Apprentissage profond et théorie de l'information

Léonard Blier

October 30, 2017

1 Introduction

L'apprentissage profond, plus fréquement désigné par l'expression *Deep Learning* est une branche du *machine learning* ou *apprentissage automatique*. Il consiste à apprendre des représentations de haut niveau des données en utilisant des *réseaux de neurones*. Ces méthodes ont été développées dans les années 1980 mais rapidement abandonnées car considérées comme peu prometteuses. Grâce à l'amélioration de la puissance de calcul des ordinateurs, de l'apparition de nouvelles bases de données plus grandes et plus riches, et de grands progrès dans les techniques d'optimisation, le Deep Learning a récemment permis d'obtenir des performances exceptionnelles pour différentes tâches [LeCun et al., 2015], notamment :

- **La classification d'images**: Les *réseaux de neurones convolutionnels* popularisés par le réseau *AlexNet* [Krizhevsky et al., 2012] ont permis d'atteindre des résultats impressionnants en classification d'images. Celui-ci a gagné en 2012 la competition *ImageNet*, ce qui a lancé la grande vague d'intérêt actuelle pour les réseaux de neurones. Les réseaux de convolution ont été successivement améliorés par la suite, notamment avec le *VGGNet* [Simonyan and Zisserman, 2014] et les *Residual Networks* [He et al., 2016].
- **Modèles génératifs:** Grâce au développement des Auto-encodeurs [Kingma and Welling, 2013] et surtout récemment des *réseaux adversariaux* [Goodfellow et al., 2014], on parvient à générer des images voire des sons extrêmement réalistes mais nouveaux, ou de modifier des images (ajouter des lunettes, vieillir ou rajeunir un visage sur une photo, ...)
- **Reconnaissance de la parole**: Ce problème a été l'une des premières applications industrielles du Deep Learning [Hinton et al., 2012]
- **Traitement automatique du langage naturel** pour différents problèmes comme :
  - la réponse automatique à des questions posées en langage naturel, soit en faisant de la recherche dans des données externes [Bordes et al., 2014] ou dans un texte [Hermann et al., 2015], ou encore en devant apprendre à raisonner à partir des informations dans la question [Sukhbaatar et al., 2015].
  - La traduction automatique a également été largement améliorée, où le deep learning a désormais remplacé dans tous les systèmes les plus modernes les méthodes développées par les linguistes fondées sur une fine étude de la grammaire [Sutskever et al., 2014]
  - Description automatique d'images en langage naturel [Xu et al., 2015]
- **Intelligence artificielle dans les jeux :** La victoire très médiatisée de AlphaGo contre Lee Sedol au Go a montré l'efficacité du deep learning dans ce domaine [Silver et al., 2016].

2 Apprentissage automatique et réseaux de neurones

2.1 Apprentissage automatique

Le problème posé en apprentissage automatique est celui de la prédiction. Dans ce qui suit, nous ne considèrerons que le problème de l'apprentissage supervisé. On dispose de données $\mathcal{X} = \{x_1, ..., x_n\} \subset$

E et $\mathcal{Y} = \{y_1, ..., y_n\} \subset F$. On suppose que les données sont les réalisations de variables aléatoires X et Y de loi jointe μ. Le problème est de prédire Y sachant X, autrement dit d'approcher la distribution $\mu(y|x)$. Si F est un espace discret, on dit que le problème est un problème de classification. Sinon, on dit que le problème est un problème de régression.

La démarche consiste à choisir une métrique $\ell : F \times F \to \mathbb{R}^+$ telle que $\ell(y, y) = 0$, puis à *apprendre* un modèle $\nu(y|x)$ qui minimise

$$L(\nu) = \mathbb{E}_X \left[\mathbb{E}_{Y,\hat{Y}} [\ell(Y, \hat{Y})|X] \right] \tag{2.1}$$

où $\hat{Y}$ est une variable aléatoire suivant la loi $\nu(y|x)$ conditionnellement à X.

On utilise une estimation de L :

$$\hat{L}(\nu) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\hat{Y} \sim \nu(.|x_i)} \left[\ell(y_i, \hat{Y}) \right] \tag{2.2}$$

De plus, si on choisit pour un modèle déterministe $\hat{Y} = f(X)$, on obtient :

$$\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) \tag{2.3}$$

Apprendre un modèle signifie choisir un modèle ν dans une famille de modèles $\mathcal{F}$. Le problème est celui de la généralisation. Il est facile de trouver un modèle ν tel que $\hat{L}(\nu) = 0$. Autrement dit, il est facile de trouver un modèle pour un nombre fini de données. La difficulté est de trouver un modèle qui sera capable de généraliser sur de nouvelles données.

Afin d'évaluer la généralisation, on *apprend* le modèle sur une partie des données, et on l'évalue sur une autre partie. Si le modèle est significativement meilleur sur les données d'entrainement que sur les données de test, on dit qu'il surapprend.

Example 1. Le cas que nous prendrons en exemple dans ce mémoire est la classification d'images. Les x_i sont des images, représentées par des vecteurs. Si l'image est en nuance de gris, chaque dimension représente un pixel, et une coordonnée du vecteur représente un niveau de gris sur ce pixel. Si l'image est en couleur, chaque pixel est représenté par trois dimensions, représentant chacun une couleur.

Les bases de données de reconnaissance d'images sur lesquelles les réseaux de neurones ont prouvé leur efficacité sont par exemple `MNIST`, `CIFAR` et `ImageNet`.

2.2 Réseaux de neurones

Les réseaux de neurones sont une méthode d'apprentissage automatique.

Définition 1. Soient $E = E_0, E_1, ..., E_L = F$ et Θ des espaces vectoriels. On note $n_j = dim(E_j)$. Soit $G \subset \{0, 1, ..., L\} \times \{0, 1, ..., L\}$ un graphe orienté acyclique tel qu'il existe un chemin de 0 à L. On définit alors récursivement :

$$\begin{array}{rccc} h_0 : & E_0 \times \Theta & \longrightarrow & E_0 \\ & (x, \theta) & \longmapsto & x \end{array} \tag{2.4}$$

$$\begin{array}{rccc} h_j : & E_0 \times \Theta & \longrightarrow & E_j \\ & (x, \theta) & \longmapsto & \sigma_i \left(b_j(\theta) + \sum_{(i,j) \in G} W_{i,j}(\theta) \cdot h_i(x, \theta) \right) \end{array} \tag{2.5}$$

où :

- $\sigma_i : E_j \to E_j$ est une fonction appliquée à chaque coordonnée d'un vecteur : soit $x \in E_j$, $\sigma_j(x) = (\sigma_j(x_1), ..., \sigma_j(x_{n_j}))$. Ces fonctions sont choisies pour être non-linéaires, et sont presque toujours l'une des fonctions suivantes : $\sigma(x) = \max(0, x) = (x)_+$; $\sigma(x) = \tanh(x)$, $\sigma(x) = \frac{1}{1+e^{-x}}$,
- $W_{i,j}(\theta)$ et $b_j(\theta)$ sont les poids ou les paramètres du réseau,

- les fonctions :

$$b_j : \Theta \to E_j \tag{2.6}$$
$$W_{i,j} : \Theta \to \mathcal{M}_{n_i,n_j}(\mathbb{R}) \tag{2.7}$$

forment une paramétrisation des poids du réseau. Ces fonctions sont choisies différentiables.

Un réseau de neurone est alors la fonction définie par

$$\begin{array}{rccc} f: & E \times \Theta & \longrightarrow & F \\ & (x,\theta) & \longmapsto & h_n(x,\theta) \end{array} \tag{2.8}$$

On appelle *représentation cachée* ou *représentation intermédiaire* les $h_i(x,\theta)$. b_i est appelé le biais, $W_{i,j}$ la partie linéaire et σ_i l'activation.

Le graphe G est appelé la *structure*, l'*architecture* ou la *topologie* du réseau de neurone.

Remarque 1. Très souvent, le graphe définissant la structure du réseau est le simple graphe linéaire $1 \to 2 \to ... \to L$. Dans ce cas, on désigne par *couche* chaque fonction calculant $f_{i+1} : E_i \to E_{i+1}$. Le nombre de couche, ou profondeur du réseau, est alors la distance entre l'entrée et la sortie dans le graphe.

Les paramètres des réseaux de neurones sont appris en utilisant des algorithmes de type *descente de gradient*. On veut minimiser :

$$\hat{L}(f) = L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i,\theta)) \tag{2.9}$$

On commence par choisir une valeur initiale θ_0, et à chaque étape on calcule :

$$\theta_{t+1} = \theta_t - \eta_t \times \partial_\theta \hat{L}(\theta_t) \tag{2.10}$$
$$= \theta_t - \eta_t \times \frac{1}{n}\sum_{i=1}^{n} \partial_\theta \ell(y_i, f(x_i,\theta_t)) \tag{2.11}$$

Ici η_t est appelé le pas de gradient. Les algorithmes utilisés en pratique sont des variations autour de la simple descente de gradient. Notamment, on utilise la *descente de gradient stochastique*, qui consiste à rajouter une composante aléatoire à la procédure en n'évaluant pas le gradient pour chaque élément de la base de donnée, mais seulement en l'estimant sur un petit nombre d'entre eux appelé *mini-batch*. Il existe ensuite différentes variations, notamment Adam ou RMSprop.

2.3 Deep Learning et généralisation

Comme présenté dans l'introduction, le deep learning permet d'obtenir d'excellents résultats en prédiction pour différentes tâches. Pourtant, les capacités de généralisation des réseaux posent plusieurs questions.

Propriété 1. *Soient $n, N \in \mathbb{N}$. Alors il existe un réseau de neurone $f : \mathbb{R}^n \times \Theta \to \mathbb{R}$ à deux couches avec $2n + N$ paramètres tel que pour toutes données $\mathcal{X} = \{x_1, ..., x_n\} \subset \mathbb{R}^N$ et $\mathcal{Y} = \{y_1, ..., y_n\} \subset \mathbb{R}$ il existe θ^* tel que*

$$f(x_i, \theta^*) = y_i \ , \ \forall 1 \leq i \leq n \tag{2.12}$$

De plus, si $k > 2$, il existe un réseau de neurones f à k couches avec $\mathcal{O}(n/k)$ paramètres tel que pour toutes données $\mathcal{X} = \{x_1, ..., x_n\} \subset \mathbb{R}^N$ et $\mathcal{Y} = \{y_1, ..., y_n\} \subset \mathbb{R}$ il existe θ^* tel que

$$f(x_i, \theta^*) = y_i \ , \ \forall 1 \leq i \leq n \tag{2.13}$$

Voir [Zhang et al., 2016] pour plus de détails.

Les nombres de paramètres évoqués étant plus faibles que le nombre de paramètres dans les réseaux de neurones utilisés en pratique, on peut supposer que les réseaux de neurones ont la capacité de sur-apprendre les données d'entraînement.

Ce phénomène peut s'observer en pratique sur de vraies données, dans le cas de la classification d'images [Zhang et al., 2016]. On fait l'expérience suivante. On prend un réseau connu pour donner de très bonnes prédictions lorsqu'il est entrainé sur une base de donnée comme `ImageNet`. On ré-initialise ses poids et on l'entraine cette fois avec la même base de donnée, mais dont les labels ont été permutés aléatoirement. On observe alors que le réseau est capable d'apprendre avec 100% de prédiction (sur les données d'entrainement) à classifier avec ces faux labels, alors qu'il n'y a pourtant aucune structure sous-jascente à apprendre. Ceci montre que les réseaux qui pourtant fonctionnent en pratique ont pourtant la capacité de sur-apprendre les données.

De plus, le cas des *exemples adversariaux* montrent que les capacités de généralisation des réseaux de neurones sont à tempérer. Toujours dans le cas de la classification d'image, si l'on prend une image classifiée correctement par un réseau, il est possible de la perturber, c'est-à-dire de lui ajouter un bruit imperceptible à l'oeil nu, qui suffit à faire changer la prédiction du réseau [Szegedy et al., 2013]. Ces images modifiées sont appelées des exemples adversariaux. En dehors du fait qu'ils peuvent provoquer des inquiétudes pour l'utilisation des systèmes de reconnaissance utilisés en pratique, notamment dans les voitures sans conducteurs, ces exemples adversariaux montrent que, si les réseaux de neurones semblent bien généraliser, ils ne le font pas de la même manière que l'humain.

2.4 Approches théoriques du deep learning

Plusieurs travaux théoriques ont tenté d'expliquer les résultats des réseaux de neurones. Je vais présenter ici succintement quelques directions de recherche sur ce sujet.

La première question qui s'est posée est celle de l'expressivité des réseaux de neurones, c'est-à-dire l'ensemble des fonctions pouvant être représentées par un réseau de neurone. Le premier résultat, souvent cité comme un argument en faveur des réseaux de neurones est le suivant : l'ensemble des réseaux de neurones à 2 couches (1 couche cachée) dont les non-linéarités sont des fonctions sigmoid est dense pour la norme $\|.\|_\infty$ dans l'ensemble des fonctions continues de de $[0,1]^n$ dans $\mathbb{R}$ [Cybenkot, 1989]. Toutefois, la construction des réseaux de neurones approchant des fonctions continues réalisée pour prouver ce théorème ne ressemble pas aux réseaux utilisés en pratique. En effet, les réseaux construits sont beaucoup plus *larges* que ceux utilisés habituellement, c'est-à-dire que la dimension de la couche cachée devient très importante. D'autres travaux étudient les capacités d'approximation des réseaux en faisant évoluer leur profondeur plutôt que leur largeur [Shaham et al.,][Mhaskar, 2016], et ceci est un problème ouvert.

Les réseaux de neurones appris étant le résultat d'un problème d'optimisation non convexe, différents travaux ont tenté d'étudier les propriétés de la fonction minimisée, en utilisant ses nombreuses symétries. Notamment, ils ont tenté d'expliquer pourquoi la procédure d'optimisation convergeait presque toujours vers des minima qui, bien que distincts, atteignaient des valeurs pour la fonction minimisée très proches. Certains travaux ont montré que, en prenant un modèle de réseau de neurone simplifié, *presque tous* les minima locaux de la fonction minimisée étaient proches du minimum global [Lu and Kawaguchi, 2017] [Choromanska et al., 2014]. De plus, différents travaux ont tenté de montrer que les minima correspondant à des solutions qui vont généraliser sur de nouvelles données ont des propriétés différentes de ceux qui correspondent à du *sur-apprentissage*. En particulier, que les bassins d'attraction des minima qui généralisent seraient plus *larges* que ceux des minima qui ne généralisent pas [Wu and Zhu, 2017]. Ceci pourrait être un élément expliquant pourquoi un algorithme comme la descente de gradient stochastique semble converger vers des minima qui généralisent.

Dans la même prespective, beaucoup de question sont actuellement posées autour de la procédure d'optimisation en elle-même. On constate empiriquement que les différents paramètres (pas de gradient, taille du sous-ensemble des données sur lequel un pas de gradient est calculé, méthode d'optimisation, ainsi que d'autres astuces très souvent utilisés en Deep Learning comme le *dropout* ou la *batch-normalisation*) ont une très grande influence sur les résultats de l'apprentissage. Différents travaux assez descriptifs tentent d'étudier l'impact de ces paramètres et d'en déduire des informations

sur l'apprentissage des réseaux de neurones en général [Keskar et al., 2016][Zhang et al., 2016].

Dans une perspective différente, certains tentent d'utiliser la théorie de l'information pour comprendre la façon dont fonctionnent les réseaux de neurones. En particulier, ils utilisent le principe de l'*information bottleneck* pour étudier les représentations cachées calculées par les réseaux de neurones, et montrer que ces représentations *éliminent* à chaque couche l'information qui n'est pas nécessaire à la tâche de classification pour ne garder que la quantité d'information minimale [Shwartz-Ziv and Tishby,][Achille and Soatto, 2017].

Enfin, on peut citer des méthodes venant du traitement du signal, cherchant à partir des invariants géométriques connus des données, comme les invariances par translation et déformations dans le cas des images, pour étudier la façon dont les réseaux de neurones peuvent apprendre ces invariants [Mallat, 2016].

Ces différentes approches sont très récentes, et ne permettent pour l'instant ni de comprendre l'efficacité des réseaux de neurones, ni de dépasser leurs limites.

3 Complexité de Kolmogorov et induction de Solomonoff

Les principales sources pour cette section sont [Bensadon, 2016] and [**?**].

3.1 Raisonnement inductif

La complexité de Kolmogorov est un outil puissant qui peut être la fondation d'une théorie générale de l'induction. Qu'est-ce que le raisonnement inductif ? Le raisonnement inductif est le mode de raisonnement qui permet de déduire des lois générales à partir de cas particuliers. Il est bien sûr à la base de toutes les sciences expérimentales, mais est aussi utilisé au quotidien lorsque nous adaptons nos choix à ce que nous pensons que serons les conséquences futures. Comment ce mode de raisonnement peut-il être formalisé ? Prenons un exemple concret. Supposons qu'on vous présente la séquence

`010101010101x`

où `x` est un chiffre caché que vous devez deviner. Que proposeriez-vous ? Il semble que `0` soit la solution la plus raisonnable. Pourquoi ? Une explication possible est que nous cherchons l'explication la plus simple qui explique les premiers chiffres observés. Cette explication est sans aucun doute que la suite est composé d'un motif simple, `01`, répété un certain nombre de fois. On suppose alors que cette explication est la bonne, et on s'en sert pour faire de la prédiction.

Cette méthode est souvent appelée le *rasoir d'Occam*, et se formulerait de la façon suivante : Il faut toujours choisir le modèle le plus simple qui permet d'expliquer les observations. La complexité de Kolmogorov permet de formaliser cette notion, en considérant que la simplicité d'un modèle est la longueur du plus court programme qui permet de le calculer.

3.2 Définitions

Dans la suite, on note $\mathcal{A} = \{0, 1\}$ l'alphabet binaire, $\mathcal{A}^n = \mathcal{A} \times ... \times \mathcal{A}$ l'ensemble des mots de longueur n sur $\mathcal{A}$, et $\mathcal{A}^* = \cup_{n\in\mathbb{N}}\mathcal{A}^n$ l'ensemble de tous les mots de longueur finie sur $\mathcal{A}$.

La complexité de Kolmogorov se définit à partir d'une machine de Turing universelle. Nous ne présenterons ici ce concept que superficiellement, mais peu de connaissances sur le sujet sont nécessaires. Une *machine de Turing* est un modèle de système de calcul. Il est composé d'une *bande* infinie sur laquelle peuvent être écrits des symboles de l'alphabet $\mathcal{A}$ (ici l'alphabet binaire) et d'une *tête de lecture* qui peut se déplacer sur la bande, lire les caractères de la bande, et écrire sur la bande. La tête de lecture peut être dans un certain nombre (fini) d'*états*. Ses actions sont déterminées à chaque étape du calcul par l'état dans lequel elle se trouve et le caractère qu'elle lit sur la bande. Toute fonction calculable peut être calculée par une machine de Turing. Une *machine de Turing universelle* est une machine de Turing U capable de simuler toutes les machines de Turing. Autrement dit, pour toute machine de Turing T, il existe une certaine chaîne telle que si elle est écrite sur la bande de U, alors U calculera la même chose que T.

Moins formellement, une machine de Turing est un modèle pour l'execution d'un programme informatique. Une machine de Turing est un modèle pour un ordinateur. Un ordinateur peut (théoriquement, si on oublie les contraintes physiques) calculer toutes les fonctions calculables : il suffit de mettre dans sa mémoire le programme qui permet de calculer cette fonction.

Définition 2 (Complexité de Kolmogorov)**.** La complexité de Kolmogorov $\mathcal{K}_U(x)$ d'une séquence $x \in \{0,1\}^*$ sur une machine de Turing universelle U est la longueur du plus court programme $\mathtt{p}$ sur U tel que le programme p s'arrête, et que sa sortie est x. Cette quantité se mesure en bits.

Le principal résultat, qui justifie la définition, est que la complexité de Kolmogorov ne dépend pas, à une constante près, de la machine de Turing sur laquelle elle est calculée.

Théorème 1. *Si U et U' sont deux machines de Turing universelles, et x une séquence binaire, alors $|\mathcal{K}_U(x) - \mathcal{K}_{U'}(x)| \leq C_{U,U'}$ où $C_{U,U'}$ est une constante qui ne dépend pas de x.*

Démonstration : On présente le principe de la preuve. Si U et U' sont deux machines de Turing universelles, alors il existe un programme I sur U' qui prend en entrée un programme $\mathtt{p}$ sur U et dont la sortie est un programme $\mathtt{p}'$, qui lancé sur U' a exactement la même sortie que $\mathtt{p}$ sur U. Ce programme peut être considéré comme un interpréteur d'un langage à un autre.

Alors si on a un programme $\mathtt{p}$ sur U qui encode x, le programme $\mathtt{p}' = I :: \mathtt{p}$ (où :: désigne la concaténation) va d'abord interpréter $\mathtt{p}$ en $\mathtt{p'}$, puis executer $\mathtt{p'}$ dont la sortie est x. Donc $K_{U'}(x) \leq |I| + K_U(x)$. L'argument étant symétrique, on a notre résultat.

Définition 3 (Complexité de Kolmogorov conditionnelle)**.** Si x et y sont deux séquences binaires, on définit la complexité de Kolmogorov conditionnelle de y sachant x notée $\mathcal{K}(y|x)$ comme la longueur du plus court programme $\mathtt{p}$ tel que $\mathtt{p}(x) = y$.

Toutes les bornes étant définies à une constante près, on notera $\mathcal{K}(x) \leq f(x)$ à la place de $\mathcal{K}(x) \leq a + f(x)$, a ne dépendant pas de x.

La complexité de Kolmogorov semble être un outil puissant pour la compression de données en pratique. Mais le résultat suivant empêche de la calculer en pratique.

Théorème 2. *La complexité de Kolmogorov de $x \in \mathcal{A}^*$ n'est pas calculable.*

Par conséquent, nous devrons nous contenter de calculer des bornes sur la complexité de Kolmogorov.

3.3 Bornes génériques

Bien que la complexité de Kolmogorov ne soit pas calculable, il est possible d'obtenir des bornes de compression. On peut tout d'abord donner des bornes génériques pour tout entier ou toute séquence binaire.

Propriété 2. *Si n est un entier, on a la borne :*

$$\mathcal{K}(n) \leq 2 \log_2 n. \tag{3.1}$$

Démonstration : On définit $\mathtt{c_0}(n) = 11...10$, composé de n 1,suivis par un zéro. Cet encodage est une version prefixe de l'encodage en base 1.

Si l'écriture binaire de x est $\mathtt{b}$, on définit $\mathtt{c_1}(n) = \mathtt{c_0}(|b|) :: b = 11...10b$. Ceci forme bien un encodage préfixe. On a alors :

$$\mathcal{K}(n) \leq |\mathtt{c_0}(|\mathtt{b}|) + |b| = 2|b| = 2 \log_2 n. \tag{3.2}$$

Remarque 2. On peut affiner cette propriété pour obtenir la borne $\mathcal{K}(n) \leq \log_2 n + 2\log_2 \log_2 n$.

Propriété 3. *Soit x une séquence binaire. On a la borne:*

$$\mathcal{K}(x) \leq |x| + 2\log_2 |x| \tag{3.3}$$

Démonstration : On peut encoder la séquence x, et la faire précéder d'un encodage de sa longueur, afin que le code soit préfixe.

3.4 Compression et modèles

Nous allons voir comment avoir un *modèle* des données permet d'obtenir des bornes de compression.

3.4.1 Théorie de l'information

La théorie de Shannon [Shannon, 1948], initialement développée pour les télécommunications, permet d'avoir des bornes précises sur les longueurs de codes nécessaires pour définir un élément dans un ensemble muni d'une mesure de probabilité. Cela va nous permettre d'obtenir de nouvelles bornes de compression utilisant des modèles.

Définition 4 (Entropie)**.** Soit μ une mesure sur un ensemble E. On définit l'entropie de μ comme :

$$H(\mu) = \mathbb{E}[-\log_2 \mu(X)] = \sum_{x \in E} -\mu(x)\log_2(\mu(x)). \tag{3.4}$$

Définition 5 (Divergence de Kullback-Leibler)**.** Soient μ et ν deux mesures de probabilités sur E. On définit la divergence de Kullback Leibler entre μ et ν comme :

$$D_{KL}(\mu \| \nu) = \mathbb{E}_\mu \left[\log_2 \left(\frac{\mu(X)}{\nu(X)} \right) \right] = \sum_x \mu(x) \log_2 \frac{\mu(x)}{\nu(x)}. \tag{3.5}$$

La divergence de Kullback Leibler n'est pas une distance car elle n'est pas symétrique. Mais elle vérifie la propriété suivante : elle est positive, et nulle si et seulement si $\mu = \nu$.

3.4.2 Codage et théorie de Shannon

Le problème de Shannon était de définir un encodage prefixe d'éléments d'un ensemble, de telle sorte que les codes soient les plus courts posible. Un encodage des éléments d'un ensemble E sur un alphabet $\mathcal{A}$ est défini comme une injection $S : E \to \mathcal{A}^*$ telle que $\{S(x), x \in E\}$ soit un ensemble préfixe. Le problème est de minimiser $\mathbb{E}[|S(X)|] = \sum_x \mu(x)|S(x)|$. On a alors le théorème suivant :

Théorème 3 (Théorème de Shannon)**.** *Soit S un encodage préfixe. On a*

$$\mathbb{E}[|S(X)|] \geq H(\mu). \tag{3.6}$$

De plus, il existe un encodage S_μ calculable tel que

$$-\log_2 \mu(x) \leq S_\mu(x) \leq -\log_2 \mu(x) + 1, \tag{3.7}$$

et par conséquent :

$$H(\mu) \leq \mathbb{E}_\mu[S_\mu(X)] \leq H(\mu) + 1. \tag{3.8}$$

Démonstration : Le premier résultat est bien connu, et se démontre à l'aide de l'inégalité de Kraft. Le second peut se construire par exemple avec le codage de Huffman.

3.4.3 Codes de Shannon et compression de Kolmogorov

Définition 6 (Complexité d'une mesure de probabilité)**.** Soit E un ensemble, et μ une mesure sur E. On définit $\mathcal{K}(\mu)$ comme la longueur du plus court programme $\mathtt{p}_\mu$ tel que $\mathtt{p}_\mu(x) = \mu(x)$. Cette grandeur peut éventuellement être infinie si μ n'est pas calculable.

Le théorème suivant [Li and Vitányi, 2008] est le théorème principal de cette approche, et montre le lien entre la complexité de Kolmogorov et l'apprentissage statistique.

Théorème 4. *Soit E un ensemble et μ une mesure sur E. On a la borne :*

$$\mathcal{K}(x) \leq \mathcal{K}(\mu) - \log_2 \mu(x) \tag{3.9}$$

L'équation (3.9) mérite que l'on s'y attarde. La borne est composée de deux parties. La seconde est bien connue, il s'agit de la vraisemblance du modèle μ étant données les données x. Minimiser ce terme est ce qui est habituellement fait si on se limite à une famille paramétrique de modèles et que l'on cherche le maximum de vraisemblance. Toutefois, il est clair que le minimum de ce terme sur l'ensemble de tous les modèles est 0, car on peut prendre $\mu = \delta_x$ où δ_x est la mesure de Dirac en x. Le premier terme mesure la complexité du modèle. Il joue ici le rôle de régulariseur. Par conséquent, minimiser l'équation (3.9) revient à trouver un compromis entre complexité du modèle, et adéquation du modèle aux données.

Démonstration : On sait que si S est un encodage prefixe de E, on a la borne de compression suivante :

$$\mathcal{K}(x) \leq \mathcal{K}(S) - |S(x)|. \tag{3.10}$$

En effet, on peut compresser x en utilisant cet encodage, et en stockant le programme permettant de le décoder. Or on sait qu'on a un encodage prefixe S tel que $|S(x)| \leq -\log_2 \mu(x) + 1$, ce qui nous permet d'obtenir la borne suivante :

$$\mathcal{K}(x) \leq \mathcal{K}(S) - \log_2 \mu(x) + 1. \tag{3.11}$$

La complexité de Kolmogorov étant définie à une constante additive près, on écrit simplement :

$$\mathcal{K}(x) \leq \mathcal{K}(\mu) - \log_2 \mu(x). \tag{3.12}$$

3.4.4 Principe de la description minimale

On peut utiliser la notion de compression de Kolmogorov pour définir une méthode de sélection de modèle :

Définition 7 (Principe de la description minimale)**.** Soient $x_1, ..., x_n$ des données. Le modèle μ^* choisit par le *principe de la description minimale* est le modèle qui minimise : $\mathcal{K}(\mu) - \sum_i \log_2 \mu(x_i)$

Cette définition correspond bien à une formalisation du rasoir d'Occam présenté précédemment. On a alors le théorème suivant :

Théorème 5. *Soit $X_1, ..., X_n, ...$ une suite de variable aléatoire identiquement distribuées et indépendantes de loi μ sur un espace discret, où μ est une mesure calculable. Soient $\nu_1, ..., \nu_n, ...$ la suite de modèles obtenus par le principe de la description minimale, (ie ν_n minimise $\mathcal{K}(\nu) - \sum_{i=1}^n \log_2 \nu(x_i)$).*
Alors $D_{KL}(\mu, \nu_n) \to 0$.

Démonstration : On sait que ν_n minimise :

$$\mathcal{K}(X_1, ..., X_n) \leq \mathcal{K}(\nu) + \sum_i - \log_2 \nu(X_i) \tag{3.13}$$

En particulier, on a :

$$\mathcal{K}(\nu_n) + \sum_{i=1}^{n} -\log_2 \nu_n(X_i) \leq \mathcal{K}(\mu) + \sum_{i=1}^{n} -\log_2 \mu(X_i) \tag{3.14}$$

On en déduit que :

$$\sum_{i=1}^{n} \log_2 \frac{\mu(X_i)}{\nu_n(X_i)} \leq \mathcal{K}(\mu) - \mathcal{K}(\nu_n) \tag{3.15}$$

En utilisant la loi des grands nombres, on a presque sûrement :

$$D_{\mathrm{KL}}(\mu\|\nu) \leq \frac{1}{n}\left(\mathcal{K}(\mu) - \mathcal{K}(\nu_n)\right) + o(1) \leq \frac{1}{n}\mathcal{K}(\mu) + o(1) \to 0 \tag{3.16}$$

3.5 Théorie de Solomonoff de l'induction

La théorie de Solomonoff de l'induction permet non seulement de faire de la prédiction, mais aussi de l'inference. En définissant un *prior universel* sur les modèles, elle propose une théorie *universelle* de l'induction.

Définition 8 (Prior universel)**.** On définit le prior universel sur les distributions de probabilité calculables par :

$$\alpha(\mu) = \frac{1}{C} 2^{-\mathcal{K}(\mu)} \tag{3.17}$$

où $C = \sum_\mu 2^{-\mathcal{K}(\mu)}$.

Pour justifier que cette définition est cohérente, il faut montrer que $\sum_\mu 2^{-\mathcal{K}(\mu)} < \infty$, nous ne le démontrerons pas ici.

On peut alors définir la distribution de probabilité jointe :

$$P(x, \mu) = \alpha(\mu)\mu(x) = \frac{1}{C} 2^{-\mathcal{K}(\mu)} \mu(x). \tag{3.18}$$

On peut alors remarquer que la selection de modèle avec le principe de la description de longueur minimale est équivalente à choisir μ qui maximise $P(x, \mu)$. En effet, maximiser $P(x, \mu)$ est équivalent à minimiser $-\log_2 P(x, \mu) = \mathcal{K}(\mu) - \log_2 \mu(x)$. Par conséquent, la méthode de sélection de modèle avec le principe de la description de longueur minimale est équivalente à la méthode du *maximum a posteriori* (MAP) avec pour prior le prior universel sur les modèles.

Définition 9 (Distribution de probabilité universelle)**.** On peut définir la distribution de probabilité universelle sur les séquences de $\mathcal{A}^*$.

$$P(x) \propto \sum_{\mu \text{ probability distribution}} 2^{-\mathcal{K}(\mu)} \mu(x) \tag{3.19}$$

On peut ainsi utiliser cette distribution pour faire de la prédiction, en utilisant la distribution :

$$p(x_{n+1}) = P(x_{n+1}|x_1, ..., x_n) \tag{3.20}$$

$$= \frac{P(x_1, ..., x_{n+1})}{P(x_1, ..., x_n)} \tag{3.21}$$

Ainsi, la théorie de l'induction de Solomonoff est une théorie générale de l'induction. Elle est équivalente à une méthode d'inférence bayesienne classique avec pour prior le prior universel sur les modèles. Cela permet d'apporter une solution au problème du choix du prior.

4 Bornes de compression et réseaux de neurones

Dans cette partie, je vais présenter les grandes lignes de mon mémoire de M2.

La théorie de la compression de Kolmogorov et le principe de la description de longueur minimale semblent se heurter à une contradiction avec les réseaux de neurones. En effet, ces modèles sont les plus efficaces en prédiction, et pourtant, ils semblent aussi être les plus complexes. Leur nombre de paramètres dépasse très largement le nombre d'éléments dans la base de données. Par conséquent, ils semblent ne pas compresser les données, et ne seraient donc pas choisis par une procédure de sélection de modèle du type principe de la description de longueur minimale.

Nous allons présenter différentes façons d'obtenir des bornes de compression à l'aide des réseaux de neurones, et plus généralement de modèles paramétriques. L'exemple que nous prendrons pour illustrer le propos sera la classification d'image sur les bases de données suivantes :

- **MNIST** est une base de données de 60,000 images de taille 28×28 en niveaux de gris, représentant les chiffres manuscrits de 0 à 9. Le but est de classifier ces images. Cette tâche est considérée comme relativement facile, puisqu'une simple méthode des k plus proches voisins permet d'obtenir un résultat de 95% en classification. Les meilleurs modèles, des réseaux de neurones, permettent d'atteindre une performance de $99,5\%$. Le but est donc ici de comprimer les labels $\mathcal{Y}_{MNIST}$ sachant les images $\mathcal{X}_{MNIST}$. La compression triviale en utilisant le modèle uniforme sur les labels permet de comprimer en $60,000 \times \log_2 10 = 199$kbits.
- **CIFAR10** est une base de données de 50,000 photographies de taille 32×32 en couleur dont les 10 classes sont soit des animaux (vache, chien, ...) soit des moyens de transport (voiture, train). Le but est encore de classifier ces images. Cette tâche est considérée comme plus difficile. Les meilleurs modèles, encore obtenus avec des réseaux de neurones, obtiennent des résultats en classification de 93%. Le but est de comprimer les labels $\mathcal{Y}_{CIFAR}$ sachant les images $\mathcal{X}_{CIFAR}$. La compression triviale en utilisant le modèle uniforme sur les labels permet de comprimer en $50,000 \times \log_2 10 = 166$kbits.

4.1 Borne de compression triviale

On note dans la suite $p_\theta(y|x)$ le modèle décrit par un réseau de neurone de paramètre θ. On a la borne de compression suivante :

$$\mathcal{K}(x) \leq \mathcal{K}(p_\theta|\theta) + \mathcal{K}(\theta) - \log_2 p_\theta(\mathcal{Y}|\mathcal{X}) \tag{4.1}$$

$\mathcal{K}(p_\theta|\theta)$ est le coût d'encodage du programme qui, étant donné le paramètre θ permet de calculer le modèle p_θ. En pratique, ce coût est le coût d'encodage du fichier de code qui décrit le réseau de neurone. Ce fichier étant de quelques dizaines de lignes, on peut supposer que ce terme ne dépasse pas quelques centaines de bits.

Le dernier terme est connu, et réprésente l'adéquation du modèle aux données. Dans le cas d'un très bon modèle, ce terme peut ne pas dépasser quelques centaines de bits.

Le second terme est le plus important. Il encode les paramètres du réseau. En pratique, il correspond au fichier stockant tous les poids. Un réseau de neurone en classification d'image pouvant avoir plusieurs millions de paramètres, tous stockés sur 32 bits, ce coût peut être a priori de plusieurs dizaine de mégabits !

Par conséquent, la borne de compression triviale donnée par le modèle uniforme donne une borne de compression plus de 100 fois inférieure à celle obtenue avec un réseau de neurone, pourtant bien meilleur en prédiction. Nous allons présenter différentes manières de résoudre ce problème.

4.2 Méthodes pratiques de compression de réseaux

Une méthode de compression utilisée en pratique consiste à modifier légèrement un réseau de neurone entrainé afin de réduire le coût de stockage des paramètres[Han et al., 2015]. La première possibilité consiste à mettre à 0 tous les paramètres proches de 0 afin de rendre le modèle parcimonieux. Ensuite,

on peut diminuer drastiquement la précision donnée sur les poids, et les stocker non pas sur 32 bits mais sur quelques bits seulement.

Ces méthodes sont utilisées pour transmettre des réseaux de neurones sur des petites machines, comme des téléphones portables par exemple, ou pour fabriquer des puces très rapides utilisées en reconnaissance d'images, dans les voitures sans conducteurs par exemple. Elles permettent de diviser par 50 le coût d'encodage des paramètres, sans pour autant perdre significativement en qualité de prédiction.

Toutefois, ces modèles compressés pèsent encore plusieurs Mb, voire plusieurs dizaines de Mb, ce qui est au moins 10 fois plus élevé que la borne de compression obtenue avec le modèle uniforme. Nous allons présenter d'autres méthodes de calcul de bornes de compression, moins utiles en pratique, mais plus proches de décrire la quantité minimale d'information nécessaire pour reconstruire les labels à partir des images.

4.3 Bornes de compression variationelles

On choisit un prior α sur Θ l'ensemble des paramètres. Le principe de la borne variationnelle consiste non pas à encoder un seul paramètre θ mais une distribution de probabilité $q \in \mathcal{P}(\Theta)$ [Hinton and Van Camp,]. On obtient la borne suivante :

Théorème 6 (Borne variationnelle)**.** *Soit $p_\theta(y|x)$ un modèle paramétrique, $\theta \in \Theta$. Soit α un prior sur Θ, et q une distribution de probabilité sur Θ. On a alors :*

$$\mathcal{K}(x) \leq \mathcal{K}(p_\theta|\theta) + D_{KL}(q\|\alpha) + \mathbb{E}_{\theta\sim q}\left[-\log_2 p_\theta(\mathcal{Y}|\mathcal{X})\right] \tag{4.2}$$

Example 2. Soit θ^* le paramètre d'un réseau pré-entraîné. On peut décider non pas de stocker θ^* mais la distribution q uniforme sur le pavé $[\theta_1^* - \frac{\varepsilon}{2}, \theta_1^* + \frac{\varepsilon}{2}] \times ... \times [\theta_N^* - \frac{\varepsilon}{2}, \theta_N^* + \frac{\varepsilon}{2}]$ où $\varepsilon > 0$. Cela revient à coder θ^* avec une précision ε pour la norme $\|.\|_\infty$. Cela permet de réduire le coût d'encodage du modèle. Pour la prédiction, on choisit aléatoirement $\theta \sim q$ et on utilise ce paramètre. Ceci explique le dernier terme de l'equation (**??**).

En pratique on choisit $q \in \mathcal{Q}$ où $\mathcal{Q} = \{\mathcal{N}(\mu, \Sigma), \mu \in \mathbb{R}^n, \Sigma \text{ diagonale}\}$ et un prior gaussien sur les poids.

Comme on peut le constater dans le tableau 1, la borne variationnelle est significativement inférieure à la borne triviale, et les modèles variationnels permettent tout de même d'obtenir de bons résultats en classification. Toutefois, elles présentent l'inconvénient de sélectionner des modèles moins bons en prédiction que les meilleurs réseaux de neurones. En particulier, elles poussent à choisir des modèles plus petits.

4.4 Bornes de compression online

Une autre façon de compresser les labels est de le faire de façon online. On choisit $0 = t_0 < t_1 < ... < t_S = n$ où n est le nombre d'éléments de la base de données. L'algorithme est le suivant :

1. On encode un modèle paramétrique (réseau de neurone) $\theta \to p_\theta$ et un algorithme d'apprentissage déterministe `OPT`, tel que $\texttt{OPT}(y_1, ..., y_k|x_1, ..., x_k) = \theta$ minimise $-\log_2 p_\theta(y_1, ..., y_k|x_1, ..., x_k)$. `OPT`.
2. On encode les données $y_{t_0}, y_{t_0+1}, ..., y_{t_1-1}$ avec le modèle uniforme.
3. $s \to 1$
4. On entraîne un modèle avec les données déjà encodées avec `OPT`. Autrement dit, on définit $\theta_s = \texttt{OPT}(y_{t_0}, y_{t_0+1}, ..., y_{t_s-1}|x_{t_0}, x_{t_0+1}, ..., x_{t_s-1})$.
5. On encode les données $y_{t_s}, y_{t_0+1}, ..., y_{t_{s+1}-1}$ avec le modèle p_{θ_s}, ce qui coûte :

$$\sum_{i=t_s}^{t_{s+1}-1} -\log_2 p_{\theta_s}(y_i|x_i) \tag{4.3}$$

6. $s \leftarrow s+1$

7. Si $s = S$, on s'arrête. Sinon on retourne à l'étape 3.

On a alors la borne de compression suivante :

Théorème 7 (Borne de compression online)**.** *L'algorithme de compression online fournit la borne suivante :*

$$\mathcal{K}(\mathcal{Y}|\mathcal{X}) \leq \mathcal{K}(p_\theta|\theta) + \mathcal{K}(\texttt{OPT}) + \sum_{s=1}^{S} \sum_{i=t_s}^{t_{s+1}-1} -\log_2 p_{\theta_s}(y_i|x_i) \tag{4.4}$$

L'efficacité de cette borne dépend du nombre d'échantillons à partir duquel le modèle utilisé a de bonnes performances. En pratique, on constate que les réseaux de neurones généralisent très bien à partir de seulement quelques centaines d'exemples pour MNIST, quelques milliers pour CIFAR. Par conséquent, les bornes online permettent d'obtenir d'excellents taux de compression, largement supérieurs à ceux obtenus avec les méthodes variationnelles. De plus, les modèles sélectionnées par cette méthode s'avèrent être également ceux qui donnent les meilleurs résultats en prédiction.

Méthode	$\mathcal{K}$(MNIST)	MNIST acc	$\mathcal{K}$(CIFAR)	CIFAR (acc)
Modèle uniforme	199 kb	10%	166 kb	10%
Borne triviale	$\approx$ 100 kb	99,5%	$\approx$ 500kb	93%
Compression de réseau	$\approx$ 3 kb	$\approx$ 99%	$\approx$ 15kb	$\approx$ 92%
Borne variationnelle	23,9 kb	95,5%	89,0 kb	61,6%
Borne Online	4,09 kb	99,5%	45,4 kb	93%

Table 1: Résumé des bornes de compression obtenues avec des méthodes de Deep Learning. $\mathcal{K}$(DATASET) est le cout d'encodage des labels de la base de donnée étant données les images. DATASET acc est le score en classification du modèle ayant atteint cette borne. Les scores précédés du symbole $\approx$ ne sont pas exacts mais sont des ordres de grandeur. Leurs bornes de compression étant largement supérieures à la borne obtenue avec le modèle uniforme, nous ne les avons pas calculées précisément.

5 Conclusion

La théorie de la complexité de Kolmogorov et de l'induction de Solomonoff donnent un théorie générale du raisonnement inductif. Les liens entre cette théorie et l'approche traditionnelle de l'apprentissage statistique sont donc très éclairants. Un objectif pourrait être d'approcher le prior de Solomonoff. Les réseaux de neurones semblent alors un outil approprié, car leur structure très générale semble pouvoir décrire l'ensemble des modèles possibles. La question de l'évaluation de la complexité des réseaux de neurones, ainsi que des bornes de compression obtenues à l'aide de ces méthodes. Dans le mémoire de M2 succintement présenté ici, nous avons montré empiriquement que les réseaux de neurones, bien qu'ils possèdent un très grand nombre de paramètres, permettent d'obtenir d'excellents résultats en compression. La méthode de compression online utilisée permet de voir que le modèle donnant les meilleurs résultats en prédiction est aussi celui qui donne les meilleurs résultats en compression.

References

[Achille and Soatto, 2017] Achille, A. and Soatto, S. (2017). On the Emergence of Invariance and Disentangling in Deep Representations.

[Bensadon, 2016] Bensadon, J. (2016). *Applications of Information Theory to Machine Learning.* PhD thesis, Paris Saclay.

[Bordes et al., 2014] Bordes, A., Weston, J., and Chopra, S. (2014). Question Answering with Subgraph Embeddings. *arXiv preprint arXiv:1406.3676.*

[Choromanska et al., 2014] Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2014). The Loss Surfaces of Multilayer Networks.

[Cybenkot, 1989] Cybenkot, G. (1989). Mathematics of Control, Signals, and Systems Approximation by Superpositions of a Sigmoidal Function*. *Math. Control Signals Systems*, 2:303–314.

[Goodfellow et al., 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*.

[Han et al., 2015] Han, S., Mao, H., and Dally, W. J. (2015). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv preprint arXiv:1510.00149*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[Hermann et al., 2015] Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching Machines to Read and Comprehend.

[Hinton et al., 2012] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[Hinton and Van Camp,] Hinton, G. E. and Van Camp, D. Keeping Neural Networks Simple by Minimizing the Description Length of the Weights.

[Keskar et al., 2016] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv preprint arXiv:1609.04836*.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436—-444.

[Li and Vitányi, 2008] Li, M. and Vitányi, P. (2008). *An introduction to Kolmogorov complexity*. Springer.

[Lu and Kawaguchi, 2017] Lu, H. and Kawaguchi, K. (2017). Depth Creates No Bad Local Minima.

[Mallat, 2016] Mallat, S. (2016). Understanding Deep Convolutional Networks.

[Mhaskar, 2016] Mhaskar, H. (2016). Learning Functions: When Is Deep Better Than Shallow.

[Shaham et al.,] Shaham, U., Cloninger, A., and Coifman, R. R. Provable approximation properties for deep neural networks.

[Shannon, 1948] Shannon, C. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27.

[Shwartz-Ziv and Tishby,] Shwartz-Ziv, R. and Tishby, N. Opening the Black Box of Deep Neural Networks via Information.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.

[Sukhbaatar et al., 2015] Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-To-End Memory Networks.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*.

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

[Wu and Zhu, 2017] Wu, L. and Zhu, Z. (2017). Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes.

[Xu et al., 2015] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.

[Zhang et al., 2016] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.