about objects being parts of scenes (a mapping from images to object labels) and about parts of objects belonging to the objects (and again, the labels of those parts). We hence now describe the generalization of the word, object model from the previous section to this case.

Notation summary

— $f_L : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}$ returns a score defining the similarity between an object annotation and a part annotation.

— $f_q : \mathcal{I} \times \mathcal{L} \times \mathcal{I} \times \mathcal{L} \rightarrow \mathbb{R}$ returns a score given a quadruplet of images and annotations of a part and an object.

— $f_I^U : \mathcal{I} \times \mathcal{L} \rightarrow \mathbb{R}$ differs from f_I since it has its own set of parameters and these parameters correspond to the annotation embeddings that are not shared between the score functions f_I^U and f_L. In this case, the annotations semantic space is different than the images semantic space.

Our full model takes the form:

$$f_q(I_o, L_o, I_p, L_p) = f_I(I_o, L_o) + f_I(I_p, L_p) + f_L(L_o, L_p)$$

where I_o is the image in which an object of interest is located, I_p is the image in which an object part of interest is located, L_o is the label of the object and L_p is the label of the part. The function $f_q(I_o, L_o, I_p, L_p)$ which scores a given quadruplet of two images and two labels is decomposed into three functions. The function $f_I(I_o, L_o)$ scores a given object label with the image of an object, and $f_I(I_p, L_p)$ does the same for a part label with the image of a part. The last function f_L scores the match between an object label and a part label. (Note, we do not have a direct $f_{II}(I_o, I_p)$ term in our model to capture image-image relationships directly although that is also possible.)

This model can be used in several setups. Given the image of an object I_o and a subregion of that image denoted with I_p, with no other prior knowledge, we can label the object in I_o and the part in I_p with:

$$\arg\max_{L'_o, L'_p} f_q(I_o, L'_o, I_p, L'_p).$$

If the object label is already known and we are just looking for the names of the parts we could fix L_o as well:

$$\arg\max_{L'_p} f_q(I_o, L_o, I_p, L'_p).$$

Finally, if we are only interested in labeling objects (and not their parts) we can use:

$$\arg\max_{L'_o} f_I(I_o, L'_o).$$

The experiments below evaluate these different setups. It now remains to define the particular makeup of the functions f_I and f_L.

As before, we assume that we are given a dictionary of K possible labels but now these labels are for both objects and parts, all in the same dictionary. For example, a house has a door, and in turn, the door has a handle, and so on. Again, for each label we will learn a D dimensional vector that will represent it in the semantic space, resulting in a $D \times K$ dimensional matrix W of parameters to learn for the K labels. The similarity between two labels is then defined with:

$$f_L(L_o, L_p) = W^\top_{\Phi_\mathcal{L}(L_o)} W_{\Phi_\mathcal{L}(L_p)},$$

where W_i indexes the i^{th} column (label) of W.

In the function f_I we deal with images of parts and objects. To measure similarity between an image and a label, we then have to transform them into the same space. This is again achieved with a fixed mapping $\Phi_\mathcal{I}(\cdot)$ and another $D \times N$ matrix V of parameters:

$$f_I(I, L) = (V\Phi_\mathcal{I}(I))^\top W_{\Phi_\mathcal{L}(L)},$$

Note also that the label embeddings W are also shared between all functions f_L and f_I. In our experiments we also consider a "non-sharing" setting where we decouple some of these parameters, so we instead consider:

$$f_I^U(I, L) = (V\Phi_{\mathcal{I}}(I))^\top U_{\Phi_{\mathcal{L}}(L)},$$

where U is now a different set of parameters to W.

12.4 Training Our Models

The two following sections (12.4.1, 12.4.2) are taken from previous work (Weston et al., 2011) and describe the original object-label setting and the WARP loss method. Afterwards, the object-parts model is built upon that and described in detail in Section 12.4.3.

12.4.1 Ranking Loss Function

We first consider the task of ranking labels $i \in \mathcal{Y}$ given an image x, i.e. the image annotation problem. In our setting labeled pairs (x, y) will be provided for training where only a single annotation $y_i \in \mathcal{Y}$ is labeled correct [1]. Let $f(x) \in \mathbb{R}^K$ be a vector function providing a score for each of the labels, where $f_i(x)$ is the value for label i.

A classical loss function for learning to rank is to maximize AUC by minimizing:

$$err_{AUC}(f(x), y) = \sum_x \sum_y \sum_{\bar{y} \neq y} \max(0, 1 + f_{\bar{y}}(x) - f_y(x)),$$

see, e.g. (Herbrich et al., 2000). This tries to make the positive label y ranked above negative labels $\bar{y}$, because it sums over all negative labels it optimizes the mean rank. It also enforces a margin of 1 as in margin-based methods like SVMs (Boser et al., 1992). To make training of such a loss function scalable to large datasets with our model, one can optimize this loss using stochastic gradient descent (SGD): sample triplets $(x, y, \bar{y})$ to make a gradient step on the hinge loss.

However, there is an issue with the loss above that, because all pairwise errors are the same (because it optimizes the mean rank), it may not be the best loss function for getting the correct label at the top of the ranked list (e.g. within

1. However, the methods described in this paper could be generalized to the multi-label case, naively by averaging the loss over all positive labels.

the top k). To give a simple example, suppose we are given only two functions to choose from, during our learning step we wish to pick the best one of the two. Given two training images, if function 1 ranks their true labels at position 1 and position 100 respectively, and function 2 ranks both at position 50, then the AUC loss function prefers these functions equally as they both have 100 "constraint violations", assuming the margin is the same. However, function 1 gives a superior precision at 1, because at least it gets one of the two examples correct.

To fix this problem, a class of ranking error functions was recently defined in (Usunier et al., 2009a) as:

$$err(f(x), y) = L(rank_y(f(x))) \tag{12.1}$$

where $rank_y(f(x))$ is the rank of the true label y given by $f(x)$:

$$rank_y(f(x)) = \sum_{i \neq y} I(f_i(x) \geq f_y(x))$$

where I is the indicator function, and $L(\cdot)$ transforms this rank into a loss:

$$L(k) = \sum_{j=1}^{k} \alpha_j, \text{ with } \alpha_1 \geq \alpha_2 \geq \cdots \geq 0. \tag{12.2}$$

This class of functions allows one to define different choices of $L(\cdot)$ with different minimizers. Minimizing L with $\alpha_j = \frac{1}{K-1}$ would optimize the mean rank, $\alpha_1 = 1$ and $\alpha_{j>1} = 0$ the proportion of top-ranked correct labels, and larger values of α in the first few positions optimize the top k in the ranked list, which is of interest for optimizing precision at k. Now, given our example of two functions from before where function 1 ranks their true labels at position 1 and position 100 respectively, and function 2 ranks both at position 50, then a choice of $\alpha_j = \frac{1}{K-1}$ prefers these functions equally (just like AUC), whereas a choice of $\alpha_j = 1/j$ prefers the first function, which gives superior precision at 1.

To optimize (12.1) for large scale data one can also use stochastic gradient descent by making updates of the parameters β over randomly sampled examples and negative labels of the form:

$$\beta_{t+1} = \beta_t - \gamma_t \frac{\partial \overline{err}(f(x), y, \bar{y})}{\partial \beta_t}. \tag{12.3}$$

where γ_t is the learning rate and

$$\overline{err}(f(x), y, \bar{y}) = L(rank_y(f(x))) \max(1 - f_{\bar{y}}(x) + f_y(x), 0).$$

Note the difference to the AUC optimization is just that the weighting of the update is now dependent on the (current) rank of y.

To perform this SGD step we still have the problem that computing $rank^1_y(f(x))$ is quite inefficient: to know the rank of $f_y(x)$ we have to compute the values $f_i(x)$ for $i = 1, \ldots, K$. However, there is a sampling trick that can solve this problem making it much more efficient: the idea is that we can sample labels i uniformly with replacement until we find a violating label. If there are $k = rank^1_y(f(x))$ violating labels, the random variable N_k which counts the number of trials in our sampling step follows a geometric distribution of parameter $\frac{k}{K-1}$ (i.e. $\Pr(N_k > q) = (1 - \frac{k}{K-1})^q$). Thus $k = \frac{K-1}{E[N_k]}$. This suggests that the value of $rank^1_y(f(x))$ may be approximated by:

$$rank^1_y(f(x)) \approx \left\lfloor \frac{K-1}{N} \right\rfloor$$

where $\lfloor . \rfloor$ is the floor function and N the number of trials in the sampling step. This method is called the Weighted Approximate Ranked Pairwise (WARP) loss.

12.4.2 Training Object and Label Models

Solving the image annotation problem with the semantic embedding model with objects and labels hence consists of the joint word-image embedding model of Section 12.2 trained with the WARP loss of Section 12.4.1. This method is called WSABIE in (Weston et al., 2011). The mapping matrices V and W are initialized at random with mean 0, standard deviation $\frac{1}{\sqrt{d}}$, which is a common choice. We regularize the weights of our models by giving them constrained norm:

$$||V_i||_2 \leq C, \quad i = 1, \ldots, d, \tag{12.4}$$

$$||W_i||_2 \leq C, \quad i = 1, \dots, K. \tag{12.5}$$

which acts as a regularizer in the same way as is used in lasso (Tibshirani, 1996). During SGD, the initial weights are rescaled if they violate the constraints (12.4)-(12.5).

The task described here is fully supervised. The task of learning object and part models is the subject of the next subsection.

12.4.3 Training Word, Object and Part models

Proxy supervision

Learning to label objects and their parts in images requires a vast amount of precisely labeled data which is costly to collect. This problem is particularly challenging because it involves both a hard data collection problem and a nontrivial model learning problem. We first propose an approach to address the data collection issue.

Our method is based on two observations. First, large datasets exist today with images labeled with their depicted objects (e.g. (Deng et al., 2009a)). These datasets contain millions of images annotated with thousands of terms. Second, many knowledge bases (including (Miller, 1995; Bollacker et al., 2008; Suchanek et al., 2008)) provide various semantic relation between words (such as *part-of*). We propose to combine these two kinds of data sources to provide training data for our task.

Specifically, we use here ImageNet (Deng et al., 2009a) and WordNet (Miller, 1995), two databases based on a common set of semantic categories. WordNet is a large database encompassing a comprehensive lexical knowledge within its graph structure, whose nodes (termed *synsets*) correspond to word senses, and edges define different types of relations including *is-a* and *part-of*. ImageNet is an image database organized according to the WordNet graph, thus providing a visual counterpart to WordNet, whereby a set of images is associated to each synset. As it is hard to obtain many images with labeled objects and parts, we propose to use instead pairs of images whose labels are semantically linked using a *part-of* relation in WordNet. This is the key idea of our proxy supervision. For instance, if we want to learn to label a "car" and its parts, WordNet provides a list of candidates for those such as "wheel", "windshield" or "sunroof". Now, using ImageNet, we can access many images of cars, of wheels, of windshields and of sunroofs. In this way, we can create many training examples by pairing images of cars to images of its parts. The difficulty, and the learning challenge, is that the images of parts do not come from the same images as the object they are supposed to belong to – they can even be very different in scale, texture and lighting. The learning algorithm must be able to somewhat abstractly represent the objects to be able to train under such an indirect supervision.

The next section details how we created our dataset. Note that, in this paper, we only consider the *part-of* relation in WordNet but our approach can be extended to other relations such as *type-of* or *instance-of*, and/or using knowledge bases other than WordNet.

The ImageNet+WordNet data set

To build our dataset, we selected $1,000$ synsets appearing in both sides of *part-of* relations of WordNet and which were depicted by (at least) 400 images in ImageNet. These $1,000$ synsets compose 771 *part-of* relations and are associated with a total of $400,490$ pictures. After splitting, we were left with a training set of $324,158$ word-image couples, a validation set of $37,920$, and a test set of $38,412$. To ensure the representativeness of the different sets, we made sure that at least one pair of examples representing all 771 *part-of* relations was present in each split.

The different models presented in the next section are trained using quadruplets (I_o, L_o, I_p, L_p) where I_o is an object image (and L_o its label) and I_p a part image (and L_p its label). We recall that the originality of our approach comes from the fact that I_p is not taken from the image I_o. Hence this proxy training data will have quadruplets that are labeled images of doors and door handles, for example, but the *door handle image is not from that particular door*. Nevertheless, one can hope to generalize from this type of proxy data in the absence of direct supervision. Examples of such quadruplets are given in Figure 12.2. An image is potentially labeled with several synsets since different objects can be present in the same image. Using the more than $400,000$ labeled images and the 771 *part-of* relations, we could construct more than 100M of such quadruplets. Such a number of training examples is impossible to reach using exhaustive image labeling. We trained on 10M quadruplets constructed using the $324,158$ training couples. For evaluation, we created 50k validation quadruplets out of the $37,920$ pairs and 100k test ones

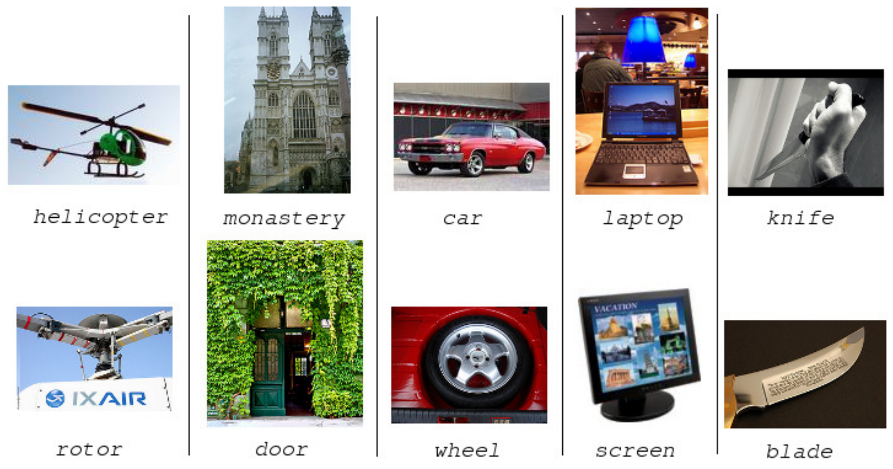


Figure 12.2 – Quadruplets from ImageNet+WordNet dataset e.g "rotor" is *part-of* "helicopter". Note that objects and their parts are not coming from the same image

out of the 38, 412 pairs.

Proxy Training Method

Proxy Ranking Task Given a positive quadruplet $x = (I_o, L_o, I_p, L_p)$ we find the parameters that try to satisfy two kinds of constraint:

$$f_q(I_o, L_o, I_p, L_p) > f_q(I_o, L', I_p, L_p), \quad \forall L' \neq L_o \tag{12.6}$$

$$f_q(I_o, L_o, I_p, L_p) > f_q(I_o, L_o, I_p, L'), \quad \forall L' \neq L_p \tag{12.7}$$

In practice not all training data constraints can be satisfied so we instead optimize a *ranking loss* that tries to optimize the precision at the top of the ranked list of labels. Following the loss defined in (Usunier et al., 2009b; Weston, Bengio, and Usunier, Weston et al.) we would like to minimize (for a single training example):

$$\mathcal{E}\left(rank_o(f_q(x))\right) + \mathcal{E}\left(rank_p(f_q(x))\right)$$

where:

$$rank_o(f_q(x)) = \sum_{L' \neq L_o} I(f_q(I_o, L', I_p, L_p) \geq f_q(x)),$$

$$rank_p(f_q(x)) = \sum_{L' \neq L_p} I(f_q(I_o, L_o, I_p, L') \geq f_q(x)),$$

i.e. $rank_o$ counts the number of labels that are ranked above the true object label and $rank_p$ counts the number of labels that are ranked above the true part label, and

$$\mathcal{E}(k) = \sum_{j=1}^{k} \alpha_j, \text{ with } \alpha_1 \geq \alpha_2 \geq \cdots \geq 0$$

is a function that transforms the ranks into a loss. If the first few indices of α index large values and smaller values follow this gives more weight to optimizing the top of the ranked list (Usunier et al., 2009b). In our experiments we set $\alpha_1 = 1$ and $\alpha_{j>1} = 0$ to optimize the proportion of top-ranked correct labels.

Object and Part Training Algorithm Following the authors of (Weston, Bengio, and Usunier, Weston et al.) we optimize such a loss function via stochastic gradient descent by iterating over the following scheme:

1. Select a positive training quadruplet x at random.
2. Select at random either constraint (12.6) or (12.7).
3. Set $N = 0$.
4. Repeat:
   (a) Create a negative triplet $\tilde{x}$ by sampling a label L' at random and constructing $\tilde{x} = (I_o, L', I_p, L_p)$ for constraint (12.6) or $\tilde{x} = (I_o, L_o, I_p, L')$ for constraint (12.7).
   (b) N=N+1

   until $f_q(x) > f_q(\tilde{x}) + 1$ or $N \geq \gamma$.
5. Make a stochastic gradient step to minimize $\mathcal{E}(\lfloor \frac{\gamma}{N} \rfloor)|1 - f_q(x) + f_q(\tilde{x})|_+$.
6. Enforce the constraints that each column $||W_i|| \leq 1$, $||V_i|| \leq 1$, $\forall i$ (this only involves updating the parameters that have changed in the gradient step).

Similarly to the method previously described in Section 12.4.1, step (4) approximates the calculation of $rank_o$ or $rank_p$ rather than explicitly calculating them: instead of counting all violating constraints one keeps sampling N times (up to a maximum of γ) until one finds a single violating constraint. The rank is then approximated with $\frac{\gamma}{N}$ instead of using the true rank. Intuitively, if we have to sample for a long time to find a violating constraint, the true rank must be relatively small. This results in practically useful algorithms in terms of training time.

Hyperparameters We use an embedding dimension of $D = \{50, 100\}$ in all our experiments. For sampling negative samples, we choose $\gamma \in \{10, 100\}$ on various size of mini-batch $\{100, 1000\}$. The learning rates are set to $\{0.01, 0.001\}$ for W, U and $\{0.1, 0.01\}$ for V.

For the image mapping $\Phi(\cdot)$ we use a standard bag-of-visual-terms type representation, which has a sparse vector representation. In particular, we use the bag-of-terms feature setup of (Grangier and Bengio, 2008), which was shown to perform well on image ranking tasks. The image is first segmented into several overlapping square blocks at various scales. Each block is then represented by the concatenation of color and edge features. These are discretized into a dictionary of $N = 10,000$ blocks, by training k-means on a large corpus of images. Each image can then be represented as a bag of visual words: a histogram of the number of times each visual word was present in the image, yielding N dimensional vectors with an average of 245 non-zero values.

12.5 Related Work

Part-based approaches There has been extensive work on decomposing objects into parts, and using parts and their relations to model objects and their shapes. Usually, the "parts" in these approaches are not viewed as isolated objects by their own merits (like the flashlight of a camera), but rather capture recurring patterns in patches that cover parts of the images (like a bottom left side of a camera). For example, (Agarwal et al., 2004) learned models of parts and their spatial relations from sets of patches in images, and then used them to label objects, such as cars in urban environments. The current work has a different aim of explicitly tagging the presence of parts in images. Another related work is by (Crandall and Huttenlocher, 2006), who used a Markov random field to learn a rich appearance and shape model with training data that only had class labels. Once again the parts learned in this work do not correspond to smaller objects but rather to patches that cover parts of objects.

Object Detection using Part or Attribute-based models Using parts or attributes has shown promising results for object detection. (Farhadi et al., 2009b) build an object representation where objects are described as a collection of semantically meaningful attributes e.g. parts, material. Also, (Felzenszwalb et al., 2009) shows how to train state of the art object detectors which first localize parts of an object in combination with a whole object detector. These approaches differ from ours since we are focusing on classification and image retrieval and not on detection.

Image labeling with embedding-based models The idea of using embedding algorithms for image labeling has been tried using several varying methods. PLSA (Monay and Gatica-Perez, 2004) was applied but has been shown to perform worse than (non-embedding based) supervised ranking models such as the large margin classifier PAMIR (Grangier and Bengio, 2008). Embedding for image retrieval (rather than annotation) using KCCA was also explored in (Zhou et al., 2007). The most related work to ours is that of (Weston, Bengio, and Usunier, Weston et al.) which uses a ranking criterion optimizing the top of the ranked list like ours to label images where they obtained very good results compared to non-embedding approaches (such as PAMIR) on large scale tasks. None of these works however explored labeling parts in images. The current work aims to learn richer models which incorporate both image-label similarity functions and object-label to part-of-label similarity functions. We focus on the difficulty of training such models given a lack of large-scale supervision for the part-based tasks we are interested in.

12.6 Empirical Evaluation

We first briefly give experimental results in an image annotation setup which labels *objects only* (Section 12.6.1). These results justify the general embedding approach and choice of loss function that we use in the rest of our experiments on objects and parts, and are a summary of the results from (Weston et al., 2011). Then, we present new results for the object and part model we are interested in (Sec.12.6.2).

12.6.1 Objects Only

The approach we advocate in this setup is to use the embedding algorithm detailed in Section 12.2 with the WARP loss of Section 12.4.1. This method is called WSABIE (Web Scale Annotation By Image Embedding).

Dataset ImageNet (Deng et al., 2009b) is a large-scale image dataset organized according to WordNet (Fellbaum, 1998). Concepts in WordNet, described by multiple words or word phrases, are hierarchically organized. ImageNet is a growing image dataset that attaches quality-controlled human-verified images to these concepts. The version that was downloaded for these experiments had 4.1 million images and 15,952 classes. The data was split into 2.5M images for training, 0.8M for validation and 0.8M for testing, removing duplicates between train, validation and test by throwing away test examples which had too close a nearest neighbor training or validation example in feature space.

Feature Representations As mentioned before, we will consider a bag-of-visual-terms type representation, but many other types of feature representation appear in the literature. We hence also explored the possibility that an ensemble of feature representations that can improve performance as has been shown before (Makadia et al., 2008). We thus combined multiple feature representations which are the concatenation of various spatial (Grauman and Darrell, 2007) and multiscale color and texton histograms (Leung and Malik, 1999) for a total of about 5×10^5 dimensions. We then perform Kernel PCA (Schoelkopf et al., 1999) on the combined feature representation using the intersection kernel (Barla et al., 2003) to produce a 1024 dimensional input vector for training Wsabie. We then train Wsabie as before.

Baselines We compare the embedding approach WSABIE to several baselines: approximate k-nearest neighbors (k-NN), exact k-NN one-versus-rest linear large margin classifiers (One-Vs-Rest) of the form $f(I_o, L_o) = w_i^\top \Phi_{\mathcal{I}}(I_o)$ trained separately for each class L_o, or the same models trained with an AUC type ranking loss instead (sometimes called the multiclass loss). For all methods, hyperparameters are chosen via the validation set.

Table 12.1 – Summary of Image Annotation (Object labeling only) Test Set Results on ImageNet. Precision at 1 and 10 are given.

Algorithm	Features used	p@1	p@10
Approx. k-NN	Bag of Visterms	1.55%	-
Exact k-NN	Bag of Visterms	4.50%	-
One-vs-Rest Linear Machine	Bag of Visterms	2.27%	1.02%
AUC Ranking-loss Linear Machine	Bag of Visterms	3.14%	1.26%
Wsabie [d=500]	Bag of Visterms	6.14%	2.09%
Exact k-NN	Ensemble+KPCA	7.73%	-
Wsabie [d=1000]	Ensemble+KPCA	10.03%	3.02%

Table 12.2 – WARP vs. AUC optimization. We compared the two types of loss function WARP and AUC or two types of model: semantic embedding models and linear models. For both model types, WARP improves over AUC.

Model	AUC p@1	WARP p@1
$f(I_o, L_o) = (V\Phi_I(I_o))^\top W_{\Phi_L(L_o)}$ [D=100]	1.65%	**4.03%**
$f(I_o, L_o) = w_i^\top \Phi_I(I_o))$	3.14%	**4.25%**

We tested approximate k-NN because k-NN is often not feasible. There are many flavors of approximation (see, e.g (Torralba et al., 2008)). We chose the following: a random projection at each node of the tree is chosen with a threshold to go left or right that is the median of the projected training data to make the tree balanced. After traversing p nodes we arrive at a leaf node containing $t \approx n/2^p$ of the original n training points from which we calculate the nearest neighbors. Choosing p trades off accuracy with speed.

Metrics We compared our models using the precision at the top k of the list (p@k). This metric gives more importance to the true annotations appearing in the top of the list and is defined as:

$$\text{p@k} = \frac{1}{N}\sum_{i=1}^{N}\frac{I\{\text{PredictedRank}(x_i) < k\}}{k}$$

where N is the size of the set, x_i is an image and its label, PredictedRank a function that returns the rank of x_i according to the considered score function and $I\{X\}$ returns the number of elements of the set X.

Results The results of comparing all methods on ImageNet are summarized in Table 12.1. As expected, the choice of feature representation has a large impact

Table 12.3 – Summary of Test Set Results on ImageNet-WordNet. Precision at 1 and 10, and Mean Reciprocal Rank (MRR) are given. (IW) resp. (I) refers to the (Image,Word) setup resp. (Image).

	Image Annotation			Part-Object Labeling			Triplet		
Models	p@1	p@10	MRR	p@1	p@10	MRR	p@1	p@10	MRR
Shared (IW)	–	–	–	**11.48%**	**3.40%**	**0.1892**	26.31%	**9.90%**	0.5545
UnShared (IW)	–	–	–	10.01%	3.02%	0.1669	**33.13%**	9.62%	**0.5595**
Shared (I)	11.21%	3.85%	0.2021	5.13%	1.84 %	0.0955	11.21%	3.85%	0.2021
UnShared (I)	**12.94%**	**4.10%**	**0.2219**	6.08%	2.11%	0.1118	12.94%	4.10%	0.2219
SVM	10.02%	3.72%	0.1864	–	–	–	10.02%	3.72%	0.1864

on the results. The ensemble+KPCA features outperform Bag of Visterms independent of the learning algorithm. However, for both feature types Wsabie outperforms the competing methods tried. The choice of embedding dimension D for Wsabie is given in the Table. Smaller values give slightly worse results, e.g. $D = 100$ with Bag of Visterms gives 4.03%.

We also compared different models trained with either WARP or AUC optimization: either the embedding approach of Wsabie ($D = 100$) or a full linear model. The results given in Table 12.2 show WARP gives superior performance.

12.6.2 Objects and Parts

In this Section we present results with our models that learn the semantics of both objects and their parts.

Notation summary

— $f^S_{IW} : \mathcal{I} \times \mathcal{L} \times \mathcal{I} \times \mathcal{L} \to \mathbb{R}$ returns a score given a quadruplet of images and annotations of a part and an object. This model uses the same semantic space for encoding the WordNet relationships and image visual similarities.

— $f^U_{IW} : \mathcal{I} \times \mathcal{L} \times \mathcal{I} \times \mathcal{L} \to \mathbb{R}$ uses different unshared semantic spaces for WordNet relationships and image visual similarities.

— $f^S_I : \mathcal{I} \times \mathcal{L} \times \mathcal{I} \times \mathcal{L} \to \mathbb{R}$ shares the same semantic space for part-of relationships and image visual similarities but does not use the Part-Object score f_L for ranking.

— $f^U_I : \mathcal{I} \times \mathcal{L} \times \mathcal{I} \times \mathcal{L} \to \mathbb{R}$ corresponds to the original WSABIE system.

Models We consider two models denoted "Shared" and "Unshared". The "Shared" model shares the label embedding matrix W for both functions f_I and f_L, and the "Unshared" model has an additional label embedding matrix U for f_I^U (see Section 12.3).

The two models are considered in two different setups. In the first case, "Image-Word", the score of a quadruplet (I_o, L_o, I_p, L_p) is computed using images and word label-label interactions. The score function for the "Shared" model is:

$$f_{IW}^S(I_o, L_o, I_p, L_p) = f_I(I_o, L_o) + f_I(I_p, L_p) + \alpha f_L(L_o, L_p)$$

and for the "Unshared" model:

$$f_{IW}^U(I_o, L_o, I_p, L_p) = f_I^U(I_o, L_o) + f_I^U(I_p, L_p) + \alpha f_L(L_o, L_p)$$

where α is an hyperparameter set to 0.1 in our experiments.

In the second case (denoted "Image"), the score is given by image-label interactions only (it does not use the within-labels score f_L) in the "Shared" model:

$$f_I^S(I_o, L_o, I_p, L_p) = f_I(I_o, L_o) + f_I(I_p, L_p).$$

and in the "Unshared" model:

$$f_I^U(I_o, L_o, I_p, L_p) = f_I^U(I_o, L_o) + f_I^U(I_p, L_p).$$

As a baseline classifier, we also compare our models with a multiclass SVM trained in a one-vs-rest setting with LibLinear (Fan et al., 2008).

Metrics As before, we compared our models using the precision at the top k of the list (p@k), as well as the mean reciprocal rank (MRR), which is defined as:

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{PredictedRank}(x_i)},$$

where N is the size of the set, x_i is a quadruplet, PredictedRank a function that returns the rank of x_i according to the considered score function and $I\{X\}$ returns the number of elements of the set X.

Tasks We consider three different tasks.

The first task, termed **Image Annotation**, is a standard image labeling task where our part-based architecture is not really useful, but still serves as a kind of sanity check of our system. Given one image I_o of an object or a part I_p, the model has to predict the correct label among $K = 1000$ different synsets. This task is used to assess the basic quality of our image labeling model f_I^U or f_I^S w.r.t. the SVM. Since the Image-Word models use $f_L(.,.)$ to compute the ranking score which has no relation whatsoever with the image property, we will not evaluate them in this task.

Our second task, termed **Part-Object Labeling**, is particularly of interest for our setting to evaluate the gain of using a joint prediction model using image and words as we propose. Given an object I_o and a part I_p, one has to predict the correct pair of labels (L_o, L_p) over $K^2 = 1$M possible pairs.

Our third task, termed **Triplet**, is derived from the previous, but it assumes that either the object I_o or the object part I_p has already been labeled (or is known using prior knowledge). Hence, given a pair of images I_p and I_o, and the label of either the part L_p or the object L_o, the model has to correctly predict the missing label.

The two last tasks use the ranking functions f as defined above i.e f_{IW}^U, f_{IW}^S, f_I^U and f_I^S depending on the setup of the models that are being evaluated. The best models are selected w.r.t. their mean rank error on the joint part-object labeling task by using f_{IW}^S, f_{IW}^U, f_I^S or f_I^U computed on 50k quadruplets of the validation set from ImageNet+WordNet and ranked over 1 M pairs.

Training and Testing The difference between the Shared (Image) and Unshared (Image) appears during training. Considering **Image Annotation**, even if f_L is not used for computing the score at the end, the training process will perform a gradient step on W with respect to (L_p, L_o), (I_o, L_o) and (I_p, L_p) for the shared model while the unshared parameters only receive updates coming from (I_o, L_o) and (I_p, L_p).

Unshared (Image) is similar to the original WSABIE model and is presumably better than Shared (Image) which will push part-object word embeddings to be closer. For example in the Shared (Image) model, *car* and *wheel* will be close in the embedding space but having *wheel* ranked high for *car* is naturally not optimal.

12.6.3 ImageNet-WordNet Data

Results on the test set of our custom data are presented in Table 12.3. On the standard **Image annotation** task, all considered models reach reasonable performance and compare nicely with the SVM. We would expect that the models using (I) would achieve the same value because on this particular task. However our models are selected using a validation score on the **Part-Object Labeling** and thus different configurations are picked, resulting in different performances.

On the **Triplet** task, models (I) have the same performance as for Image annotation because they are unable, by construction, to use the information coming from the already assigned label. Models (IW) using that information experience an increase in accuracy for all criteria: using the within-label score f_L is a major plus that these methods can exploit. They reach values of 33.13% p@1 which on a task with 1,000 labels is a fairly remarkable achievement.

Results of the **Part-Object Labeling** task confirm the usefulness of using our joint scoring model. Indeed, both models (IW) using all scores clearly outperform the competing methods. On top of that, sharing embeddings during training brings an additional improvement because the labels have acquired richer representations through their multi-task training within f_I and f_L.

12.6.4 LHI Data

We further evaluated our models on a set of objects pairs that were manually annotated as objects and their parts. To create this data, we used a set of images annotated by Labelme (Russell et al., 2008) and parsed by the Lotus Hill Research Institute (LHI) (Yao et al., 2009).

Data collection We downloaded the LHI data from *yoshi.cs.ucla.edu/yao.data/*. For every image, this dataset contains an annotation of the image and list of bounding boxes surrounding parts of the object together with the annotation of all parts. We manually matched these annotations with the list of WordNet synsets (for example by matching the term 'camera' to the relevant synset "n03358726"). We then intersected the list of synset-pairs which exist in ImageNet with the list of annotation pairs in the LHI data, and extracted images of parts from the bounding boxes. This yielded a set of image pairs, each having an object and its part, and

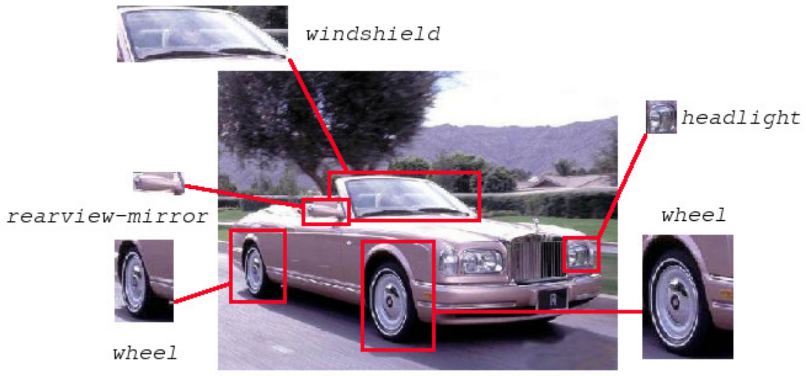


Figure 12.3 – Example from LHI data depicting an image of an object and its parts.

each labeled with a synset that can be modeled using ImageNet. An example of such a set is depicted in Figure 12.3.

The LHI data contains 489 image pairs (Object,Part) with 19 *part-of* relations. Note that 10 of those relations do not appear in our training set. A perfect example: ”Propeller is *part-of* Wing is *part-of* Aircraft”. ”Propeller is *part-of* Aircraft” is not in our training set of 771 relations but ”Propeller is *part-of* Wing” and ”Wing is *part-of* Aircraft” are. Moreover, it is also important to add that the patches representing objects are also usually in a much lower resolution than the images from ImageNet.

Results This evaluation set is harder than the previous one due to its real-world conditions. This is reflected in the results presented in Table 12.4. We can still draw the same conclusions as for the previous section. That is, using the within-label score improves performances i.e. (IW) is always better than (I). Furthermore, these results show that our proxy supervision is useful and allows the model to learn a prediction function that can nicely transfer to real-world data even though such data *has never been seen in training.*

12.6.5 WordNet tree and generalization to unseen relationships

The Image-Word models (Shared and Unshared) learn an embedding encoding the part-of relationships but one can think of directly using the WordNet tree.

Table 12.4 – Summary of Results on LHI data

	Image Annotation			Part-Object Labeling			Triplet		
Models	p@1	p@10	MRR	p@1	p@10	MRR	p@1	p@10	MRR
Shared (IW)	–	–	–	**0.25%**	**0.13%**	**0.0075**	9.10%	4.43%	0.2044
UnShared (IW)	–	–	–	0.0%	0.10%	0.0043	**18.20%**	**4.79%**	**0.2778**
Shared (I)	**10.85%**	**2.92%**	**0.1728**	0.0%	0.08%	0.0039	10.85%	2.92%	0.1728
UnShared (I)	7.26%	2.37%	0.1288	0.0%	0.03%	0.0025	7.26%	2.37%	0.1288

Notation summary

— $f_{IO}^{S} : \mathcal{I} \times \mathcal{L} \times \mathcal{I} \times \mathcal{L} \rightarrow \mathbb{R}$ shares the same semantic space for part-of relationships and images visual similarities. An offset is added to the ranking score if the part-of relationship was present in the training set.

— $f_{IO}^{U} : \mathcal{I} \times \mathcal{L} \times \mathcal{I} \times \mathcal{L} \rightarrow \mathbb{R}$ corresponds to the original WSABIE system where an offset is added to the score in the same way as before in order to filter out the training relationships.

In order to use this prior in the Image models (Shared and Unshared), we add an offset to the original score if the part-of relationship is in the training set:

$$f_{IO}^{S}(I_o, L_o, I_p, L_p) = f_{I}^{S}(I_o, L_o, I_p, L_p) + O(L_p, L_o).$$

$$f_{IO}^{U}(I_o, L_o, I_p, L_p) = f_{I}^{U}(I_o, L_o, I_p, L_p) + O(L_p, L_o).$$

where

$$O(L_p, L_o) = \begin{cases} \lambda & \text{if } (L_p, L_o) \text{ is in the training relationships.} \\ 0 & \text{otherwise.} \end{cases}$$

In addition, we wanted to stress the ability of Image-Word models for generalizing to unseen part-of relationships. Indeed given 'wheel is-part-of car' and that 'car' shares visual similarities with 'truck', Image-Word models learn to push truck-wheel embeddings together even if 'wheel is-part-of truck' is not present in the training data.

The LHI dataset is particularly appropriate for these two experiments. All the models have been trained on 771 part-of relationships but 50% of the part-of relationships in LHI (10/19) are not in those training relationships. When testing

Table 12.5 – WordNet Tree and Unseen relationships Image-Word Shared models allow to generalize to unseen relationships. Using the WordNet tree also improves the performance. (IO) refers to the original Image where an offset was added to filter out relations both in the WordNet tree and the training set. Part-Object relationships are ranked among 781 couples.

	Part-Object Labeling using WordNet		
Models	p@1	p@10	MRR
Shared (IW)	**3.06%**	**2.45%**	**0.1005**
UnShared (IW)	2.04%	1.64%	0.0670
Shared (IO)	1.43%	1.17%	0.0483
UnShared (IO)	1.23%	1.45%	0.0519

Table 12.6 – Nearest Neighbors of word embeddings learned with Shared and Unshared models. Shared embeddings exhibit visual similarities in addition to semantics *e.g.* strings of an instrument and laces of a shoe, a radio-phonograph is usually set up on a desk or a dresser.

Toe		**Radio-phonograph**		**Loudspeaker**	
shared	*unshared*	*shared*	*unshared*	*shared*	*unshared*
footwear	shoe	dresser	tuner	lens	public address system
tongue	accelerator pedal	rotor head	amplifier	sustaining pedal	amplifier
stringed instrument	storage	firebox	public address system	public address system	tuner
shoe	boot	rear light	loudspeaker	optical instrument	radio-phonograph

on the LHI dataset, we filtered out the results keeping all the part-of relationships among these 781 couples (771 training relationships and 10 unseen during training coming from LHI). Therefore, in the Image models score (Shared and Unshared), an offset (we used $\lambda = 100$ in our experiments) was added if the part-of relationship was among the 771 training relationships. The Image-Word models (Shared and Unshared) are kept as before.

Results in Table 12.5 highlight the advantage of using Image-Word models, those are able to generalize to unknown relationships (10 relationships in LHI out of the training set) through word smoothing in the embedding space. Using the WordNet tree to filter out relationships among the 1M possible couples (by keeping only 781 relationships) is also helpful as the performance increased drastically for all models (in comparison with results w/o using WordNet tree in Table 12.4).

12.6.6 Encoded Semantics

Example word embeddings learned by our shared and unshared models on the ImageNet-WordNet data are given in Table 12.6. We observe that the embeddings

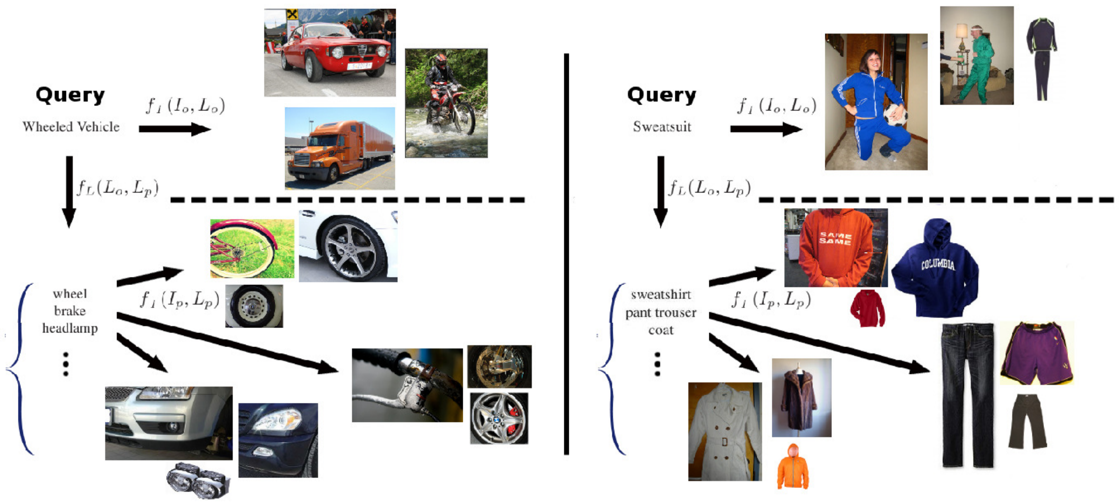


Figure 12.4 – Semantically Augmented Image Search. For a given query (”wheeled vehicule” on the left and ”Sweatsuit” on the right), our model can retrieve corresponding images using f_I (above the dashed line) but can also automatically extend the query to associated terms (object parts in this work) using f_L and hence retrieve semantically related images using f_I (below the dashed line). These results were obtained with the unshared model.

capture the semantic structure of the labels. Besides, the shared model also encodes some information regarding the visual aspect of the object (or part) in its features. Hence, stringed instruments and shoes have strings and laces as parts, which are visually alike.

Figure 12.4 illustrates an application for which our model could readily be applied, which we could term *semantically augmented search*. Given a text query, our model can retrieve corresponding images using the f_I function, as standard *object only* image retrieval approaches do. However, for no additional cost, our model can also extend the original query by searching for semantically associated terms (which are only parts in our current work) using the f_L function, and can finally return part images related to those by leveraging the f_I function. Therefore, our model can generalize an image query to search for other objects which have little or no visual similarity with the original object of interest but are semantically linked to it, hence improving the original search and browsing experience. This is related to recent efforts on attribute-based search pulled off by search engines.

12.7 Conclusion and Future Directions

This paper presented an original approach to learn to automatically tag images with their objects and their object-parts. In particular, we proposed a proxy supervision that allows to bypass the high cost of annotating such data. We illustrated this setting by introducing a new dataset created by connecting ImageNet and WordNet through part-based within-label relations. We also presented a model that can be trained under such an indirect supervision by jointly encoding both object image to object label, and object label to object part label similarities. Our experimental evaluation performed on our custom test set as well as on exhaustively annotated data showed that this model can be successfully trained under our proxy supervision.

Our idea of merging vision and semantic data is not limited to labeling objects and object parts. We chose this setting because it may be the most obvious and because it has many potential applications. However, our proxy supervision and our associated algorithm could also be used in other contexts and with other databases. For instance, one could think of connecting Labeled Faces in the Wild (LFW) (Huang et al., 2007) as the image data and Freebase (Bollacker et al., 2008) as the semantic graph among labels. Indeed, Freebase proposes a vast graph linking persons with various relations such as *married-to*, *son-of*, etc. that matches well the set of labels of LFW. By connecting these two resources, one could build up a model able to leverage some underlying connection between the members of a group picture to jointly label them. This is just an example of future directions/applications that could be explored with our new proxy supervision for image annotation.

12.8 Acknowledgements

This work was supported by the DARPA Deep Learning Program, NSERC, CIFAR, the Canada Research Chairs, Compute Canada and by the French ANR Project ASAP ANR-09-EMER-001. Codes for the experiments has been implemented using both Torch (Collobert et al., 2011) and Theano (Bergstra et al., 2010a) Machine Learning libraries.

13 Conclusion Générale

L'apprentissage de réseaux de neurones profonds a évolué très rapidement au cours de la réalisation de ce doctorat. Ces changements sont reflétés dans nos travaux qui ont suivi l'évolution riche de la recherche en architectures profondes des dernières années.

Nous avons commencé par présenter de nombreuses techniques d'apprentissage non supervisé, puis comment les mettre en oeuvre pour gagner une compétition de transfert d'apprentissage ou améliorer la performance d'un système de vision tout en réduisant la dimensionnalité de l'espace d'entrée original. Ces solutions nous ont permis d'apprendre un espace sémantique de représentation.

Dans le chapitre 4, nous avons présenté des solutions performantes pour extraire un espace sémantique de représentation permettant une meilleure séparabilité linéaire. La PCA transductive a aussi permis d'effectuer un transfert d'apprentissage tout en limitant la présence de bruit préjudiciable à une classification linéaire. Ces idées ont permis d'améliorer notre solution finale soumise lors de la compétition et ainsi de remporter la première place.

Dans le chapitre 6, nous avons appliqué ces idées à un espace d'entrée de dimension élevée mais structurée. La qualité de l'espace sémantique de représentation par rapport à l'espace d'entrée original a nettement progressé en matière de compression d'information et de performance. Nous avons aussi démontré la capacité des réseaux de neurones à effectuer du transfert d'apprentissage.

Ensuite, nos intuitions concernant l'espace sémantique des représentations apprises ont été résumées dans nos recherches sur l'amélioration du mixage lors du processus d'échantillonnage effectué dans les couches profondes les plus abstraites (chapitre 8). Il a été très intéressant de sortir du cadre probabiliste de l'échantillonnage des RBMs pour s'intéresser à des intuitions plus géométriques sur la sémantique de l'espace et se déplacer ainsi sur la variété.

La présentation des vidéos permettant un déplacement en images sur la variété des exemples d'entraînement a donné lieu à d'intéressantes conversations scientifiques lors de plusieurs colloques. Ces vidéos contenaient des chiffres passant d'une classe à l'autre ou des visages changeant d'expression.

Pour terminer, nous avons étudié deux cas où l'espace sémantique est appris soit en sortie soit en entrée du réseau de neurones. L'utilisation d'architectures prometteuses telles que les réseaux de neurones récurrents a été la base d'une transition vers l'apprentissage supervisé d'espaces sémantiques associés au langage (chapitre 10). À la fin du processus d'apprentissage, nous avons observé que l'espace sémantique était structuré sous forme de grappes de mots possédant la même classe.

À l'heure actuelle, les récents succès de l'industrie résident dans l'utilisation d'ensemble de données de très grande échelle. Avec un grand ensemble de données (plusieurs millions d'exemples), le chapitre 12 se concentre sur le transfert d'apprentissage au travers d'un espace sémantique multidomaine. Un aspect intéressant de ce travail a été de sortir du cadre de maximisation de la vraisemblance pour optimiser un système à base de maximisation de marge pour une classification en haute dimension.

Dans le futur, nous espérons poursuivre nos travaux de recherche pour apprendre des espaces sémantiques avec une structure plus riche et un nombre de domaine arbitraire. L'apprentissage non supervisé n'atteint pas encore le niveau des performances récemment obtenues en apprentissage supervisé d'architectures profondes pour des tâches de vision, de reconnaissance de la parole ou de traitement du langage. Il est possible que le problème actuel puisse être résolu autrement qu'avec une classique erreur de reconstruction accompagnée d'une régularisation sur les paramètres. Des contraintes différentes peuvent être envisagées. J'espère pouvoir apporter des solutions originales à ce problème dans les années à venir.

Bibliographie Personnelle

2015

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu and Geoffrey Zweig **Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding** (2015) IEEE Transactions on Audio, Signal and Language Processing, in press

Grégoire Mesnil, Salah Rifai, Antoine Bordes, Xavier Glorot, Yoshua Bengio and Pascal Vincent **Unsupervised Learning of Object Detections Semantics for Scene Categorization** (2015) Pattern Recognition Applications and Methods, Advances in Intelligent Systems and Computing, pages 209-224

Grégoire Mesnil, Tomas Mikolov, Marc'Aurelio Ranzato and Yoshua Bengio **Ensemble of Generative and Discriminative Techniques for Sentiment Analysis of Movie Reviews** (2015) International Conference on Learning Representations (submitted in the workshop track)

2014

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng and Grégoire Mesnil **Learning Semantic Representations Using Convolutional Neural Networks for Web Search** (2014) Proceedings of the World Wide Web Conference, poster.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng and Grégoire Mesnil **A Convolutional Neural Network based Latent Semantic Model for Web Search** (2014) Proceedings of the Conference on Knowledge and Information Management.

2013

Grégoire Mesnil, Xiaodong He, Li Deng and Yoshua Bengio **Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding** (2013) Proceedings of the Interspeech Conference

Yoshua Bengio, Grégoire Mesnil, Yann Dauphin and Salah Rifai **Better Mixing via Deep Representations** (2013) Proceedings of the 30th International Conference on Machine Learning

Grégoire Mesnil, Antoine Bordes, Jason Weston, Gal Chechik and Yoshua Bengio, **Learning Semantic Representations Of Objects and Their Parts** (2013) Machine Learning Journal, volume 94, pages 281–301

Grégoire Mesnil, Salah Rifai, Xavier Glorot, Antoine Bordes, Yoshua Bengio and Pascal Vincent, **Unsupervised and Transfer Learning under Uncertainty: from Object Detections to Scene Categorization** (2013) Proceedings of the International Conference on Pattern Recognition Applications and Methods (oral)

2012

Grégoire Mesnil, Salah Rifai, Yann Dauphin, Yoshua Bengio and Pascal Vincent, **Surfing on the Manifold** (2012) Learning Workshop (poster), Snowbird, Utah, U.S.A.

Grégoire Mesnil, Yann Dauphin, Xavier Glorot, Salah Rifai, Yoshua Bengio, et al. **Unsupervised and Transfer Learning Challenge: a Deep Learning Approach** (2012) Journal of Machine Learning Workshop and Conference Papers, Proceedings of the Unsupervised and Transfer Learning challenge and workshop, pages 97-110

2011

Grégoire Mesnil, Salah Rifai, Xavier Glorot, Antoine Bordes, Pascal Vincent and Yoshua Bengio, **Exploiting context-based Information for Scene**

Categorization (2011) Neural Information Processing Systems Workshop on Learning Semantics (poster)

Salah Rifai, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin and Xavier Glorot, **Higher Order Contractive Auto-Encoder** (2011) Proceedings of the European Conference on Machine Learning

Salah Rifai, Xavier Muller, Grégoire Mesnil, Yoshua Bengio and Pascal Vincent, **Learning Invariant features through local space contraction** in Technical Report 1360. Département d'informatique et de recherche opérationnelle. Université de Montréal 2011.

Bibliographie

Agarwal, S., A. Awan, and D. Roth (2004). Learning to detect objects in images via a sparse, part-based representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*(11), 1475–1490.

Alain, G. and Y. Bengio (2012). What regularized auto-encoders learn from the data generating distribution. Technical Report Arxiv report 1211.4246, Université de Montréal.

Baldi, P. and K. Hornik (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks 2*, 53–58.

Barla, A., F. Odone, and A. Verri (2003). Histogram intersection kernel for image classification. *International Conference on Image Processing (ICIP) 3*, III–513–16 vol.2.

Baxter, J. (1997). A Bayesian/information theoretic model of learning via multiple task sampling. *Machine Learning 28*, 7–40.

Bengio, S. (2013). Devise: A deep visual-semantic embedding. In *NIPS Workshop: Extreme Classification: Multi-Class and Multi-Label Learning with Millions of Categories, Session Extreme Deep and Visual Learning.*

Bengio, Y. (2008). Neural net language models. *Scholarpedia 3*(1), 3881.

Bengio, Y. (2009a). Learning deep architectures for AI. *Foundations and Trends in Machine Learning 2*(1), 1–127. Also published as a book. Now Publishers, 2009.

Bengio, Y. (2009b). Learning deep architectures for ai. In *, Now Publishers.*

Bengio, Y. (2011, June). Deep learning of representations for unsupervised and transfer learning. In *Workshop on Unsupervised and Transfer Learning (ICML'11).*

Bengio, Y. and O. Delalleau (2011). On the expressive power of deep architectures. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory.* Eds. Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann.

Bengio, Y., O. Delalleau, and N. Le Roux (2006). The curse of highly variable functions for local kernel machines. In Y. Weiss, B. Schölkopf, and J. Platt (Eds.), *Advances in Neural Information Processing Systems 18 (NIPS'05)*, pp. 107–114. Cambridge, MA: MIT Press.

Bengio, Y., R. Ducharme, and P. Vincent (2000). A neural probabilistic language model. In *, NIPS*.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin (2003). A neural probabilistic language model. See**?**, pp. 1137–1155.

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle (2007a). Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman (Eds.), *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pp. 153–160. MIT Press.

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle (2007b). Greedy layer-wise training of deep networks. In *Adv. Neural Inf. Proc. Sys. 19*, pp. 153–160.

Bengio, Y. and Y. LeCun (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston (Eds.), *Large Scale Kernel Machines.* MIT Press.

Bergstra, J. and Y. Bengio (2012). Random search for hyper-parameter optimization. In *, Journal of Machine Learning Research 13 (2012) 281-305.*

Bergstra, J., O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio (2010a, June). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy).* Oral.

Bergstra, J., O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio (2010b). Theano: A cpu and gpu math expression compiler,. In *Proc. Python for Scientific Computing Conference (SciPy)*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bollacker, K., C. Evans, P. Paritosh, T. Sturge, and J. Taylor (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250. ACM.

Bosch, A., A. Zisserman, and X. Muñoz (2006). Scene classification via plsa. In *In Proc. ECCV*, pp. 517–530.

Boser, B., I. Guyon, and V. Vapnik (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152. ACM.

Bottou, L. (2011). From machine learning to machine reasoning,.

Breuleux, O., Y. Bengio, and P. Vincent (2011). Quickly generating representative samples from an rbm-derived process. *Neural Computation 23*(8), 2058–2073.

Byrd, R. H., P. Lu, J. Nocedal, and C. Zhu (1995, September). A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput. 16*(5), 1190–1208.

Caruana, R. (1995). Learning many related tasks at the same time with backpropagation. In G. Tesauro, D. Touretzky, and T. Leen (Eds.), *Advances in Neural Information Processing Systems 7 (NIPS'94)*, Cambridge, MA, pp. 657–664. MIT Press.

Cayton, L. (2005). Algorithms for manifold learning. Technical Report CS2008-0923, UCSD.

Cho, K., T. Raiko, and A. Ilin (2010, July). Parallel tempering is efficient for learning restricted Boltzmann machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010)*, Barcelona, Spain.

Coates, A., H. Lee, and A. Y. Ng (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011).*

Collobert, R., K. Kavukcuoglu, and C. Farabet (2011). Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop.*

Collobert, R. and J. Weston (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In W. W. Cohen, A. McCallum, and S. T. Roweis (Eds.), *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, pp. 160–167. ACM.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa (2011a). Natural language processing (almost) from scratch. In *Journal of Machine Learning Research, vol 12, pages 2493-2537.*

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa (2011b). Natural language processing (almost) from scratch. *Journal of Machine Learning Research 12*, 2493–2537.

Comon, P. (1994). Independent component analysis - a new concept ? *Signal Processing 36*, 287–314.

Courville, A., J. Bergstra, and Y. Bengio (2011, June). Unsupervised models of images by spike-and-slab RBMs. In *Proceedings of the Twenty-eight International Conference on Machine Learning (ICML'11).*

Crandall, D. and D. Huttenlocher (2006). Weakly supervised learning of part-based spatial models for visual object recognition. *Computer Vision–ECCV 2006*, 16–29.

Dahl, G., D. Yu, L. Deng, and A. Acero (2012). Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. In *IEEE Transactions on Audio, Speech, and Language Processing, 20 (1), 33–42.*

Dauphin, Y., X. Glorot, and Y. Bengio (2011, June). Sampled reconstruction for large-scale learning of embeddings. In *Proceedings of the Twenty-eight International Conference on Machine Learning (ICML'11).*

Dauphin, Y., G.Tur, D. Hakkani-Tur, and L. Heck (2013). Zero-shot learning and clustering for semantic utterance classification,. In *International Conference on Learning Representations (ICLR).*

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009a). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009b). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Deng, L., G. Tur, X. He, and D. Hakkani-Tur (2012). Use of kernel deep convex networks and end-to-end learning for spoken language understanding. In *, IEEE SLT*.

Deoras, A. and R. Sarikaya (2013). Deep belief network based semantic tagger for spoken language understanding. In *, Interspeech*.

Desjardins, G., A. Courville, Y. Bengio, P. Vincent, and O. Delalleau (2010, May). Tempered Markov chain monte carlo for training of restricted Boltzmann machine. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, pp. 145–152.

Dolan, B., C. Quirk, and C. Brockett (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics (COLING 2004)*, pp. 350–356.

Elman, J. (1990). Finding structure in time. In *Cognitive Science, 14 (2).*

Espinace, P., T. Kollar, A. Soto, and N. Roy (2010). Indoor scene recognition through object detection. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK.

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin (2008, June). Liblinear: A library for large linear classification. *J. Mach. Learn. Res. 9*, 1871–1874.

Farhadi, A., I. Endres, D. Hoiem, and D. Forsyth (2009a). Describing objects by their attributes. *IEEE Conference on Computer Vision and Pattern Recognition*, 1778–1785.

Farhadi, A., I. Endres, D. Hoiem, and D. Forsyth (2009b). Describing Objects by their Attributes. *CVPR*.

Fei-Fei, L. and P. Perona (2005). A bayesian hierarchical model for learning natural scene categories. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pp. 524–531. IEEE Computer Society.

Fellbaum, C. (Ed.) (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Felzenszwalb, P., R. Girshick, D. McAllester, and D. Ramanan (2009). Object Detection with Discriminatively Trained Part-Based Models. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*.

Felzenszwalb, P., D. McAllester, and D. Ramanan (2008). A discrimitatively trained, multiscale, deformable part model. *CVPR*.

Felzenszwalb, P. F., R. B. Girshick, D. McAllester, and D. Ramanan (2010). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*(9), 1627–1645.

Flor, H. (2003). Remapping somatosensory cortex after injury. *Advances in Neurology 83*, 195–204.

Gallinari, P., Y. LeCun, S. Thiria, and F. Fogelman-Soulie (1987). Memoires associatives distribuees. In *Proceedings of COGNITIVA 87*, Paris, La Villette.

Gao, S., I. Tsang, L. Chia, and P. Zhao (2010). Local features are not lonely – laplacian sparse coding for image classification. *IEEE Conference on Computer Vision and Pattern Recognition*.

Glorot, X., A. Bordes, and Y. Bengio (2011a, April). Deep sparse rectifier neural networks. In *JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*.

Glorot, X., A. Bordes, and Y. Bengio (2011b, June). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the Twenty-eight International Conference on Machine Learning (ICML'11)*.

Goodfellow, I., Q. Le, A. Saxe, and A. Ng (2009). Measuring invariances in deep networks. In Y. Bengio, D. Schuurmans, C. Williams, J. Lafferty, and A. Culotta (Eds.), *Advances in Neural Information Processing Systems 22 (NIPS'09)*, pp. 646–654.

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial networks. Technical Report arXiv:1406.2661, arxiv.

Grangier, D. and S. Bengio (2008). A discriminative kernel-based model to rank images from text queries. *Transactions on Pattern Analysis and Machine Intelligence 30*(8), 1371–1384.

Grauman, K. and T. Darrell (2007, April). The Pyramid Match Kernel: Efficient Learning with Sets of Features. *Journal of Machine Learning Research 8*, 725–760.

Graves, A., A. Mohamed, and G. Hinton (2013). Speech recognition with deep recurrent neural networks. In *ICASSP*.

Haffner, P., G. Tur, and J. Wright (2003). Optimizing svms for complex call classification. In *ICASSP*.

Håstad, J. (1986). Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th annual ACM Symposium on Theory of Computing*, Berkeley, California, pp. 6–20. ACM Press.

Håstad, J. and M. Goldmann (1991). On the power of small-depth threshold circuits. *Computational Complexity 1*, 113–129.

He, G. Tur, L. D. D. H.-T. X. (2012). Towards deeper understanding: Deep convex networks for semantic utterance classification. In *, ICASSP*.

He, Y. and S. Young (2003). A data-driven spoken language understanding system,. In *IEEE ASRU, pp. 583–588*.

Herbrich, R., T. Graepel, and K. Obermayer (2000). Large margin rank boundaries for ordinal regression. *Advances in large margin classifiers 88*(2), 115–132.

Hinton, G., S. Osindero, and Y. Teh (2006a). A fast learning algorithm for deep belief nets. In *Neural Computation 18, 1527–1554*.

Hinton, G. E. (2006). To recognize shapes, first learn to generate images. Technical Report UTML TR 2006-003, University of Toronto.

Hinton, G. E., S. Osindero, and Y. Teh (2006b). A fast learning algorithm for deep belief nets. *Neural Computation 18*, 1527–1554.

Hinton, G. E. and R. S. Zemel (1994). Autoencoders, minimum description length, and Helmholtz free energy. In D. Cowan, G. Tesauro, and J. Alspector (Eds.), *Advances in Neural Information Processing Systems 6 (NIPS'93)*, pp. 3–10. Morgan Kaufmann Publishers, Inc.

Hofmann, T. (2001, January). Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn. 42*, 177–196.

Hoiem, D., A. Efros, and M. Hebert (2005). Automatic photo pop-up. *SIGGRAPH 24(3)*, 577–584.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology 24*, 417–441, 498–520.

Huang, G. B., M. Ramesh, T. Berg, and E. Learned-Miller (2007, October). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst.

Huang, P.-S., X. He, J. Gao, L. Deng, A. Acero, and L. Heck (2013). Learning deep structured semantic models for web search using clickthrough data. In *, ACM International Conference on Information and Knowledge Management (CIKM).*

Hyvärinen, A., J. Karhunen, and E. Oja (2001, May). *Independent Component Analysis*. Wiley-Interscience.

Jarrett, K., K. Kavukcuoglu, M. Ranzato, and Y. LeCun (2009). What is the best multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision (ICCV'09)*, pp. 2146–2153. IEEE.

Jeong, M. and G. Lee (2007). Structures for spoken language understanding: a two-step approach. In *, ICASSP.*

Jorda, M. (1997). Serial order: a parallel distributed processing approach. In *Tech. Rep no 8604, San Diego, University of California, Institute of Computer Science.*

Kavukcuoglu, K., M. Ranzato, R. Fergus, and Y. LeCun (2009). Learning invariant features through topographic filter maps. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'09)*, pp. 1605–1612. IEEE.

Kavukcuoglu, K., M. Ranzato, and Y. LeCun (2008). Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU. Tech Report CBLL-TR-2008-12-01.

Kavukcuoglu, K., P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun (2010). Learning convolutional feature hierarchies for visual recognition. In *NIPS'2010*.

Kingsbury, B., T. N. Sainath, and H. Soltau (2012). Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization,. In *INTERSPEECH*.

Krizhevsky, A., I. Sutskever, and G. Hinton (2012a). ImageNet classification with deep convolutional neural networks. In *NIPS'2012*.

Krizhevsky, A., I. Sutskever, and G. Hinton (2012b). Imagenet classification with deep convolutional neural networks. In *Advances Neural Information Processing Systems 25*.

Kudo, T. and Y. Matsumoto (2001). Chunking with support vector machine. In *, ACL*.

Kuhn, R. and R. D. Mori (1995). The application of semantic classification trees to natural language understanding. In *, IEEE Transactions on Pattern Analysis and Machine Intelligence v. 17, pp. 449-460.*

Lafferty, J., A. McCallum, and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Lang, K. J. and G. E. Hinton (1988). The development of the time-delay neural network architecture for speech recognition. Technical Report CMU-CS-88-152, Carnegie-Mellon University.

Larochelle, H., Y. Bengio, J. Louradour, and P. Lamblin (2009, January). Exploring strategies for training deep neural networks. See**?**, pp. 1–40.

Lazebnik, S., C. Schmid, and J. Ponce (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *IEEE Conference on Computer Vision and Pattern Recognition*.

LeCun, Y. (1987). *Modèles connexionistes de l'apprentissage.* Ph. D. thesis, Université de Paris VI.

LeCun, Y. (1989). Generalization and network design strategies. Technical Report CRG-TR-89-4, University of Toronto.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998a). Gradient based learning applied to document recognition. *Proc. IEEE*.

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

LeCun, Y., P. Haffner, L. Bottou, and Y. Bengio (1999). Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pp. 319–345. Springer.

Lee, H., P. Pham, Y. Largman, and A. Ng (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS'2009*.

Leung, T. and J. Malik (1999). Recognizing surface using three-dimensional textons. *Proc. of 7th Int'l Conf. on Computer Vision, Corfu, Greece*.

Li, L.-J. and L. Fei-Fei (2007). What, where and who ? classifying events by scene and object recognition. *ICCV*.

Li-Jia Li, Hao Su, E. P. X. and L. Fei-Fei (2010a). Object bank: A high-level image representation for scene classification and semantic feature sparsification. *Proceedings of the Neural Information Processing Systems (NIPS)*.

Li-Jia Li, Hao Su, Y. L. and L. Fei-Fei (2010b, September). Objects as attributes for scene classification. In *European Conference of Computer Vision (ECCV), International Workshop on Parts and Attributes*, Crete, Greece.

Liu, J., S. Cyphers, P. Pasupat, I. McGraw, and J. Glass (2012). A conversational movie search system based on conditional random fields. In *, INTERSPEECH*.

Macherey, M., F. Och, and H. Ney (2001). Natural language understanding using statistical machine translation. In *, European Conference on Speech Communication and Technology, pp. 2205-2208*.

Makadia, A., V. Pavlovic, and S. Kumar (2008). A new baseline for image annotation. In *European conference on Computer Vision (ECCV)*.

McCallum, A., D. Freitag, and F. Pereira (2000). Maximum entropy markov models for information extraction and segmentation. In *, ICML. Pp. 591-598*.

Mesnil, G., Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. J. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, P. Vincent, A. Courville, and J. Bergstra (2012). Unsupervised and transfer learning challenge: a deep learning approach. In I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver (Eds.), *JMLR W& CP: Proceedings of the Unsupervised and Transfer Learning challenge and workshop*, Volume 27, pp. 97–110.

Mesnil, G., X. He, L. Deng, and Y. Bengio (2013a). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*.

Mesnil, G., X. He, L. Deng, and Y. Bengio (2013b). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*.

Mesnil, G., S. Rifai, A. Bordes, X. Glorot, Y. Bengio, and P. Vincent (2013). Unsupervised and transfer learning under uncertainty: from object detections to scene categorization,. In *Proceedings of the 2rd International Conference on Pattern Recognition Applications and Methods (ICPRAM)*.

Mesnil, G., S. Rifai, A. Bordes, X. Glorot, Y. Bengio, and P. Vincent (2015). Unsupervised learning of semantics of object detections for scene categorization,. In *Advances in Intelligent Systems, Computing*.

Mesnil, G., S. Rifai, Y. Dauphin, Y. Bengio, and P. Vincent (2012). Surfing on the manifold. Learning Workshop, Snowbird.

Mesnil, G., S. Rifai, X. Glorot, A. Bordes, P. Vincent, and Y. Bengio (2011). Exploiting context-based information for scene categorization,. In *NIPS Workshop on Learning Semantics*.

Mikolov, T., S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur (2011). Extensions of recurrent neural network based language model. In *ICASSP*.

Mikolov, T., W. Yih, and G. Zweig (2013). Linguistic regularities in continuous space word representations. In *NAACL-HLT*.

Mikolov, T. and G. Zweig (2012). Context dependent recurrent neural network language model. In *IEEE SLT*.

Miller, G. (1995). WordNet: a Lexical Database for English. *Communications of the ACM 38*(11), 39–41.

Miller, S., R. Bobrow, R. Ingria, and R. Schwartz (1994). Hidden understanding models of natural language,. In *ACL*.

Monay, F. and D. Gatica-Perez (2004). PLSA-based image auto-annotation: constraining the latent space. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pp. 348–351. ACM New York, NY, USA.

Nair, V. and G. E. Hinton (2010). Rectified linear units improve restricted Boltzmann machines. In *Proc. 27th International Conference on Machine Learning*.

Narayanan, H. and S. Mitter (2010). Sample complexity of testing the manifold hypothesis. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Eds.), *Advances in Neural Information Processing Systems 23*, pp. 1786–1794.

Neal, R. M. (1994). Sampling from multimodal distributions using tempered transitions. Technical Report 9421, Dept. of Statistics, University of Toronto.

Oliva, A. and A. Torralba (2006). Building the gist of a scene: The role of global image features in recognition. *Visual Perception, Progress in Brain Research 155*.

Pan, S. J. and Q. Yang (2010). A survey on transfer learning. In *IEEE Transactions on Knowledge and Data Engineering*, pp. 1345–1359.

Pandey, M. and S. Lazebnik (2011). Scene recognition and weakly supervised object localization with deformable part-based models. *ICCV*.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine 2*(6), 559–572.

Peng, J., L. Bo, and J. Xu (2009). Conditional neural fields,. In *NIPS*.

Pieraccini, R., E. Tzoukermann, Z. Gorelov, J.-L. Gauvain, E. Levin, C.-H. Lee, and J. G. Wilpon (1992). A speech understanding system based on statistical representation of semantics,. In *ICASSP*.

Quattoni, A. and A. Torralba (2009). Recognizing indoor scenes. *CVPR*.

Ranzato, M., Y. Boureau, S. Chopra, and Y. LeCun (2007). A unified energy-based framework for unsupervised learning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS'07)*, San Juan, Porto Rico. Omnipress.

Ranzato, M., Y.-L. Boureau, and Y. LeCun (2008). Sparse feature learning for deep belief networks. In J. Platt, D. Koller, Y. Singer, and S. Roweis (Eds.), *Advances in Neural Information Processing Systems 20 (NIPS'07)*, Cambridge, MA, pp. 1185–1192. MIT Press.

Ranzato, M. and G. H. Hinton (2010). Modeling pixel means and covariances using factorized third-order Boltzmann machines. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'10)*, pp. 2551–2558. IEEE Press.

Ranzato, M., C. Poultney, S. Chopra, and Y. LeCun (2007). Efficient learning of sparse representations with an energy-based model. In *NIPS'2006*.

Raymond, C. and G. Riccardi (2007). Generative and discriminative algorithms for spoken language understanding,. In *INTERSPEECH*.

Rifai, S., Y. Bengio, Y. Dauphin, and P. Vincent (2012). A generative process for sampling contractive auto-encoders. In *ICML'12*.

Rifai, S., Y. Dauphin, P. Vincent, Y. Bengio, and X. Muller (2011). The manifold tangent classifier. In *NIPS'2011*. Student paper award.

Rifai, S., G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot (2011). Higher order contractive auto-encoder. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD).*

Rifai, S., P. Vincent, X. Muller, X. Glorot, and Y. Bengio (2011, June). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the Twenty-eight International Conference on Machine Learning (ICML'11).*

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning representations by back-propagating errors. *Nature 323*, 533–536.

Rumelhart, D. E., J. L. McClelland, and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge: MIT Press.

Russell, B. C., A. Torralba, K. P. Murphy, and W. T. Freeman (2008). Labelme: A database and web-based tool for image annotation. *Int. J. Comput. Vision 77*, 157–173.

Salakhutdinov, R. (2010a). Learning deep Boltzmann machines using adaptive MCMC. In L. Bottou and M. Littman (Eds.), *Proceedings of the Twenty-seventh International Conference on Machine Learning (ICML-10)*, Volume 1, pp. 943–950. ACM.

Salakhutdinov, R. (2010b). Learning in Markov random fields using tempered transitions. In Y. Bengio, D. Schuurmans, C. Williams, J. Lafferty, and A. Culotta (Eds.), *Advances in Neural Information Processing Systems 22 (NIPS'09).*

Salakhutdinov, R. and G. E. Hinton (2009). Deep Boltzmann machines. In *Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS'09)*, Volume 5, pp. 448–455.

Sarikaya, R., G. E. Hinton, and B. Ramabhadran (2011). Deep belief nets for natural language call-routing,. In *ICASSP.*

Schapire, R. E. and Y. Singer (2000). Boostexter: A boosting-based system for text categorization. In *Machine Learning, vol. 39, no.2/3, pp. 135-168.*

Schoelkopf, B., A. J. Smola, and K. R. Müller (1999). Kernel principal component analysis. *Advances in kernel methods: support vector learning*, 327–352.

Schuster, M. and K. Paliwal (1997). Bidirectional recurrent neural networks,. In *IEEE Transactions on Signal Processing.*

Schwenk, H. and J.-L. Gauvain (2005). Training neural network language models on very large corpora. In *HLT/EMNLP*.

Serre, T., L. Wolf, and T. Poggio (2005). Object recognition with features inspired by visual cortex. *IEEE Conference on Computer Vision and Pattern Recognition*.

Shen, Y., X. He, J. Gao, L. Deng, and G. Mesnil (2014). A latent semantic model with convolutional-pooling structure for information retrieval,. In *CIKM*.

Smeulders, A. W. M., M. Worring, S. Santini, A. Gupta, and R. Jain (2000, December). Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 1349–1380.

Socher, R., B. Huval, C. Manning, and A. Ng (2012). Semantic compositionality through recursive matrix-vector spaces,. In *EMNLP-CoNLL, pp. 1201-1211.*

Su, H., G. Li, D. Yu, and F. Seide (2013). Error back propagation for sequence training of context-dependent deep neural networks for conversational speech transcription,. In *ICASSP*.

Suchanek, F., G. Kasneci, and G. Weikum (2008). Yago: A large ontology from wikipedia and wordnet. *Web Semant. 6*, 203–217.

Susskind, J., A. Anderson, and G. E. Hinton (2010). The Toronto face dataset. Technical Report UTML TR 2010-001, U. Toronto.

Sutskever, I., O. Vinyals, and Q. Le (2014). Sequence to sequence learning with neural networks. In *NIPS*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological) 58*(1), 267–288.

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In W. W. Cohen, A. McCallum, and S. T. Roweis (Eds.), *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, pp. 1064–1071. ACM.

Torralba, A. (2003). Contextual priming for object detection. *International Journal of Computer Vision 53(2)*, 169–191.

Torralba, A. B., R. Fergus, and Y. Weiss (2008). Small codes and large image databases for recognition. In *CVPR*. IEEE Computer Society.

Tur, G., D. Hakkani-Tur, L. Heck, and S. Parthasarathy (2011). Sentence simplification for spoken language understanding. In *, ICASSP. pp. 5628-5631.*

Tur, G., D. Hakkani-Tür, and L. Heck (2010). What is left to be understood in atis. In *, IEEE SLT.*

Tur, G. and R. D. Mori (2011). Spoken language understanding: Systems for extracting semantic information from speech. In *Eds. John Wiley and Sons.*

Usunier, N., D. Buffoni, and P. Gallinari (2009a, June). Ranking with ordered weighted pairwise classification. In L. Bottou and M. Littman (Eds.), *Proceedings of the 26th International Conference on Machine Learning*, Montreal, pp. 1057–1064. Omnipress.

Usunier, N., D. Buffoni, and P. Gallinari (2009b). Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1057–1064.

Vapnik, V. N. and A. Y. Chervonenkis (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications 16*, 264–280.

Vesely, K., A. Ghoshal, L. Burget, and D. Povey (2013). Sequence-discriminative training of deep neural networks,. In *INTERSPEECH.*

Vincent, P., H. Larochelle, Y. Bengio, and P.-A. Manzagol (2008). Extracting and composing robust features with denoising autoencoders. In W. W. Cohen, A. McCallum, and S. T. Roweis (Eds.), *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, pp. 1096–1103. ACM.

Vincent, P., H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol (2010, December). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. See**?**.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm,. In *IEEE Transactions on Information. Theory, vol. IT-13, pp. 260-269, Apr.*

Vogel, J. and B. Schiele (2004, 7). Natural scene retrieval based on a semantic modeling step. In *Proceeedings of the International Conference on Image and Video Retrieval CIVR 2004, Dublin, Ireland, LNCS*, Volume 3115.

Wang, Y., L. Deng, and A. Acero. Semantic frame based spoken language understanding,. In *Chapter 3, Tur and De.*

Wang, Y., L. Deng, and A. Acero (2005). Spoken language understanding — an introduction to the statistical framework. In *IEEE Signal Processing Magazine, vol. 22, no. 5, pp. 16-31.*

Wang, Y.-Y. and A. Acero (2006). Discriminative models for spoken language understanding,. In *ICSLP.*

Weston, J., S. Bengio, and N. Usunier. Large scale image annotation learning to rank with joint word image embeddings.

Weston, J., S. Bengio, and N. Usunier (2011). Large scale image annotation: Learning to rank with joint word-image embeddings. In *International Joint Conference on Artificial Intelligence (IJCAI).*

Xiao, J., J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba (2010, June). SUN database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3485–3492. IEEE.

Xu, P. and R. Sarikaya (2013). Convolutional neural nteworks based triangular crf for joint intent detection and slot filling,. In *ASRU.*

Yaman, S., L. Deng, D. Yu, Y. Wang, and A. Acero (2008). An integrative and discriminative technique for spoken utterance classification. In *IEEE Transactions on Audio, Speech and Language Processing, vol. 16, no. 6, pp. 1207-1214.*

Yao, ., B. Peng, and F. Gao (2014). Recurrent conditional random field for language understanding,. In *ICASSP, pp. 4105-4009.*

Yao, B., X. Yang, and T. Wu (2009). Image parsing with stochastic grammar: The lotus hill dataset and inference scheme. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009*, pp. 8–8. IEEE.

Yao, B., X. Yang, and S. Zhu (2007). Introduction to a large-scale general purpose ground truth database: methodology, annotation tool and benchmarks. In *Proceedings of the 6th international conference on Energy minimization methods in computer vision and pattern recognition*, pp. 169–183.

Yao, K., B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi (2014). Spoken language understanding using long short-term memory neural networks. In *, IEEE SLT.*

Yao, K., B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao (2013). Recurrent conditional random fields,. In *NIPS Deep Learning Workshop.*

Yao, K., G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu (2013). Recurrent neural networks for language understanding. In *Interspeech.*

Yih, W., X. He, and C. Meek (2014). Semantic parsing for single-relation question answering,. In *ACL.*

Young, M. Henderson, M. G. B. T. P. T. K. Y. S. (2012). Discriminative spoken language understanding using word confusion networks. In *, IEEE SLT.*

Yu, D., S. Wang, and L. Deng (2010). Sequential labeling using deep-structured conditional random fileds,. In *Journal of Selected Topics Signal Processing, vol. 4, no. 6, pp. 965-973.*

Yu, M., T. Zhao, D. Dong, H. Tian, and D. Yu (2013). Compound embedding features for semi-supervised learning,. In *NAACL-HLT.*

Zhou, B., A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva (2014). Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems 27 (NIPS'2014).*

Zhou, Z., D. Zhan, and Q. Yang (2007). Semi-supervised learning with very few labeled training examples. In *Proceedings of the National Conference on Artificial Intelligence*, Volume 22, pp. 675. AAAI Press ; MIT Press ; 1999.

Zue, V. and J. Glass (2000). Conversational interface: advances and challenges. In *, Proceedings of the IEEE, vol. 88, no. 8, pp. 1166-1180*.