



HAL
open science

Synthetic, Adversarial and Natural Corruptions: Image Classifier Robustness Transfers From one Distribution Shift to an Other

Alfred Laugros

► **To cite this version:**

Alfred Laugros. Synthetic, Adversarial and Natural Corruptions: Image Classifier Robustness Transfers From one Distribution Shift to an Other. Signal and Image processing. Université Grenoble Alpes [2020-..], 2022. English. NNT: 2022GRALT016 . tel-03702340

HAL Id: tel-03702340

<https://theses.hal.science/tel-03702340v1>

Submitted on 23 Jun 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **SIGNAL IMAGE PAROLE TELECOMS**

Arrêtée ministériel : 25 mai 2016

Présentée par

Alfred LAUGROS (doctorant)

Thèse dirigée par **Alice CAPLIER (directeur)**
et coencadrée par **Matthieu OSPICI (co-encadrant)**

préparée au sein du **Laboratoire Grenoble Images Parole Signal Automatique**
dans **L'École Doctorale Electronique, Electrotechnique, Automatique, Traite-
ment du Signal (EEATS)**

**Synthetic, Adversarial and Natural Corruptions :
Image Classifier Robustness Transfers From one
Distribution Shift to an Other**

**Corruptions Synthétiques, Adversaires et Naturelles :
Transferts de Robustesse des Classifieurs d'Images
d'un Changement de Distribution à un Autre**

Thèse soutenue publiquement le **21/03/22**,
devant le jury composé de :

Monsieur Olivier MICHEL

Grenoble INP, Président

Madame Catherine ACHARD

Université Sorbonne, Rapporteur

Monsieur Patrick BOUTHEMY

INRIA Rennes, Rapporteur

Monsieur Julien MAIRAL

INRIA Grenoble, Examineur

Madame Alice CAPLIER

Grenoble INP, Directrice de thèse

Monsieur Matthieu OSPICI

Atos, Co-Encadrant de thèse



Abstract. The unprecedented high performances of artificial neural networks in various computer vision tasks, have drawn the interest of both academic and industrial actors. Indeed, neural networks have shown promising results when trying to detect tumors on x-ray images or reporting aggressive behaviors on video screens for instance.

However, neural network performances largely shrink when they face corrupted images. For example, neural networks are sensitive to backlights, shot noises or even patterns maliciously introduced so as to disturb them.

To address this issue, we generally make sure that models are robust to various corruptions before deploying them. Moreover, it is assumed that ensuring the robustness of neural networks to a set of diverse corruptions during their conception, should also make them robust to the unforeseen corruptions they may encounter afterwards. Unfortunately, this assumption does not always hold, i.e., robustness to the corruptions considered during model conception, may not transfer to the corruptions met after being deployed. For instance, a model guaranteed to be robust to brightness changes in images, can still be sensitive to variations in daylight.

Actually, we do not really know which corruptions should be considered, when evaluating robustness during model conceptions, in order to increase their chances to be robust to the distribution shifts they may encounter when deployed. Therefore, to address this issue, we propose in this thesis to study the circumstances under which the robustness to one corruption transfers to another. In other words, we would like to know which robustness guarantee, is provided by making sure that a model is robust to some specific corruptions.

In tackling this problem, we first demonstrate that hacker-like adversarial corruptions and hand-coded synthetic corruptions, are not correlated in terms of robustness. This result means that robustnesses to these two kinds of corruptions, are independent attributes, which should be both considered when attempting to design robust models. Then, we propose a data augmentation strategy called M-TLAT, able to increase robustness to these two aspects of neural network robustness simultaneously. As an interesting aside, the results obtained using M-TLAT, show that ensuring neural network robustness to a few corruptions, can make them robust to a large range of other distribution shifts.

Secondly, we establish a categorization of synthetic corruptions, where robustness transfers from one corruption to another within each category. On the other hand, corruptions from different categories are not correlated in terms of robustness. Establishing these categories show that some of the widely used synthetic corruption benchmarks, extensively test neural network robustness towards some specific kinds of corruptions, to the detriment of other kinds which are largely omitted. To overcome this, we finally provide recommendations to build less biased benchmarks. Interestingly, robustness to benchmarks built following our guidelines, largely transfer to naturally arising corruptions, which can be particularly useful in production context.

Résumé. Les réseaux de neurones artificiels ont obtenu des performances sans précédent dans de nombreuses tâches de vision par ordinateur. Ils ont ainsi attiré l'attention des acteurs industriels et académiques. En effet, des réseaux de neurones ont par exemple des taux de réussite extrêmement prometteurs en détection de tumeurs cancéreuses ou dans l'identification de comportements agressifs sur des images vidéo.

Cependant, les performances des réseaux chutent drastiquement lorsqu'ils rencontrent des images dites corrompues. Ils sont par exemple sensibles aux contre-jours, aux bruits de grenaille et même à des motifs trompeurs pouvant être introduits spécifiquement pour les perturber.

Pour lutter contre cette vulnérabilité, on cherche généralement à rendre les réseaux de neurones robustes à diverses corruptions avant de les déployer. L'idée étant que des modèles certifiés comme étant robustes à diverses corruptions durant leur phase de conception, ont plus de chances d'être robustes aux corruptions qu'ils rencontreront ensuite. Malheureusement, cette intuition n'est pas toujours vérifiée, car la robustesse certifiée en amont vis-à-vis de certaines corruptions, n'est pas toujours représentative de la robustesse aux corruptions survenant après déploiement. Par exemple, un modèle certifié comme étant robuste aux changements de luminosité, n'est pas nécessairement robuste aux variations de l'ensoleillement.

En réalité, nous ne savons pas vraiment quelles corruptions considérer lors d'une mesure de robustesse, pour fournir aux réseaux une garantie vis-à-vis des corruptions qu'ils pourraient rencontrer ensuite. Pour combler cette lacune, nous proposons dans cette thèse d'étudier dans quels cas, le fait d'être robuste à une corruption implique d'être robuste à une autre. En d'autres termes, on aimerait savoir quelle garantie sur la robustesse d'un réseau a-t-on, si l'on sait que celui-ci est robuste à des corruptions données.

Pour apporter des éléments de réponse, nous commençons par démontrer que la robustesse aux corruptions adversaires, qui exploitent la structure interne des réseaux, n'est pas corrélée à la robustesse aux corruptions synthétiques codées à la main. Ces deux aspects de la robustesse des réseaux sont indépendants. Puis, nous proposons une méthode de data augmentation nommée M-TLAT, capable d'améliorer la robustesse des réseaux à ces deux types de corruptions simultanément. Ces résultats montrent par ailleurs, que certifier la robustesse à quelques corruptions complexes bien choisies, peut augmenter la robustesse des réseaux à un grand ensemble de perturbations.

Dans un deuxième temps, nous établissons une nouvelle catégorisation des corruptions synthétiques, de telle manière que le fait d'être robuste à une corruption d'une catégorie donnée, implique d'être robuste aux corruptions appartenant à cette même catégorie. A l'inverse, les corruptions appartenant à des catégories différentes ne sont pas corrélées en termes de robustesse. Ces catégories montrent en l'occurrence, que des benchmarks de corruptions synthétiques dont l'usage est largement répandu, donnent trop d'importance au fait d'être robuste à certaines corruptions ; négligeant ainsi la robustesse à d'autres types de corruptions. En définitive, pour résoudre cela, nous fournissons des recommandations visant à construire des benchmarks moins biaisés. Il est par ailleurs intéressant de noter que les benchmarks construits en suivant nos critères, fournissent des estimations tout à fait prédictives de la robustesse des réseaux aux corruptions dites naturelles ; ce qui peut être particulièrement utile dans un contexte industriel.

Remerciements

Je tiens ici à adresser mes remerciements aux personnes m'ayant apporté leur soutien durant ces trois dernières années. Sans eux, la réalisation de cette thèse aurait été une expérience nettement plus pénible.

Je tiens d'abord à remercier ma directrice de thèse Alice, pour sa supervision bienveillante, son écoute, son ouverture d'esprit et sa disponibilité. Merci également à mon autre encadrant Matthieu, pour son suivi serein et accommodant, ainsi que pour les discussions que nous avons eues qui m'ont souvent poussé à prendre du recul. Je les remercie tous les deux pour la grande liberté qu'ils m'ont accordée tout au long de ma thèse, et j'insiste sur ce point qui était absolument essentiel pour moi.

Merci à ma compagne Elisa, pour son soutien indéfectible durant ces trois années. Elle m'a notamment permis de me rééquilibrer dans les moments où les problématiques de mes travaux devenaient obsessionnelles. Elle m'a apporté une reconnaissance immense, que j'ai eue tant de mal à trouver autrement durant cette thèse. Sans elle, je me serais probablement senti diminué.

Je remercie également avec ardeur mes parents, particulièrement pour les heures passées à m'écouter clarifier des pensées confuses à voix haute. Ils m'ont par ailleurs apporté leur soutien sur tant d'autres aspects, que je ne pourrais énumérer exhaustivement en quelques lignes.

Merci à mon frère Arthur, pour m'avoir fait prendre conscience de ma valeur par certains mots forts prononcés aux bons moments. Aussi, il m'a secoué lorsque je manifestais trop de négativité. Moins anecdotique qu'il n'y paraît, il a fait preuve d'un grand enthousiasme à m'écouter débâter des opinions franchement tranchées sur l'intelligence artificielle, que beaucoup d'autres auraient trouvées absolument inaudibles.

Je remercie également Lucas, pour m'avoir régulièrement fait sortir de ma caverne, en m'entraînant dans des week-ends généralement mémorables. J'aurais été bien trop fainéant pour les organiser moi-même, ce qui est idiot, car ces moments avec tous ces gens très amusants, font partie des meilleurs que l'on puisse passer.

Enfin merci à Mathias, de s'intéresser à mes travaux jusqu'au point de vouloir en assurer une certaine continuité lors de sa propre thèse. Son enthousiasme fait plaisir à voir et me rend impatient d'assister à la suite.

Table of Contents

| | |
|---|-----------|
| Remerciements | i |
| Table of Contents | iii |
| 1 Introduction | 1 |
| 1.1 Neural Network Sensitivity to Distribution Shifts | 2 |
| 1.2 Increasing Neural Network Robustness | 3 |
| 1.3 Providing Guarantees Using Robustness Specifications | 4 |
| 1.4 Problematic : The Assumption of Transferability Between Corruption Robustnesses | 5 |
| 1.5 Organization | 6 |
| 2 Background : Addressing Robustness to Adversarial, Synthetic and Natural Corruptions | 9 |
| 2.1 Definitions | 10 |
| 2.1.1 Adversarial Attacks | 10 |
| 2.1.2 Synthetic Corruptions | 11 |
| 2.1.3 Natural Corruptions | 11 |
| 2.2 Robustness Interventions | 12 |
| 2.2.1 Addressing Robustness to Adversarial Attacks | 12 |
| 2.2.2 Addressing Robustness to Synthetic Corruptions | 13 |
| 2.2.3 Addressing Robustness to Natural Corruptions | 13 |
| 2.3 Robustness Benchmarks | 15 |
| 2.3.1 Robustness Metrics | 15 |
| 2.3.2 Measuring Robustness to Adversarial Attacks | 18 |
| 2.3.3 Measuring Robustness to Synthetic Corruptions | 21 |
| 2.3.4 Measuring Robustness to Natural Corruptions | 22 |
| 2.4 Bridging the Gap Between Adversarial, Synthetic and Natural Corruption Robustnesses | 24 |
| 3 On Interaction Between Adversarial Corruption Robustness and Synthetic Corruption Robustness | 26 |
| 3.1 Introduction | 27 |
| 3.1.1 Features Extracted by Adversarially Trained Models Align with Human Perception | 27 |
| 3.1.2 Trade-off Between Clean Accuracy and Adversarial Robustness | 27 |
| 3.1.3 Organization | 28 |
| 3.2 Correlation Between Adversarial and Synthetic Corruption Robustnesses | 28 |
| 3.2.1 Introduction | 28 |
| 3.2.2 Experimental Set-up | 29 |
| 3.2.3 Does Adversarial Corruption Robustness Transfer to Synthetic Corruptions? | 31 |
| 3.2.4 Does Synthetic Corruption Robustness Transfer to Adversarial Attacks? | 32 |
| 3.2.5 Concurrent Works | 33 |

| | | |
|----------|--|-----------|
| 3.2.6 | Conclusion | 34 |
| 3.3 | Addressing Both Adversarial and Synthetic Corruption Robustnesses using M-TLAT. | 35 |
| 3.3.1 | Introduction | 35 |
| 3.3.2 | Combining Mixup with Targeted Labeling Adversarial Training : M-TLAT | 35 |
| 3.3.3 | Experimental Set-up | 38 |
| 3.3.4 | Results | 40 |
| 3.3.5 | Concurrent Works | 43 |
| 3.3.6 | Conclusion | 44 |
| 3.4 | Conclusion | 45 |
| 4 | Measuring Robustness Using Synthetic Corruptions | 47 |
| 4.1 | Introduction | 48 |
| 4.1.1 | Synthetic Corruptions : Convenient Proxies to Study Robustness to Unforeseen Distribution Shifts | 48 |
| 4.1.2 | Synthetic Corruption Selections and Biases in Benchmarks | 48 |
| 4.1.3 | Organization | 49 |
| 4.2 | Using the Overlapping Score to Reveal Flaws in Synthetic Corruption Benchmarks | 49 |
| 4.2.1 | Introduction | 49 |
| 4.2.2 | The Corruption Overlapping Score | 50 |
| 4.2.3 | Experimental Set-up | 51 |
| 4.2.4 | Overlappings in ImageNet-C | 52 |
| 4.2.5 | Corruption Overlappings and Benchmark Balance | 53 |
| 4.2.6 | Corruption Overlappings and Benchmark Coverage | 53 |
| 4.2.7 | Concurrent Works | 54 |
| 4.2.8 | Conclusion | 56 |
| 4.3 | Building Synthetic Corruption Benchmarks More Predictive of Robustness to Natural Corruptions | 57 |
| 4.3.1 | Introduction | 57 |
| 4.3.2 | Experimental Set-up | 58 |
| 4.3.3 | Corruption Categories | 60 |
| 4.3.4 | Synthetic Corruption Selection Criteria | 63 |
| 4.3.5 | Comparisons with Existing Synthetic Corruption Benchmarks | 66 |
| 4.3.6 | Discussions | 66 |
| 4.3.7 | Conclusion | 67 |
| 4.4 | Conclusion | 68 |
| 5 | Conclusion | 70 |
| 5.1 | Summary and Contributions | 71 |
| 5.2 | Further Works | 72 |
| 5.3 | Publications | 73 |
| | Table des figures | 75 |
| | Liste des tableaux | 78 |
| | Bibliographie | I |

1

Introduction

Contents

| | | |
|------------|--|----------|
| 1.1 | Neural Network Sensitivity to Distribution Shifts | 2 |
| 1.2 | Increasing Neural Network Robustness | 3 |
| 1.3 | Providing Guarantees Using Robustness Specifications | 4 |
| 1.4 | Problematic : The Assumption of Transferability Between Corruption Robustnesses | 5 |
| 1.5 | Organization | 6 |

1.1 Neural Network Sensitivity to Distribution Shifts

Neural Networks have been shown to be very efficient in various computer vision tasks such as image classification [Tan2019], face recognition [Taigman2014] or object detection [Ren2015]. One striking achievement are the first neural networks surpassing human-level performances on the ImageNet validation set [He2015]. Several results suggest that neural networks could be immensely useful in a lot of applications such as transport [Huval2015], medicine [Işın2016], etc.

However, most of neural network achievements are observed when they are tested on independent and identically distributed (i.i.d.) samples with regard to their training sets [Taigman2014], [Ren2015], [He2016]. In other words, neural networks perform well when test images are gathered using the exact same procedure as the one used to get training images. In this case, test and train samples are said to be drawn from the same distribution.

Unfortunately, neural networks are sensitive to distribution shifts [Geirhos2020]. A model is facing a distribution shift when it encounters images that are obtained using a different procedure than the one used to get its training images. For instance, samples extracted from Google Image in 2010, are drawn from a different distribution than images extracted from Qwant Image in 2020. The consequence of distribution shifts is that test images have features that are statistically different from the ones of training images. In that kind of situations, test images are said to be out-of-distribution (o.o.d). The performances obtained when facing a distribution shift are generally much lower than the ones obtained on i.i.d samples.

Unfortunately, images encountered by neural networks when deployed in real-world applications are often out of their training distributions. For example, depending on the weather, the season and the location, an autonomous vehicle may encounter a huge diversity of situations (this idea is illustrated in Figure 1.1). Besides, when unexpected events occur, such as meeting a pedestrian disguised at a carnival, the autonomous car is supposed to handle correctly the situation even if it has never seen a costume. When it comes to medicine applications, we would like neural networks to detect tumors on x-ray images whatever the used detector.



Figure 1.1 – Illustration of the distribution shift phenomenon. Scenes arising in the real-world can be drastically different from the ones seen during trainings of neural networks. Upper images are extracted from [Cordts2016].

Then, making neural networks more robust to distribution shifts is absolutely necessary in a lot of application cases. A model is said robust to a distribution shift, when its performances on considered out-of-distribution (o.o.d) samples are close to its performances obtained with trainings

images. For example, a neural network that has the same behavior at any hour of the day or night is robust to illumination changes.

Two important tools, respectively presented in the next Sections 1.2 and 1.3, are generally used to tackle the issue of neural network sensitivity to distribution shifts. These are robustness interventions and robustness specifications.

1.2 Increasing Neural Network Robustness

To make neural networks more reliable, especially in real-world applications, we need to make them more robust to corruptions. In this thesis, the term of corruption refers to what causes a distribution shift. For instance, backlights in video surveillance applications are corruptions : they make models encounter out-of-distribution samples. Making the distinction between distribution shifts and corruptions is not essential in our works, the two terms are largely interchangeable throughout the manuscript.

Various robustness interventions have been proposed to increase neural network robustness. Namely, using non-conventional architectures [Vasconcelos2020], adding new images to training data [Zhang2018], leveraging alternative losses [Kornblith2020] or different optimizers [Metz2019], can increase model robustness to various corruptions.

A robustness intervention is generally proposed to address a specific category of distribution shifts. A category of distribution shifts includes more or less corruptions. For example, a robustness intervention can be proposed to make models more robust to noises [Zhou2017], while others can be established to address robustness to synthetic corruptions in general [Hendrycks2020b] (including noises, rotations, contrast loss, compression artifacts, etc.). Here, the latter category of distribution shifts includes more corruptions than the former.

Of course, the narrower a range of corruptions, the easier it is to propose a robustness intervention to address this range. For instance, a model can be made fully robust to forty-five degrees image rotations, by being trained with data augmentation (an illustration of the data augmentation process is provided in Figure 1.2). In this extreme case, the range of considered corruptions corresponds to one corruption only, and it is easy to make models robust to it.

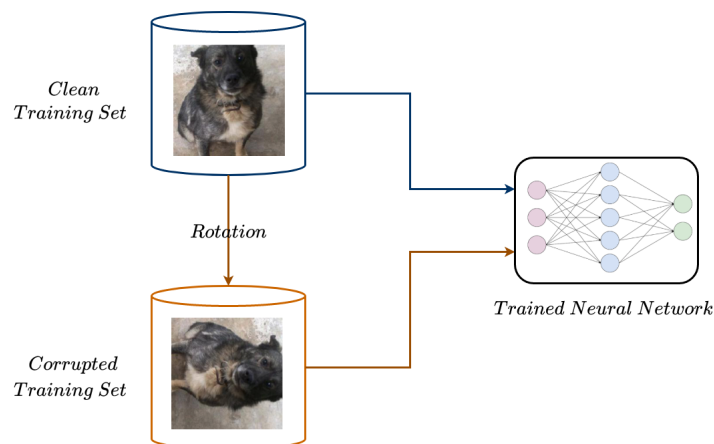


Figure 1.2 – Illustration of the data augmentation process. A duplicate of a model training set is corrupted with a corruption c (here a ninety degrees rotation). The considered neural network is then trained on both a clean training set and a corrupted one. Once trained, the model is robust to c .

At the opposite, the hardest case is when no assumption about the kind of corruptions addressed is made : it can be a jpeg compression at a specific compression rate, misleading patterns added by hackers, etc. This situation is encountered in some industrial applications, where you do not have any information about the distribution shifts that may arise. For instance, when it comes to autonomous vehicles, environments that may be encountered are numerous. They can vary because of the weather, the location, the hour of the day, the traffic density, presence of adversarial examples, replacements of some sensors, etc. These corruptions are so numerous that it is virtually impossible to identify all of them and address them precisely. In this situation, it is required to propose methods that address robustness in a broad sense, i.e., methods that make neural networks less sensitive to distribution shifts in general.

In between the two cases mentioned above, any range of corruptions may be considered : L_p bounded adversarial examples [Ross2018], geometric transformations [Engstrom2019], etc. In the industry, ranges of corruptions addressed largely depend on the kind of applications. Robustness interventions used in the context of autonomous vehicles address a much larger range of corruptions than the ones used in the case of a neural network deployed in a specific factory at a precise production line.

It is important to note that the range of corruptions addressed should be defined a priori, before designing any robustness intervention. Obviously, addressing robustness to noises will not lead to the same kinds of robustness interventions than the ones got when addressing robustness to view angle changes. Basically, the threat must be identified before trying to fight it. As described in the next Section 1.3, the range of corruptions addressed is generally defined when establishing robustness specifications.

1.3 Providing Guarantees Using Robustness Specifications

Robustness specifications are conditions to be satisfied by models to be considered robust. In production context, no model can be deployed while the defined specifications are not met. For instance, a tumor detector can be required to reach a false negative rate lower than humans when dealing with images captured using new sensors, i.e., x-ray sensors that have just been released. Robustness specifications are made of three components which are : (1) a robustness metric, (2) a set of out-of-distribution images and (3) a performance threshold.

- . The robustness metric is a process that provided a model and corrupted samples, estimates the robustness of the model towards the considered corruption. In the previous example, the used metric is the false negative rate. It computes detector predictions on corrupted images, and then compares the obtained results with the labels of these images (presence or not of a tumor). Then, the metric gives the percentage of wrong predictions made, which provides some information about detector robustness to sensor changes.
- . The out-of-distribution images (or corrupted images), are samples that are statistically different from training samples of considered neural networks. In the provided example, corrupted images are samples captured using x-ray sensors different from the ones used to get training images of considered tumor detectors. We note that these first two components taken together (a robustness metric and a set of corrupted images), form what is generally called a robustness benchmark.
- . The performance threshold corresponds to the minimum performances models should reach to be considered robust. In our example, the performance threshold is fixed considering the success rate of humans when facing x-ray images captured with a new sensor.

Obviously, these three components vary depending the application case. For instance, various

robustness metrics can be used depending on the needs. Besides, in academic contexts, the performance threshold generally used, is the highest performances obtained by state-of-the-art models. Basically, newly proposed models should be better than existing ones. Lastly, out-of-distribution images are gathered depending on the category of corruptions considered.

Overall, the aim of robustness specifications, is to provide the neural networks that meet them some robustness guarantees. A tumor detector meeting the specifications mentioned above, is not likely to become obsolete with the arrival of new x-ray sensors. Unfortunately, the guarantees provided by using robustness specifications are in general very limited. As described in the following Section 1.4, they generally do not prevent model performances to unexpectedly shrink because of unforeseen circumstances.

1.4 Problematic : The Assumption of Transferability Between Corruption Robustnesses

The process generally followed to obtain models robust to specific corruptions can be schematized by Figure 1.3. The process consists in repeating three steps until some robustness specifications are met. When this occurs, the obtained model is considered robust and can be deployed in the real-world. However, models obtained this way can still encounter unforeseen corruptions, i.e., corruptions not considered during the three steps schematized in Figure 1.3.

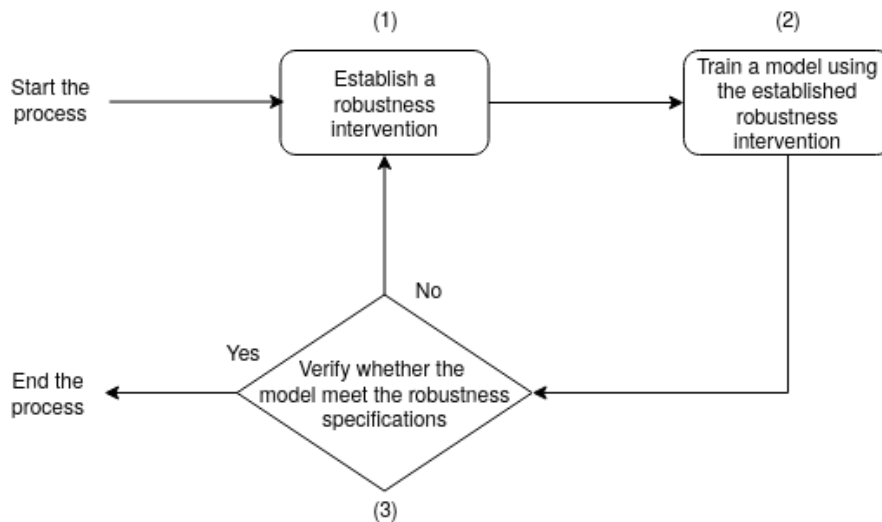


Figure 1.3 – Diagram illustrating the standard procedure followed when trying to build robust neural networks.

Formally, robustness interventions used in step (1), are known to make models robust to a set of precise corruptions we note set_1 . For instance, a data augmentation with cutmix is known to make models more robust to some corruptions of ImageNet-C [Yun2019]. The robustness estimation phase (step (3)) guarantees that models are robust to another set of corruptions we call set_2 . For example, the used robustness specifications can ensure that models are robust to translations and rotations. Unforeseen corruptions are corruptions that do not belong to set_1 nor set_2 . By definition, the model conception process described in Figure 1.3, does not provide any direct robustness guarantee towards these corruptions. Therefore, the trust given to models deployed in real-world applications, is based on the assumption that the robustness to the set_1 and set_2 corruptions transfer

to most unforeseen corruptions a model may encounter after being deployed. The idea is that a model robust to a lot of corruptions, should be also robust to other unknown corruptions.

Unfortunately, this assumption does not always hold and overconfident robustness assertions are often made. For example in academic research, new defenses are regularly claimed to protect against any adversarial attacks (artificial corruptions generated by hackers), while they are only tested using a narrow range of attacks. Of course, these defenses are most of the time shown to be weak a few months later [Athalye2018a]. In the industry, neural networks are often tested on too restricted ranges of corruptions, which bring a very limited guarantee on model robustness. For instance, an AI project from Google Health has been recently aborted because models failed to generalize to unforeseen environments [Talby2020].

To address this issue, we argue that it would be valuable to be able to predict when the robustness towards some corruptions is likely to transfer to others. Can we assume a model robust to rotations to be robust to other geometric transformations such as shear distortions? Is a model robust to several synthetic blurs likely to be robust to the natural blurs encountered in real-world applications? Can we expect a model robust to brightness changes to be robust to night-day transitions? Being able to make that kind of predictions, would help to build robustness specifications that provide robustness guarantees more likely to transfer to new corruptions. In concrete terms, we could build specifications that reduce chances of meeting harmful unforeseen corruptions in practical applications.

Unfortunately, intuition is not a good predictor in this matter : humans are not good at determining which corruptions are correlated in terms of robustness. For instance, increasing robustness to “camera shake blur” can increase neural network robustness to defocus blur, but other blurs are not correlated in terms of robustness [Vasiljevic2016]. Then, human perception cannot consistently predict in which situations the robustness to one corruption is likely to transfer to an other.

Besides, current knowledge about corruption correlation in terms of robustness is very limited : only robustness correlations between very precise corruption couples have been established. For instance, we know that robustness to a corruption called PGD [Madry2018] is correlated with robustness towards the so-called FGSM corruption [Goodfellow2015]. We also know that models robust to various synthetic corruptions can have very poor performances on some natural corruptions [Geirhos2019]. Making a model robust to noises does not make it robust to geometric transformations [Engstrom2019]. These rules are very specific and limited. In practice we can almost never assess a priori, whether two corruptions are correlated in terms of robustness. Consequently, we propose in this thesis to address this limitation.

1.5 Organization

In this thesis, we propose to establish general rules about correlations in terms of robustness between corruptions. Moreover, we want to study the circumstances under which the robustness to one corruption transfers to an other.

The works of this thesis are conducted using the categorization that splits corruptions into three great groups called adversarial, synthetic and natural corruptions. These categories of corruptions are widely studied by the community, and we provide a precise definition for each of them in Section 2.1. Besides, we respectively present in Section 2.2 and 2.3, the existing robustness interventions and robustness benchmarks, generally used to address robustness to these corruptions.

Then, we propose to establish links between robustnesses towards corruptions belonging to same category : either adversarial, synthetic or natural. We also study robustness correlations between these categories : correlation between synthetic corruption robustness and natural corruption robustness for example.

More precisely, in Chapter 3, we focus on adversarial and synthetic corruptions. In particular, we investigate the links between adversarial and synthetic corruption robustnesses in Section 3.2. Leveraging the drawn conclusions, we build in Section 3.3 a data augmentation process that can make models robust to a large range of both adversarial and synthetic corruptions.

In Chapter 4, we study how synthetic corruptions can be used to make accurate estimations of neural network robustness. At this occasion, we propose in Section 4.2, a metric to determine to what extent two synthetic corruptions are correlated in terms of robustness. Besides, we demonstrate that this metric can be used to show that traditionally used synthetic corruption benchmarks are biased, i.e., they give too much importance to the robustness to some corruptions while omitting some others. We address this issue in Section 4.3, by proposing a new methodology to build less biased benchmarks. In addition, we show that synthetic corruption benchmarks built using our methodology make robustness estimations more predictive of robustness to natural corruptions.

Overall, our contributions enable to better identify which corruptions are correlated in terms of robustness. Moreover, these works may help to build better robustness specifications, which when met, provide a stronger guarantee on neural network robustness to unforeseen corruptions.

2

Background : Addressing Robustness to Adversarial, Synthetic and Natural Corruptions

Contents

| | | |
|------------|--|-----------|
| 2.1 | Definitions | 10 |
| 2.1.1 | Adversarial Attacks | 10 |
| 2.1.2 | Synthetic Corruptions | 11 |
| 2.1.3 | Natural Corruptions | 11 |
| 2.2 | Robustness Interventions | 12 |
| 2.2.1 | Addressing Robustness to Adversarial Attacks | 12 |
| 2.2.2 | Addressing Robustness to Synthetic Corruptions | 13 |
| 2.2.3 | Addressing Robustness to Natural Corruptions | 13 |
| 2.3 | Robustness Benchmarks | 15 |
| 2.3.1 | Robustness Metrics | 15 |
| 2.3.2 | Measuring Robustness to Adversarial Attacks | 18 |
| 2.3.2.1 | Selecting Adversarial Examples to Include in Benchmarks | 18 |
| 2.3.2.2 | Examples of Adversarial Attacks | 20 |
| 2.3.3 | Measuring Robustness to Synthetic Corruptions | 21 |
| 2.3.3.1 | Selecting Synthetic Corruptions to Include in Benchmarks | 21 |
| 2.3.3.2 | Examples of Synthetic Corruption Benchmarks | 21 |
| 2.3.4 | Measuring Robustness to Natural Corruptions | 22 |
| 2.3.4.1 | Selecting Natural Corruptions to be Considered when Measuring Robustness | 22 |
| 2.3.4.2 | Examples of Natural Corruption Benchmarks | 23 |
| 2.4 | Bridging the Gap Between Adversarial, Synthetic and Natural Corruption Robustnesses | 24 |

2.1 Definitions

We identify three great categories of corruptions called adversarial attacks [Szegedy2014], synthetic corruptions [Dodge2016] and natural corruptions [Koh2021]. The vast majority of corruptions can be clearly split into these groups, which are widely studied by the community.

Each of these groups provides its own view angle for understanding neural network robustness. Indeed, as detailed in Chapters 3 and 4, neural networks tend to behave differently depending on the kind of corruptions encountered : adversarial, synthetic or natural. Therefore, these three kinds of corruptions can also be seen as different opportunities to better understand neural network functioning.

2.1.1 Adversarial Attacks

Adversarial attacks are corruptions crafted by humans so as to fool neural networks, without changing the semantic content of corrupted images [Biggio2013], [Szegedy2014]. Most of them are barely noticeable to humans. An illustration of such attacks is provided in Figure 2.1. Adversarial attacks have been shown to be particularly harmful in computer vision tasks [Kurakin2017a]. Most of adversarial examples are crafted by leveraging gradients or outputs of neural networks [Kurakin2017b]. In other words, by studying a model functioning, one can craft images that make this model have unexpected behaviors.

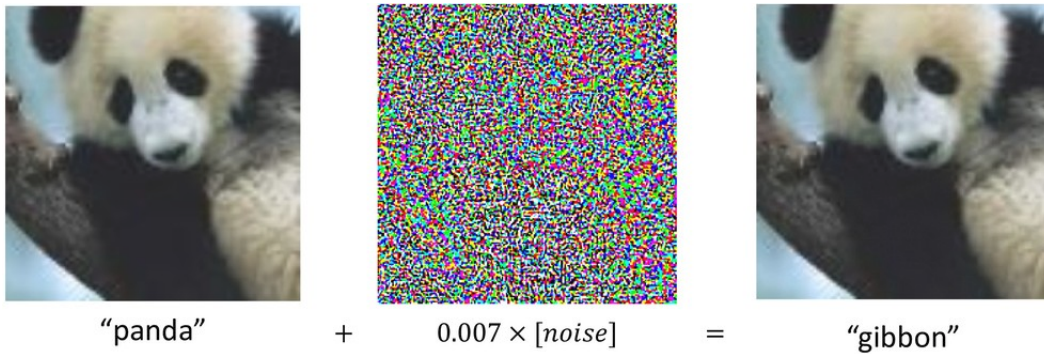


Figure 2.1 – Scheme of a traditional adversarial example crafting (extracted from [Goodfellow2015]).

Adversarial examples can be extremely harmful : tiny perturbations in images can deceive most of state-of-the art neural networks [Szegedy2014]. Obviously, they raise security concerns, we cannot let autonomous vehicle circulate if depraved hackers can stick adversarial patterns on a stop sign that make neural networks perceive a green light [Eykholt2018].

Fortunately, neural networks are much harder to attack in real-world applications compared to controlled environments used in academic research [Lu2017]. Concerning the example provided above, an autonomous vehicle may perceive a green light from a specific viewpoint with precise lightning conditions. However it would still perceive a stop sign in any other conditions. In academic studies, adversarial examples are precise and effective because they are crafted and tested in fixed environments, but adversarial examples are less harmful in production contexts because environments are numerous and diverse.

We still note that some experiments succeed in crafting adversarial examples that affect some neural network behaviors in less controlled environments [Athalye2018b]. To summarize, it is difficult to determine to what extent adversarial examples can be a serious threat to neural networks deployed in production contexts.

Anyway, the vulnerability of models to such transformations raises serious questions about the way neural networks perceive the world. Namely, adversarial sensitivity revealed that models use

patterns to make their predictions that we do not want them to use [Ilyas2019]. In other words, addressing adversarial robustness is not only useful in terms of security, it also helps to better understand the way existing artificial neural networks work, and to identify the direction we should head to build better ones.

2.1.2 Synthetic Corruptions

A synthetic corruption is an image transformation modeled by humans that changes the appearance of images without changing their semantic content [Hendrycks2019a]. On contrary to adversarial attacks, there are not especially crafted to fool neural networks, their implementation generally does not involve neural networks. Synthetic corruptions usually correspond to image transformations traditionally used in computer vision applications such as translations, brightness variations, hue changes, etc. Examples of synthetic corruptions are provided in Figure 2.2.

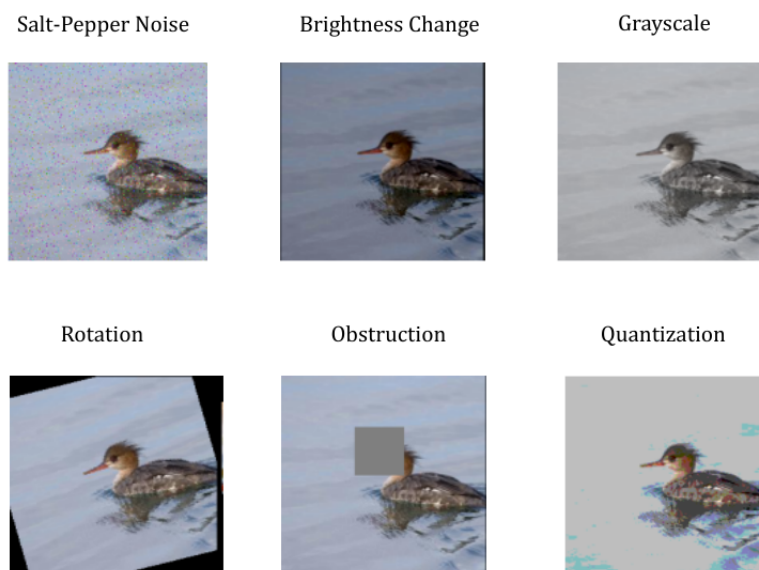


Figure 2.2 – Examples of synthetic corruptions.

Synthetic corruptions may arise in practical applications because of some digital transformations used in image processing chains such as bit quantizations, resolution changes, crops, image compression, etc. Neural networks have been shown to be sensitive to some synthetic corruptions such as noises [Koziarski2017], blurs [Vasiljevic2016], rotations, translations [Azulay2019], colorimetry changes [Karahana2016], compression artifacts [Ghosh2018], etc. In production contexts, image preprocessings used with deployed neural networks are not always known a priori. So, there is a need to build neural networks that are robust to the digital image transformations that may arise in image processing chains.

Similarly to adversarial examples, synthetic corruptions make apparent that neural networks have a superficial "understanding" of the true underlying notions we want them to learn. An image of a rotated dog still represents a dog, and a model should not answer that it represents a truck. Moreover, it would be conceptually interesting to clearly identify the reasons why neural networks are so sensitive to these distribution shifts.

2.1.3 Natural Corruptions

Natural corruptions are corruptions that naturally arise without any human interfering [Taori2020]. On contrary to adversarial examples and synthetic corruptions, these corruptions are too complex

to be modeled. Natural corruptions generally change a large range of features in images. For instance, a neural network trained on images occurring in European cities, is facing naturally corrupted images when deployed in an African city. Indeed, the considered model is facing unexpected weather conditions, sun expositions, city architecture, pedestrian clothes, etc. Another example is given in Figure 2.3, illustrating the situation that may be encountered by a re-identification system. We see that naturally corrupted images can have an aspect largely different from training images.

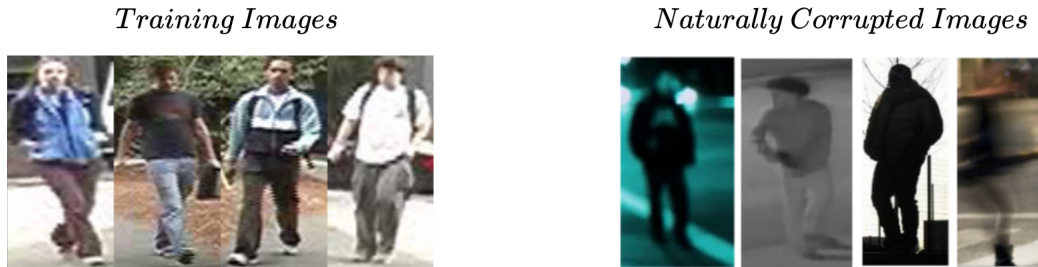


Figure 2.3 – Illustration of natural corruption effects : training images (extracted from the VI-PeR dataset [Gray2007]) have features that are statistically different from features of naturally corrupted images.

Images encountered in production contexts are almost always captured in a different environment than images used to train neural networks. In other words, natural corruptions are most of the time impossible to avoid in practical applications. Besides, similarly to adversarial attacks and synthetic corruptions, natural corruptions are really harmful to neural networks [Beery2018], [Zech2018]. Consequently, they represent a major obstacle preventing neural networks from being considered reliable in a lot of practical cases [Amodei2016], [Hendrycks2021a].

2.2 Robustness Interventions

As defined in Section 1.2, robustness interventions are methods designed to increase neural network robustness to a range of corruptions. In particular, robustness interventions have been proposed to tackle the issue of adversarial attacks (Section 2.2.1), synthetic corruptions (Section 2.2.2) and natural corruptions (Section 2.2.3).

We note that the context of this thesis is about addressing robustness to *unforeseen* corruptions. This means that during model conception, we do not have any sample drawn from the distributions that could arise at inference time. In concrete terms, this means that no domain adaptation strategy can be used.

2.2.1 Addressing Robustness to Adversarial Attacks

Adversarial robustness has been widely studied in the last few years, and a huge number of robustness interventions have been proposed to solve this issue. We present here the most widely studied.

Adversarial training [Goodfellow2015], [Madry2018], [Tramèr2018], [Na2018], is probably the most successful defense against adversarial examples. It consists in training neural networks with data augmentation on adversarial examples. This is currently the only defense, considered as able to significantly increase robustness to adversarial attacks that have been designed having a complete access to the attacked model [Athalye2018a]. However, we note two important issues

when using adversarial training (1) it increases neural network training time [Madry2018] and (2) the trained models have a lower accuracy on clean images (i.i.d samples) compared to models trained without adversarial training [Tsipras2019]. We note that solutions have been proposed to address these issues by reducing training time [Shafahi2019], [Wong2020] and increasing clean accuracies of adversarially trained models [Zhang2019].

Another approach consists in introducing random transformations within neural networks processings [Guo2018], [Xu2018]. For instance, a random crop, a random rescale and a JPEG compression at randomly chosen compression amount, can be applied to input images of a model. These transformations are used to make internal functioning of neural networks harder to access to attackers. These random transformations are also used to filter the brittle adversarial patterns that may be present in processed images. Random transformations can also be applied in the hidden layers of neural networks, for instance by using a randomly zeroing weights [Dhillon2018] or by adding noise after each convolution layer [Liu2018]. On contrary to adversarial training, these methods do not significantly increase training time. On the other hand, they can reduce the accuracy on clean samples too, and they have been shown to be weak against the attacks crafted in the so-called white-box settings (see Section 2.3.2) [Athalye2018a].

Generative Adversarial Networks (GAN), can be used to protect against adversarial examples [Samangouei2018], [Santhanam2018], [Song2018a]. The idea is to project images into the data manifold learnt by generators of GAN before sending them to classifiers. This projection aims to remove adversarial patterns, which are not considered as realistic by GAN, while conserving the semantic content of images. One drawback of this approach is the computational overhead entailed by projections. Besides, it has been shown to provide limited protection against white-box attacks [Athalye2018a].

2.2.2 Addressing Robustness to Synthetic Corruptions

A data augmentation on a specific synthetic corruption makes models robust to the considered corruption [Dodge2017a]. However, there is an infinite number of synthetic corruptions and we cannot train a model with data augmentation on all possible corruptions.

Fortunately, it has been shown that data augmentation on complex corruptions can be used to make models more robust to a large range of synthetic corruptions [Zhang2018], [Cubuk2019], [Lim2019], [Hendrycks2020b]. The idea is to use corruptions that are complex enough to change a large range of features in images (textures, object position, colorimetry, etc.). It has been shown that neural networks can be leveraged to obtain such complex corruptions [Geirhos2019], [Rusak2020]. For example, DeepAugment corrupts images, by propagating them into an autoencoder, on which random transformations such as weight zeroing or convolutional filter inversion are applied [Hendrycks2020a]. Examples of obtained images are displayed in Figure 2.4.

Self-supervised learning can be used to build models that are extremely robust to synthetic corruptions [Mahajan2018], [Yalniz2019], [Xie2020b]. However, these approaches have been designed mostly to deal with natural corruptions, so we describe these interventions in Section 2.2.3.

Although most of synthetic corruption robustness interventions are data driven, we note that several defenses propose to use alternative losses [Kornblith2020], architectures [Vasconcelos2020] or optimizers [Metz2019] instead. These approaches could be complementary to the data-driven ones.

2.2.3 Addressing Robustness to Natural Corruptions

The most successful approach proposed to make neural networks robust to natural corruptions is self-supervised learning. In a few years, we have seen the emergence of models that are incomparably more robust to natural corruptions than the models we had before [Mahajan2018], [Yalniz2019], [Xie2020b]. The self-supervised paradigm consists in making trained models complete auxiliary tasks on a huge diversity of unlabeled samples, in addition to the traditional main task

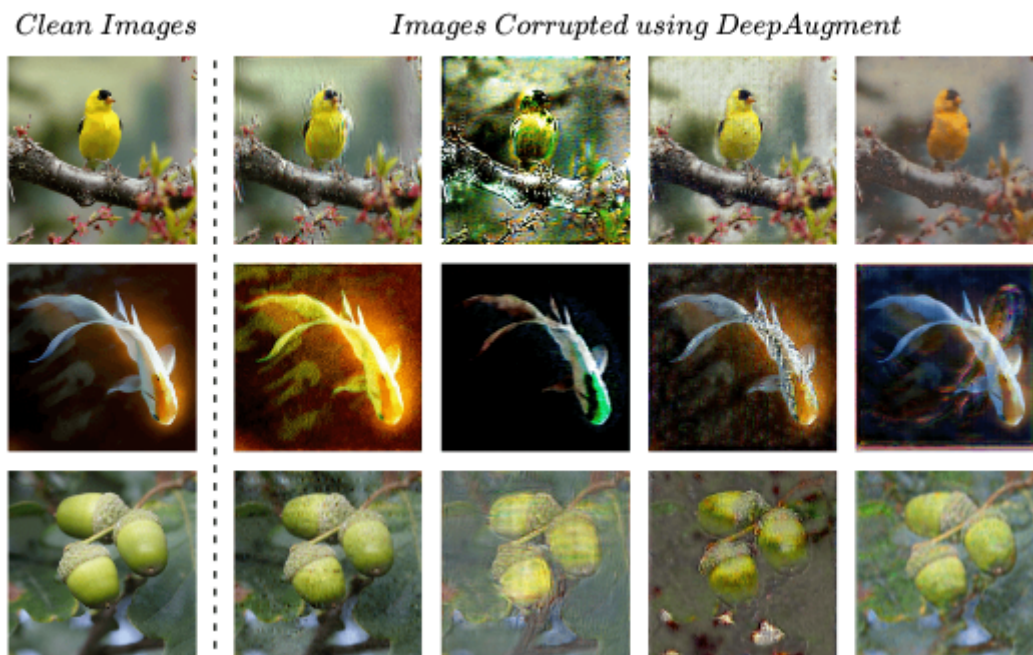


Figure 2.4 – Illustrations of complex corruptions generated using DeepAugment (images extracted from [Hendrycks2020a]).

carried out using labeled samples. An illustration of self-supervised training is provided in Figure 2.5. In this Figure, a model is trained to classify labeled images (the main task), and also trained to predict the rotation angle of unlabeled images, that have been randomly rotated before being sent to the model (the auxiliary task). The underlying idea behind the auxiliary task is to leverage information that can be extracted from a massive amount of images, which generally come from a huge diversity of distributions. Unfortunately, the models trained this way are very large and are incredibly expensive to train in terms of computation. Besides, the idea of using more and more data to make models perform well is conceptually unsatisfying. Since a child does not need thousands of images to reliably recognize objects, we can hope for cheaper learning approaches to address robustness.

Promising results in terms of robustness to natural corruptions have also been obtained using domain generalization [Gulrajani2021]. The idea is to enforce trained models to rely only on features that are predictive of several training distributions [Arjovsky2019], [Huang2020], [Sagawa2020]. The underlying idea is that features identified this way, which are consistently predictive for several of distributions, should also be predictive of other unseen distributions. Besides, models trained this way are enforced to ignore the features that are not predictive of all training distributions, because they are likely to be brittle and could be misleading when dealing with new distributions. For instance, we consider a human detector system that is trained on several distributions, among which only one distribution contains images of humans that are not white. Domain generalization methods prevent models trained on these training sets to use the skin color as a predictive attribute to recognize humans. Indeed, in the provided example, this attribute is not always predictive and then should not be used.

Domain generalization have two main drawbacks. Firstly, they obviously require to have several training sets drawn from different distributions to complete a training. This condition is rarely easy to satisfy. Secondly, domain generalization methods largely constraint models and can make them ignore some predictive features. As a consequence, the accuracy of models on their training distributions may be reduced compared to models trained with standard methodologies.

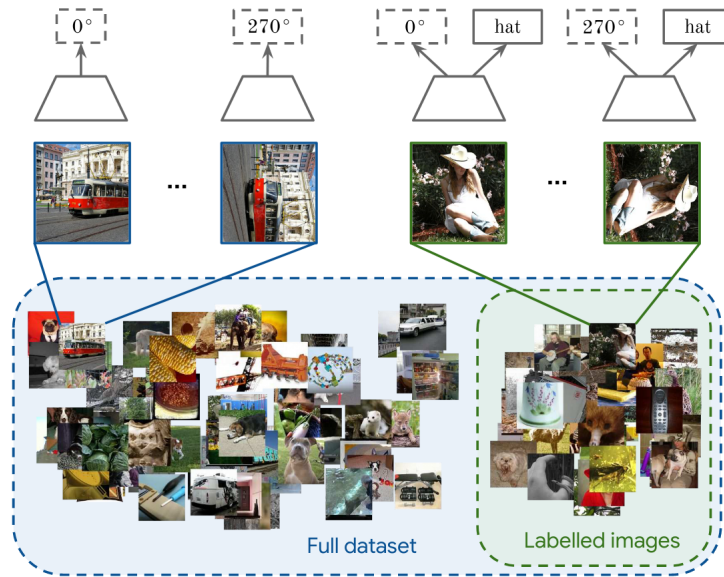


Figure 2.5 – Illustration of the self-supervised training paradigm. Here the complementary task to carry out in addition to the traditional classification task, consists in predicting rotation angles of images (scheme extracted from [Zhai2019]).

In the same way as for adversarial examples and synthetic corruptions, data augmentation is also considered as a promising approach to address robustness to natural corruptions. In particular, data augmentation on corruptions that are relatively complex, which alter a large diversity of features in images (colorimetry, brightness, image orientation...), can make models more robust to some natural corruptions [Geirhos2019], [Hendrycks2020b], [Rusak2020], [Hendrycks2020a].

2.3 Robustness Benchmarks

To determine which robustness interventions should be preferred to others, we need a method to estimate which interventions provide the highest robustness gains towards the considered corruption range. In other words, for each kind of corruptions considered, we need benchmarks able to estimate neural network robustness to this kind.

Robustness benchmarks are generally built on two components : a robustness metric and some distribution shifts [Hendrycks2019a], [Shankar2019], [Hendrycks2021b]. One can compare robustness of models using these two components. We note that by definition, one can associate a performance threshold to be reached with a robustness benchmark, to establish some robustness specifications (see Section 1.3). In the following sections, we present the most commonly used robustness metrics and the distribution shifts, employed to measure robustness to adversarial, synthetic and natural corruptions.

2.3.1 Robustness Metrics

A robustness metric is a procedure, that provided a neural network n and a corruption c , gives a robustness estimation of n towards c . We present here the most commonly used robustness metrics.

Accuracy. A simple approach consists in computing the percentage of accurate predictions of

models on corrupted images, it is called the accuracy metric. This metric directly gives the absolute performances of a neural network that faces a specific corruption. This metric is often used in production contexts, because the absolute performances of models is generally the only criterion that matters.

However, the accuracy metric does not fit the definition of robustness provided in Section 1.1 : *a model is considered as robust to a distribution shift, when its performances on samples drawn from this distribution are close to its performances obtained with training images.* To illustrate this, let us consider two models m_1 and m_2 which have accuracies on clean samples respectively equal to acc_1 and acc_2 , with $acc_1 = 0.9$ and $acc_2 = 0.7$. On samples corrupted with a corruption c , they respectively obtain acc'_1 and acc'_2 , with $acc'_1 = 0.68$ and $acc'_2 = 0.66$. When using the accuracy function as a robustness metric, we would conclude that m_1 is more robust than m_2 to c . However, the performance reduction due to the considered corruption is much higher for m_1 than for m_2 . Intuitively, we could argue that it is m_2 that is the less sensitive to c and should be considered as more robust. That is why in the academic context, to study the notion of neural network robustness, metrics that fit the definition provided above are generally preferred to the accuracy metric. Namely, a model should be considered as robust when its performances on i.i.d samples are close to its performances on o.o.d samples.

Residual Robustness. The residual robustness metric measures the accuracy drop caused by a corruption. This metric is easy to interpret, and is consistent with our definition of robustness : the lower a residual robustness score, the more the considered model has performances on corrupted samples close to its performances on clean samples. The residual robustness RR of a model m computed for a corruption c can be obtained using the following expression :

$$RR_c^m = acc_{clean}^m - acc_c^m \quad (2.1)$$

With acc_{clean}^m and acc_c^m corresponding to the accuracies of m respectively computed on clean images and images corrupted by c .

An important remark : this metric (and also all the metrics presented below), on contrary to the accuracy metric, does not provide any information in absolute terms on the performances of models on corrupted samples. An untrained model, which provides random answers, would have a residual score of zero (in non-adversarial scenarios), which is the best possible score. In other words, using this metric generally requires to monitor the accuracy on clean samples and verify that this one is high enough to certify that the considered models handle correctly their tasks. To illustrate this, let us consider a robustness intervention that increases the residual robustness scores computed with several corruptions but that also reduces the clean accuracy. In this case, determining if the robustness intervention is valuable is a compromise between the importance given to robustness to the considered corruptions and the one given to accuracy on clean samples.

APP. The Accuracy Percentage Preserved (APP), is a simple robustness metric that can be computed using the following formula :

$$APP_c^m = \frac{acc_c^m}{acc_{clean}^m} \quad (2.2)$$

The notations are the same as the ones used in the equation (2.1). APP is similar to the residual robustness metric : it directly tells whether the performances of a model on clean samples are close to its performances on corrupted samples. The difference between these two metrics is that the residual robustness gives the performances drop in terms of accuracy point lost, while APP gives the performances drop in terms of percentage of preserved accuracy. Moreover, instead of being between 0 and the accuracy on clean samples as the residual robustness score, APP is between 0 and 1. These metrics can generally be used indifferently. When using APP , the higher is the better.

Effective Robustness. Taori et al. carried out experiments suggesting that the accuracy on clean samples is in practice correlated with the accuracy on corrupted samples [Taori2020]. This phenomenon is illustrated for two robustness benchmarks (ImageNet-V2 [Recht2019] and ObjectNet [Barbu2019]) in Figure 2.6.

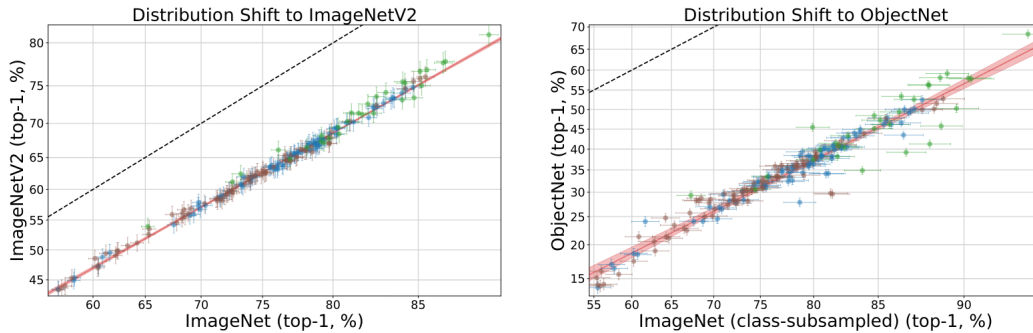


Figure 2.6 – Illustration of the empirically measured linear correlation between accuracy on clean samples and accuracy on corrupted samples. Each point of this figure provides two accuracies of one model, respectively computed with clean samples and samples extracted from the natural corruption benchmarks ImageNet-V2 (left) and ObjectNet (right) (scheme extracted from [Taori2020]).

A consequence of this observation is that methods increasing accuracy on clean samples are generally seen as robustness interventions because they also increase accuracy on corrupted samples. But Taori et al. argue that this last increase is often just a natural consequence of the correlation between accuracy on clean samples and accuracy on corrupted samples. Then, they propose a metric that measures robustness gains beyond what is expected from having higher accuracies on clean samples. This metric noted ER is computed using the following formula :

$$ER_c^m = acc_{clean}^m - \beta * acc_c^m \quad (2.3)$$

β corresponds to the linear coefficient associated to the considered distribution shift, i.e., it is the slope of the straight plain lines displayed in Figure 2.6.

This robustness metric enables to directly determine whether a robustness intervention provides a robustness gain in addition to the one induced by the accuracy gain on clean samples. This can be visualized in Figure 2.6, a robustness intervention that has a positive effect on effective robustness, is traduced by models positioned above the straight line and closer to the dotted line. Unfortunately, β is measured empirically and requires to run a lot of tests for each considered distribution shifts. Then, while being useful to compare robustness interventions in theoretical studies [Taori2020], using this metric is extremely costly and is not convenient in practice.

Relative Corruption Error. When using either the APP or the residual robustness metrics alone, one cannot say to what extent a robustness measured is valuable or not. For instance, having a APP score of 0.9 on a severe Gaussian noise is much more difficult than having the same APP score on a low-power Gaussian noise. The value an APP or a residual robustness score in absolute terms depends on the severity of the considered corruption. A way to determine this absolute value, is to compare computed robustness scores, with the ones obtained using a reference model. The more the reference model obtains low performances, the more the robustness scores obtained using other models can be considered valuable. This is the underlying idea behind the relative Corruption Error (generally noted CE). Using the same notation as for equation 2.1, it can be computed using the following formula :

$$CE_c^m = 100 * \frac{acc_c^m - acc_{clean}^m}{acc_c^{alex} - acc_{clean}^{alex}} \quad (2.4)$$

This metric is based on a comparison between the robustness of tested models and the robustness of an AlexNet [Krizhevsky2012]. In concrete terms, a CE score lower than 100, means that m is more robust to c than an AlexNet. Then, obtaining $CE_c^m = 50$, means that the robustness of m to c is valuable because m is much more robust to c than AlexNet. The drawback of this metric is that the accuracy of AlexNet on samples corrupted with the considered corruption are not always known or published. In this case, it is required to compute it oneself. Of course, other reference models can be chosen but AlexNet is the one currently used by the community.

Flip Rate. Some robustness benchmarks propose to study the stability of neural network predictions on sequences of similar images [Shankar2019], [Hendrycks2019a], [RichardWebster2019]. They need specific robustness metrics to achieve it. Namely, instead of using metrics that study prediction accuracies, these benchmarks are associated with metrics that study prediction consistencies. For instance, the Flip Rate metric measures the frequency of prediction changing of models when processing sequences of similar images (examples of similar image sequences are displayed in Figure 2.7). The less model predictions vary when processing such sequences, the more tested models are considered robust.

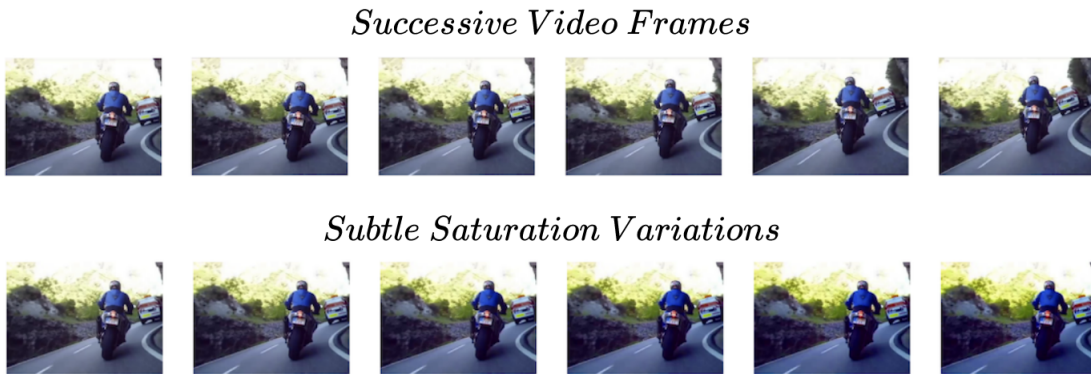


Figure 2.7 – Sequences of similar images used to measure neural network prediction stability (images extracted from [Gu2019]).

Any robustness metric presented above can be used depending on the needs. These metrics are associated with distribution shifts to constitute robustness benchmarks. Several distribution shifts may be used depending on what is measured. In the following sections, we detail how distribution shifts are generally selected when measuring robustness to adversarial examples, synthetic corruptions and natural corruptions.

2.3.2 Measuring Robustness to Adversarial Attacks

2.3.2.1 Selecting Adversarial Examples to Include in Benchmarks

To measure robustness to adversarial examples, it is recommended to use a set of attacks that is as diverse as possible [Carlini2019]. Indeed, there are a lot of kinds of adversarial examples, which can alter different types of features. For instance, it has been shown that being robust to L_p bounded adversarial examples, does not imply being robust to adversarial spatial transformations [Xiao2018]. Then, adversarial robustness estimations should include a group of corruptions representative of the diversity of adversarial examples that may be encountered. In this section,

we present parameters of adversarial attacks that can be considered to select a set of corruptions diverse enough to make accurate estimations of adversarial robustness.

White-Box, Black-Box settings. When an attacker has information about a neural network, he or she can attack it more efficiently. Basically, we distinguish two scenarios : the white-box and the black-box settings. In white-box settings, the attacker has fully access to the attacked neural network : its architecture, its weights, its training data, etc. In this setting, it is extremely difficult to defend neural networks against adversarial attacks [Athalye2018a]. Fortunately, industrials can generally keep their models and eventual defenses secret to prevent attackers from being in the white-box scenario.

In the black-box settings, the model weights, architecture and eventual defenses are hidden. Note that a few information (training data of the attacked model [Kurakin2017a] or its final decisions [Brendel2018]) may be hard to hide and can be figured out by attackers. For instance when it comes to reidentification tasks, datasets are so difficult to constitute (because of privacy, data gathering difficulty, labeling cost...) that an attacker can reasonably suppose that some open-source datasets have been used to train deployed models. Then, the idea of the black-box settings is more to hide the internal functioning of neural networks rather than hiding all information about them.

The most commonly used black-box attacks are based on the transferability of adversarial examples : adversarial examples crafted using one model m transfer to models trained using the same training set as m [Kurakin2017a]. As mentioned above, open-source datasets are widely used in some real-world applications, then this scenario is plausible and should be considered. Other adversarial examples can be crafted by observing the final outputs of models to successive queries [Brendel2018]. The idea is to determine how to fool a model by observing its external behavior, which is often available. Obviously, the behavior of circulating autonomous vehicles will be observable for example.

To summarize, the black-box scenario is more realistic than the white-box one, because it corresponds to threats that could be encountered in real-world applications. Fortunately, these attacks are way less harmful than white-box ones [Lu2017].

Targeted, Untargeted. An adversarial attack is said to be targeted [Kurakin2017b], when it aims to make attacked models having a specific behavior. The aim of an attack can be for instance to make classifiers answer that encountered images represent a dog, even if these images represent a truck. On the other hand, untargeted attacks just aim to make models having an unexpected behavior, without specifying what this behavior should be. In the previous example, untargeted attacks would just prevent models from recognizing a truck, without specifying what models should perceive. Criminal targeted attacks could be designed to be the most harmful as possible : making an autonomous vehicle confusing a stop sign with a priority road sign...

Attack Bound. An adversarial attack is generally bounded, which means that the distance between a produced adversarial image and the clean image it is derived from, must be lower than a specific value called the attack bound [Fawzi2016]. This bound is fixed to prevent adversarial corruptions from changing the true labels in images : an image representing a dog, should still represent a dog even after being modified by an adversarial attack. Otherwise, it would not be an adversarial corruption anymore because it would change semantic content of images.

The distance used to compute these bounds is generally pixel-wise such as the L_2 or the L_{inf} norms. However, different distances may be used depending on the attack. For instance adversarial rotations [Engstrom2019] can be bounded using a maximum rotation degree. It has been shown that being robust to attacks bounded by a specific value ϵ_1 is not necessarily correlated with being robust to attacks bounded using a different value ϵ_2 [Carlini2019].

Then, to summarize, a set of attacks used to estimate adversarial robustness should contain both

target and untargeted attacks, both black-box and white-box settings and attacks bounded using various maximum values. More refined parameters could be taken into account to further increase the attack diversity [Machado2020]. We just present here the main ones.

2.3.2.2 Examples of Adversarial Attacks

We present here some of the most currently used adversarial examples. These attacks can be gathered to constitute benchmarks that respect the recommendations mentioned above. FGSM (Fast Gradient Sign Method) is one of the simplest adversarial attacks [Goodfellow2015]. It exploits gradients of neural network cost functions :

$$x_{adv} = x + \epsilon * sign(\nabla_x L(x, l_{true})) \quad (2.5)$$

With x a sample to transform, l_{true} its label and L the cost function of the attacked model. We note ϵ the amount of the introduced adversarial perturbation. An illustration of this attack is provided in Figure 2.8.

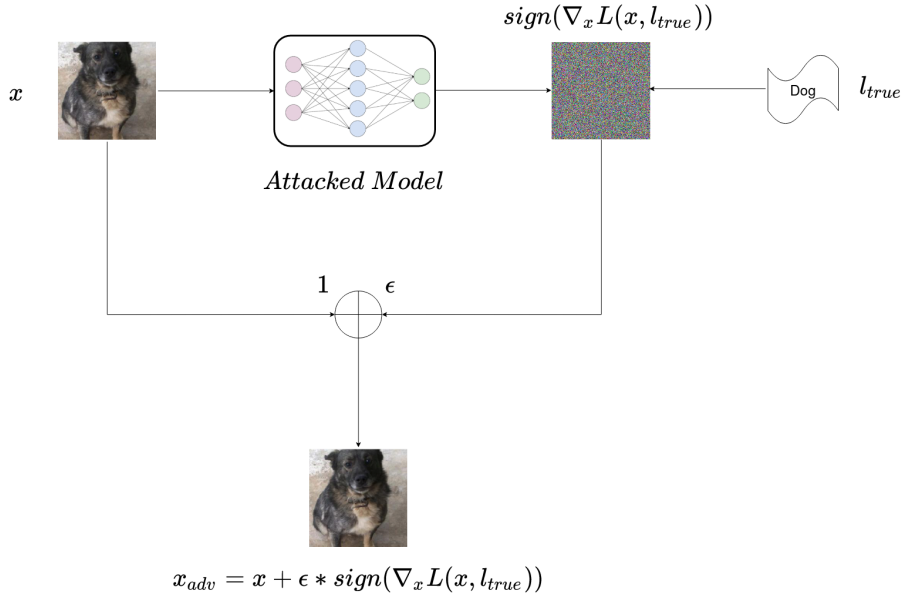


Figure 2.8 – Scheme of an adversarial example crafting using the FGSM algorithm.

Despite its simplicity, protecting models against this attack (especially in the black-box settings) is quite challenging [Kurakin2017b]. A variation of FGSM called PGD [Madry2018] (Projected Gradient Descent), which computes gradients iteratively can be computed using the following expression :

$$x^{k+1} = x^k + \frac{\epsilon}{n} * sign(\nabla_x L(x^k, l_{true})) \quad (2.6)$$

Starting with $x^0 = x$, the upper expression is computed n times to craft an adversarial example. For the same corruption amount, PGD is much more harmful than FGSM in the white-box settings. However, this attack transfers way less among neural networks in the black box settings [Kurakin2017b]. A way to make iterative attacks more transferable is to use momentum. Similarly to the momentum usually used with stochastic gradient descent, it can be useful to memorize the precedent iterations to compute the current iteration of a PGD process. Following this idea, MI-FGSM uses the same iterative process than PGD, but replaces the current gradient computed by

the accumulated gradient of all previous steps [Dong2018]. Adversarial examples crafted this way are particularly transferable among neural networks.

Attacks presented above are untargeted, but they can be turned into their targeted version. For example, Least Likely FGSM (LL-FGSM) is a variation of FGSM that targets the class that is given the lowest score by the attacked neural networks [Kurakin2017a]. Considering l_{least} the label that corresponds to this class, an adversarial example can be crafted using the following formula :

$$x_{adv} = x - \varepsilon * \text{sign}(\nabla_x L(x, l_{least})) \quad (2.7)$$

Similarly to LL-FGSM, LL-PGD is a variation of PGD that intends to make neural networks give a high score to the label l_{least} [Kurakin2017a]. Starting with $x^0 = x$, the adversarial example is built by computing several times the following expression :

$$x^{k+1} = x^k - \frac{\varepsilon}{n} * \text{sign}(\nabla_x L(x^k, l_{least})) \quad (2.8)$$

Another widely used adversarial attack is the L_2 Carlini Wagner (CW_2) attack [Carlini2017]. It is an optimization-based algorithm that tries to solve in a fixed number of iterations the following expression :

$$\min_{\delta} \{ \|x + \delta\|_2^2 + c * g(x + \delta) \} \quad (2.9)$$

where g is an objective function defined by :

$$g(x_{adv}) = \max(\max_{i \neq t} \{Z(x_{adv})_i\} - Z(x_{adv})_t, 0) \quad (2.10)$$

This is the untargeted version of the CW_2 attack where t is the label of the original sample. Z is the output of the attacked classifier before its softmax. Increasing the parameter c called confidence, encourages a misclassification with high confidence of the attacked neural network. At the end of the optimization process, δ is added to the original image to get the adversarial example.

2.3.3 Measuring Robustness to Synthetic Corruptions

2.3.3.1 Selecting Synthetic Corruptions to Include in Benchmarks

Neural networks robustness to synthetic corruptions is generally estimated by using a selection of corruptions [Karahan2016], [Temel2017], [Michaelis2019]. Namely, the robustness of considered neural networks is measured for every single corruption in this selection. The mean of all the obtained robustness scores gives an estimation of the robustness of the considered models to synthetic corruptions.

To our knowledge, conversely to adversarial examples, there is no precise criterion that has been proposed to determine which corruptions should be selected to constitute synthetic corruption benchmarks. In practice, several corruption selection approaches can be used : choosing the most currently used image transformations in signal processing [Dodge2016]; choosing various corruptions that are different according to human perception [Geirhos2018]; selecting corruptions to alter various frequency domains (according to the Fourier spectrum) [Yin2019]; etc. It is not known which corruption selection approaches should be preferred to others when measuring robustness.

2.3.3.2 Examples of Synthetic Corruption Benchmarks

Synthetic corruption selections have been proposed to estimate neural network robustness in various computer vision tasks such as face recognition [Karahan2016], object detection [Michaelis2019], image segmentation [Kamann2020], saliency region detection [Che2020], traffic sign recognition [Temel2017], scene classification [Tadros2019], etc.

ImageNet-C [Hendrycks2019a] is probably the most commonly used synthetic corruption benchmark. It is used to estimate the robustness of ImageNet classifiers. It is built on fifteen common corruptions which can be classified into noises, blurs, weather and digital corruptions. Illustration of images corrupted using the ImageNet-C corruptions are displayed in Figure 2.9. Each of these corruptions is associated with five corruption severities illustrated at the bottom the Figure.

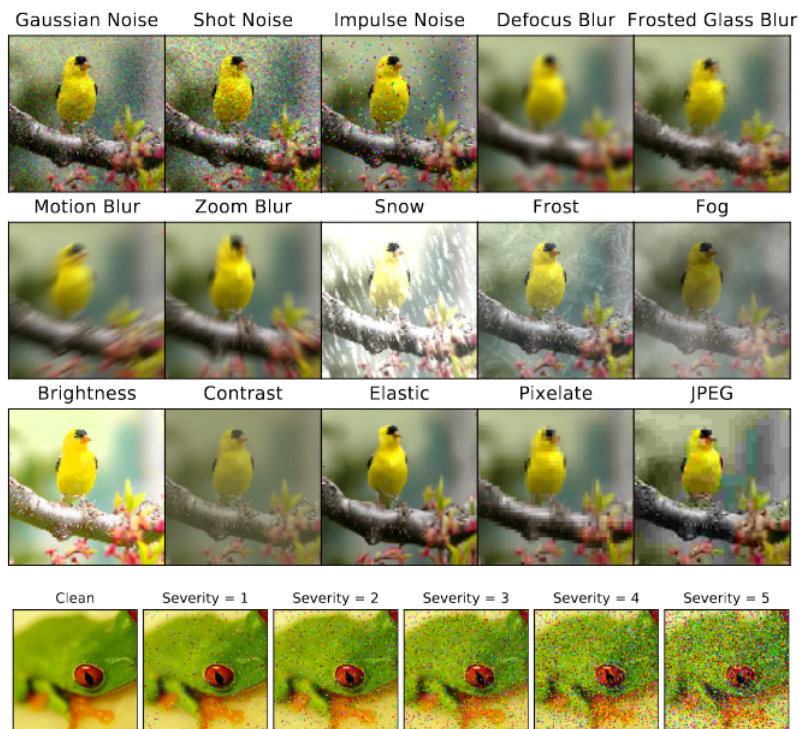


Figure 2.9 – Illustration of the fifteen ImageNet-C corruptions (Scheme extracted from [Hendrycks2019a]). The five severity levels associated with each of these corruptions are illustrated at the bottom of the Figure (using the salt-pepper noise corruption as an example).

Along with ImageNet-C, ImageNet-P has been proposed to study the stability of models to subtle variations in corruption amounts [Hendrycks2019a]. Instead of studying robustness to diverse corrupted images, ImageNet-P estimates robustness using sequences containing corrupted images derived from single images (an example of such sequence is displayed at the bottom of Figure 2.7). The corruption amount subtly changes throughout each sequence.

Synthetic corruptions used in the presented benchmarks are mostly traditional image transformations, which have been used for a relatively long time in the computer vision community such as Salt-Pepper Noise, Rotations, Gaussian Blur, Hue Variations, Contrast Loss, etc. These transformations are convenient to use because they are implemented and optimized in a lot of open-source computer vision libraries [Buslaev2020], [Jung2020], [Alexander2021].

2.3.4 Measuring Robustness to Natural Corruptions

2.3.4.1 Selecting Natural Corruptions to be Considered when Measuring Robustness

Similarly to synthetic corruptions, the way of selecting the natural corruptions to estimate robustness is not clearly established. The idea is generally to measure neural network robustness to the largest number of natural corruptions as possible.

We simply note that it has been very recently recommended to use two kinds of natural corrup-

tions when estimating robustness. They are called diversity shifts and correlation shifts [Ye2021]. Diversity shifts occur when the features of training distributions are largely different from features of test distributions. For instance, ImageNet-D, contains quickdraws and infographs, which contain radically different features than the ones found in photographs [Rusak2021]. An example of diversity shift is illustrated on the left in Figure 2.10). Correlation shifts occur when features are predictive of one class in training sets, but these same features are predictive of a different class in test sets. For instance, a model trained with images of sheep on grass and seagulls on beach, is likely to be fooled by an image of sheep on a beach. This kind of distribution shift is illustrated on the right of Figure 2.10.

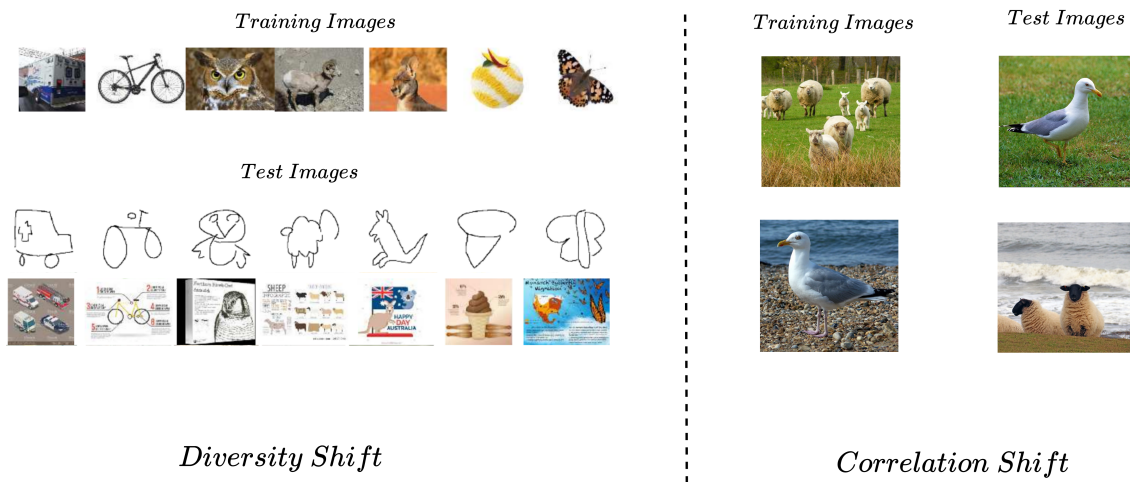


Figure 2.10 – Illustration of the two kinds of natural distribution shifts identified in [Ye2021]. The images on the left are extracted from [Rusak2021].

Ye et al. [Ye2021] argue that these two categories of natural distribution shifts are radically different and then should be both considered when estimating robustness to natural corruptions.

2.3.4.2 Examples of Natural Corruption Benchmarks

A lot of natural corruption benchmarks have been proposed to study image classifiers robustness. Some of them measure robustness to unexpected backgrounds [Beery2018], [He2021]. An example can be an image of a teapot thrown in the air and surrounded by a sky background. Other benchmarks measure the stability of neural networks on sequences of close images [Shankar2019]. Another idea consists in presenting artistic renditions to neural networks [Li2017], [Wang2019a], [Hendrycks2020a]. The idea is that if model perceives that Bugs Bunny is a rabbit, it is likely that this model "understands" the true underlying concept behind the class "rabbit". Then it can be considered as robust. Benchmarks that present unexpected object orientations or viewpoints are also widely used [Barbu2019], [Djologna2021] : a model should recognize a car even when it is flipped and landed on its roof. Challenging benchmarks are also constituted by simply picking images that are shown to be difficult to be classify by neural networks [Hendrycks2021b].

2.4 Bridging the Gap Between Adversarial, Synthetic and Natural Corruption Robustnesses

In the recent years, large improvements have been made to better address robustness to adversarial, synthetic and natural corruptions. Namely, we respectively presented in Section 2.2 and 2.3, promising robustness interventions and benchmarks that have been proposed for each of these kinds of distribution shifts.

However, adversarial, synthetic and natural corruptions are largely studied separately. There are a very few works that consider at least two of these kinds of corruptions at the same time. As a consequence, we do not really know whether the results on neural network robustness obtained studying one of these kinds are also valid for an other kind. For instance, some architectures have been shown to increase robustness to synthetic corruptions [Vasconcelos2020]. Should we also use these architectures to be less sensitive to adversarial corruptions ? On contrary could they have unexpected prejudicial effects when encountering that kind of attacks ? To answer such questions, we need to better understand the links between robustnesses to adversarial, synthetic and natural corruptions.

Moreover, as explained in Section 1.4, to establish better robustness specifications, we propose in this thesis to determine which corruptions are correlated in terms of robustness. Namely, having such knowledge should enable to build specifications that provide stronger robustness guarantee towards unforeseen corruptions that may be encountered.

Therefore, we propose in Chapter 3 and 4, to bridge the gap between adversarial, synthetic and natural corruption robustnesses. In doing so, we provide some of the first results that enable to better understand the links between robustnesses to these kinds of corruptions.

3

On Interaction Between Adversarial Corruption Robustness and Synthetic Corruption Robustness

Contents

| | | |
|------------|--|-----------|
| 3.1 | Introduction | 27 |
| 3.1.1 | Features Extracted by Adversarially Trained Models Align with Human Perception | 27 |
| 3.1.2 | Trade-off Between Clean Accuracy and Adversarial Robustness | 27 |
| 3.1.3 | Organization | 28 |
| 3.2 | Correlation Between Adversarial and Synthetic Corruption Robustnesses | 28 |
| 3.2.1 | Introduction | 28 |
| 3.2.2 | Experimental Set-up | 29 |
| 3.2.3 | Does Adversarial Corruption Robustness Transfer to Synthetic Corruptions ? | 31 |
| 3.2.4 | Does Synthetic Corruption Robustness Transfer to Adversarial Attacks ? | 32 |
| 3.2.5 | Concurrent Works | 33 |
| 3.2.6 | Conclusion | 34 |
| 3.3 | Addressing Both Adversarial and Synthetic Corruption Robustnesses using M-TLAT. | 35 |
| 3.3.1 | Introduction | 35 |
| 3.3.2 | Combining Mixup with Targeted Labeling Adversarial Training : M-TLAT | 35 |
| 3.3.2.1 | Mixup | 35 |
| 3.3.2.2 | TLAT | 36 |
| 3.3.2.3 | M-TLAT | 37 |
| 3.3.3 | Experimental Set-up | 38 |
| 3.3.3.1 | Robustness Benchmark | 38 |
| 3.3.3.2 | Training Details | 39 |
| 3.3.4 | Results | 40 |
| 3.3.4.1 | Robustness of Models trained with M-TLAT | 40 |
| 3.3.4.2 | Complementarity between Mixup and TLAT | 41 |
| 3.3.4.3 | Ablation on Labeling Strategy in Adversarial Trainings | 41 |
| 3.3.5 | Concurrent Works | 43 |
| 3.3.6 | Conclusion | 44 |
| 3.4 | Conclusion | 45 |

3.1 Introduction

Adversarial corruption robustness and synthetic corruption robustness are generally considered as two different fields, which are usually addressed separately. Robustness interventions are almost always crafted to address only one of these two kinds of corruptions and the robustness to the other kind is rarely evaluated. Consequently, we do not really know whether robustness to adversarial attacks transfers to synthetic corruptions or conversely. Yet, two experiments described in the following Sections 3.1.1 and 3.1.2, drew our attention and suggested that these corruptions could be correlated in terms of robustness.

3.1.1 Features Extracted by Adversarially Trained Models Align with Human Perception

The first experiment that made us question the idea of studying adversarial and synthetic corruption robustnesses separately has been conducted by Tsipras et al [Tsipras2019]. More precisely, they trained several models robust to adversarial attacks, using adversarial training [Madry2018]. Then, they visualized features learned by these models, by computing the gradient of their loss with respect to some input images (obtained results are displayed in Figure 3.1). In doing so, they observed that features learned by adversarially trained models are understandable to humans, on contrary to features learned by their non-robust counterparts.

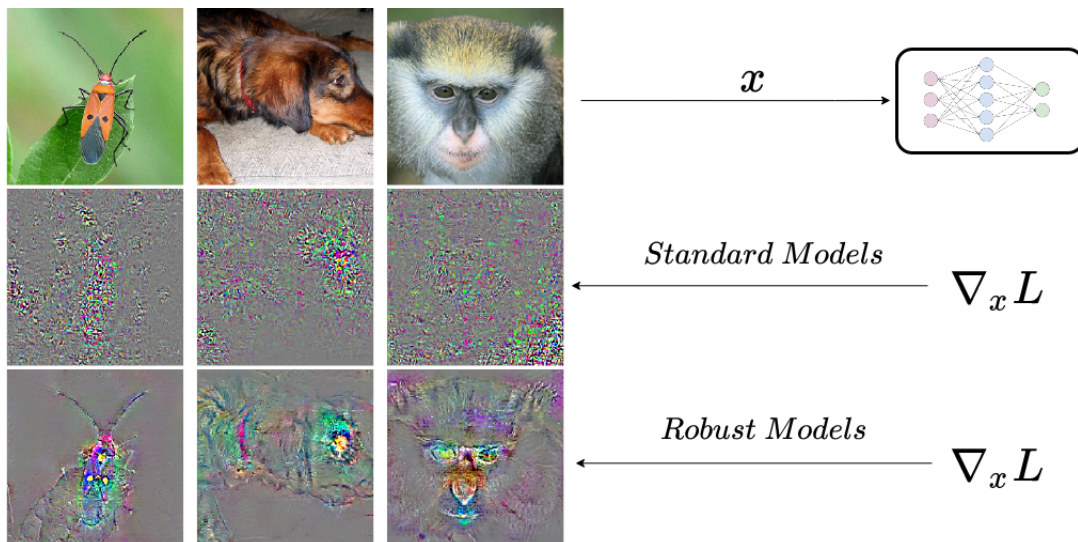


Figure 3.1 – Illustration of computed gradients for two models, of their loss L with respect to three input images x . Models robust to adversarial examples extract features that align better with human perception. (Scheme adapted from [Tsipras2019]).

Looking at these results leads us to the following reasoning : humans are much more robust than neural networks to corruptions [Goodfellow2015], [Geirhos2018]. So we postulate that learning human interpretable features, is likely to make neural networks perceive the world in a more similar way as human, and then should make trained models robust to distribution shifts just as humans. In other words, this experiment suggests that adversarial robustness could be a key to achieve robustness in a broad sense.

3.1.2 Trade-off Between Clean Accuracy and Adversarial Robustness

The second element that drew our attention, is the fact that successful defenses against adversarial examples have been shown to reduce accuracy on i.i.d samples [Madry2018], [Su2018],

[Zhang2019]. One simple interpretation of this phenomenon is that adversarial training acts as a strong regularization, that prevents models from completely fitting their training sets. Indeed strong regularizations can make neural networks underfit their training sets [Goodfellow2016], i.e., they can reduce accuracy on clean samples.

While adversarial training makes models robust to adversarial examples, we wonder whether this regularization method could also make neural networks robust to other kinds of corruptions. Indeed, some regularization strategies have been shown to provide unexpected robustness gains towards corruptions that were not studied during the conception of these regularizations. For instance, dropout has a side effect of making models more robust to noises [Koziarski2017]. Since adversarial training is a particularly restrictive regularization [Su2018], [Tsipras2019], it could have a significant effect on neural network robustness towards various corruptions.

3.1.3 Organization

The two experimental results described above suggest that adversarial robustness could imply robustness to other corruptions. We note that adversarial corruption robustness has been widely studied these last years [Szegedy2014], [Goodfellow2015], [Papernot2016], [Papernot2017], [Schmidt2018], [Ilyas2019], and it would be interesting to see whether we can leverage all this acquired knowledge to address robustness to other kinds of corruptions. Synthetic corruptions (as explained in Section 2.2.2) are particularly convenient to use. Then, they are relevant candidates to verify whether adversarial corruption robustness could be a key to achieve robustness in a broad sense. Consequently, we propose in this Chapter to study the correlation between adversarial and synthetic corruption robustnesses.

As a first step, in Section 3.2, we carry out a thorough study to determine if increasing robustness to adversarial examples implies increasing robustness to synthetic corruptions and conversely. Then, in Section 3.3, we leverage the knowledge acquired in Section 3.2, to establish one of the first robustness intervention shown to increase robustness to a large diversity of both adversarial and synthetic corruptions.

3.2 Correlation Between Adversarial and Synthetic Corruption Robustnesses

3.2.1 Introduction

In this Section, we want to study the possible links between adversarial and synthetic corruption robustnesses. Adversarial corruption robustness has been widely studied these last years, and we also observe an emergence of articles studying synthetic corruption robustness [Dodge2016], [Hendrycks2020a], [Taori2020]. However, there are very few works studying simultaneously both kinds of corruptions.

Among these rare studies, we note some interesting results found in [Fawzi2016], [Franceschi2018], showing that neural networks robust to random noises remain vulnerable to adversarial attacks. Besides, links between small geometric transformations and adversarial examples are studied in [Xiao2018], [Engstrom2019]. More precisely, it is argued in these studies that robustness to additive adversarial perturbations and robustness to geometric transformations (rotations, translations, etc.) are orthogonal concepts.

These works consider very specific kinds of synthetic corruptions. We propose in this Section 3.2 to enlarge the range of synthetic corruptions considered. In concrete terms, in Section 3.2.3, we train models that are shown to be robust to adversarial examples, and we measure the robustness

of these models to a large range of synthetic corruptions. Reciprocally, we train models that are shown to be robust to various synthetic corruptions in Section 3.2.4, and we study the robustness of these models towards adversarial examples. The obtained results enable to better understand the links between adversarial and synthetic corruption robustnesses.

3.2.2 Experimental Set-up

Here are provided the implementation details associated to the experiments carried out in Sections 3.2.3 and 3.2.4.

Used dataset. We choose the ImageNet dataset to carry out our experiments [Deng2009]. ImageNet is widely used, challenging and big enough for achieving adversarial trainings [Schmidt2018]. Adversarial training and data augmentation are used to train models in our experiments. These robustness interventions are unfortunately computationally expensive, so they increase training times of neural networks. To speed up trainings, we decided to use a subset of ImageNet. It is composed of 5 super-classes, each regrouping several ImageNet classes. The chosen classes are : *bird*, *dog*, *insect*, *primate* and *fish*. They respectively correspond to the ImageNet class ranges 80-100, 151-268, 300-319, 365-382 and 389-397. The choice of the classes was made by drawing inspiration from the experiments conducted in [Tsipras2019]. We ensure the size equality of the classes by splitting the biggest classes to fit the smallest one. The resulting classes each contain ten thousand images. The obtained dataset is more suitable for achieving dozens of trainings in a reasonable amount of time.

Training hyperparameters. All models are trained using stochastic gradient descent with a learning rate of 0.01. When the training accuracy reaches a plateau, the learning rate is divided by 10. We use the early stopping strategy to fix when trainings are ended [Goodfellow2016]. We use a weight decay of 0.0001 and a batch size of 128. Models are optimized using the cross entropy loss function. We call *standard* models, the neural networks trained with these hyperparameters and without using any data augmentation.

Synthetic Corruption Selection. To carry out our experiments, we need to gather a large group of diverse synthetic corruptions. To achieve it, we select corruptions that change different kinds of features in images : the brightness, the colorimetry, the position of observed objects, etc. More precisely, the implementation details and the illustrations of the chosen corruptions can be found in Figure 3.2.

Each of the selected perturbations except *flip* is associated with a severity range. For instance, square masks used for the occlusion perturbation can vary from 5 to 15 percent of the image size. We define a procedure to fix the upper and lower bounds of each perturbation intensity range. In order to fix the lower bound, the procedure starts with a very small perturbation. The intensity of the perturbation is progressively raised. During this increase, the behavior of the *standard* model on the corrupted images is periodically tested. The lower bound of the perturbation range is reached when the accuracy of the neural network starts to decrease : one accuracy point is lost. We keep increasing the severity until the *standard* model loses fifty percent of its original accuracy. At this point, the upper bound is fixed.

Adversarial Example Selection. Following the recommendations given in [Carlini2019], we select diverse kinds of adversarial examples. Namely, we use four different attacks : FGSM [Goodfellow2015], FGSM-LL [Kurakin2017a], PGD and LL-PGD [Madry2018] in both the black-box and white-box settings (details about these attacks can be found in Section 2.3.2.2). Some of the used adversarial attacks are targeted and others are untargeted. The amounts of corruptions used are diverse, namely they are drawn from a uniform distribution ranging from 0.005 to 0.05 (with a pixel value range of $[-1; 1]$). A VGG network [Simonyan2015] trained on the same training set as attacked neural networks, using the exact same training hyperparameters, is used to craft black-box attacks. PGD and LL-PGD are computed over ten iterations.

In Section 3.2, robustness to both adversarial attacks and synthetic corruptions is measured





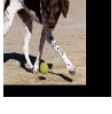


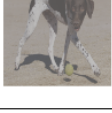

| Corruption Name | Illustration | Description | Severity Range |
|------------------|---|--|---|
| Gaussian Noise |  | Add zero mean Gaussian noise to the image | Standard deviation: 0.05 to 0.18 |
| Blur |  | Interpolate the image with its equivalent blurred with five convolution using a 3x3 filter. The filter is filled with the value 1. | Interpolation factor associated to the blurred image: 0.4 to 0.95 |
| Artifacts |  | Add artifacts into the image. The artifacts are made with a small dotted lines | Number of artifacts: 15 to 170 |
| Obstruction |  | One randomly chosen square area is filled with a unique pixel value | Size of the square 47 to 125 |
| Translation |  | One horizontal and one vertical translation | Number of pixels translated for both translations: 15 to 62 |
| Rotation |  | Clockwise or anticlockwise rotation | Degrees of the rotation: 7 to 50 |
| Brightness |  | Select a unique value. Add this value or subtract it to all the pixel values of the image | Value added or subtracted: 0.16 to 0.51 |
| Contrast |  | Reduce the contrast of images | Contrast reduction factor: 0.33 to 0.74 |
| Color Distortion |  | Add a unique value to all pixels of one of the RGB channel | Value added: 0.09 to 0.40 |

Figure 3.2 – Illustration of various image transformations used to estimate robustness to synthetic corruptions.

using the APP metric (details about this metric are provided in Section 2.3.1).

3.2.3 Does Adversarial Corruption Robustness Transfer to Synthetic Corruptions?

To determine whether adversarial corruption robustness transfers to synthetic corruptions, we propose to train models that are robust to adversarial examples and to test their robustness to synthetic corruptions. To achieve it, we use adversarial training [Madry2018], which is currently the most efficient technique used to make neural networks robust to adversarial examples. More precisely, we train four models called *fgsm*, *pgd*, *fgsm_ll* and *pgd_ll*, that are respectively augmented with the FGSM, PGD, LL-FGSM and LL-PGD attacks. These models have the ResNet-18 architecture [He2016] and are trained using the parameters detailed in Section 3.2.2. Then, we measure the robustness of these models to the four considered attacks in both the black-box and white-box settings. The obtained results are displayed in Table 3.1.

| Attack Model | Clean | B-Box fgsm | B-Box fgsm_ll | B-Box pgd | B-Box pgd_ll | W-Box fgsm | W-Box fgsm_ll | W-Box pgd | W-Box pgd_ll |
|-----------------|-------|---------------|------------------|--------------|-----------------|---------------|------------------|--------------|-----------------|
| standard | 83 | 0.68 | 0.70 | 0.73 | 0.95 | 0.02 | 0.07 | 0.00 | 0.04 |
| fgsm | 81 | 0.96 | 0.97 | 0.97 | 0.99 | 0.62 | 0.89 | 0.27 | 0.67 |
| fgsm_ll | 82 | 0.96 | 0.97 | 0.98 | 0.99 | 0.42 | 0.75 | 0.36 | 0.79 |
| pgd | 75 | 0.98 | 0.98 | 0.98 | 1.00 | 0.47 | 0.86 | 0.42 | 0.85 |
| pgd_ll | 76 | 0.93 | 0.94 | 0.98 | 0.99 | 0.41 | 0.79 | 0.38 | 0.82 |
| syn_rob | 84 | 0.67 | 0.68 | 0.73 | 0.95 | 0.02 | 0.08 | 0.01 | 0.05 |

Table 3.1 – Robustness of models to adversarial examples, estimated by using the APP metric, in both the black-box (on the left) and white-box (on the right) settings. The column entitled "Clean", contains the accuracy of tested models on clean samples.

We observe that the adversarially trained models are significantly more robust to all the tested attacks than the *standard* model. We note that even if each of the robust models has been trained only using one kind of adversarial examples, all of them are relatively robust to the other kinds of adversarial examples. For instance the *fgsm* model is more robust than the standard model to the LL-FGSM, PGD and LL-PGD adversarial examples. In other words, adversarially trained models are robust to several unforeseen adversarial attacks.

Now, we want to estimate the robustness to synthetic corruptions of these models robust to adversarial examples. Namely, we measure the robustness of the considered models to all the synthetic corruptions displayed in Figure 3.2. The obtained results are provided in Table 3.2.

We observe that adversarial training does increase robustness to some specific synthetic corruptions such as *Gaussian noise* or *blur*. However, it also makes models less robust than the standard model to other corruptions such as *contrast loss*, *translation* or *color distortion*. On average, adversarially trained models are not more robust than the standard model to synthetic corruptions. In other words, it appears that increasing robustness to adversarial examples does not imply increasing robustness to synthetic corruptions.

Consequently, the hypothesis made in Section 3.1.1 appears to be wrong : learning features that are interpretable by humans, does not necessarily make models robust to corruptions towards which humans are not sensitive. Moreover, it is unlikely that solving the issue of adversarial robustness alone would enable to achieve robustness in a broad sense. Now, what about the reciprocal :

| Corru Model | gaus | art | obstr | blur | contr | bright | color | transl | rot | flip | mean |
|----------------|------|------|-------|------|-------|--------|-------|--------|------|------|------|
| standard | 0.81 | 0.81 | 0.95 | 0.78 | 0.87 | 0.94 | 0.72 | 0.89 | 0.94 | 0.78 | 0.85 |
| fgsm | 0.91 | 0.89 | 0.93 | 0.91 | 0.78 | 0.87 | 0.69 | 0.82 | 0.91 | 0.78 | 0.85 |
| fgsm_ll | 0.96 | 0.87 | 0.95 | 0.91 | 0.73 | 0.88 | 0.67 | 0.83 | 0.92 | 0.78 | 0.85 |
| pgd | 0.95 | 0.88 | 0.94 | 0.94 | 0.75 | 0.87 | 0.69 | 0.81 | 0.91 | 0.79 | 0.85 |
| pgd_ll | 0.94 | 0.85 | 0.95 | 0.93 | 0.75 | 0.86 | 0.61 | 0.83 | 0.92 | 0.79 | 0.84 |

Table 3.2 – APP scores of adversarially trained models computed using the synthetic corruptions displayed in Figure 3.2.

does increasing robustness to synthetic corruptions results in no augmentation in robustness to adversarial examples? We study this question in the following section.

3.2.4 Does Synthetic Corruption Robustness Transfer to Adversarial Attacks ?

We propose to complete the study carried out in the previous section by studying whether synthetic corruption robustness transfers to adversarial corruptions. To achieve it, we propose to build models robust to synthetic corruptions and to measure their robustness to adversarial examples.

To this end, we first train a ResNet-18 augmented with all the synthetic perturbations presented in Figure 3.2. We call it the *syn_rob* model. As expected, we see in Table 3.2 that this model is much more robust than the *standard* model to all the synthetic corruptions presented in Figure 3.2.

However, the *syn_rob* model could still be sensitive to unforeseen synthetic corruptions. Indeed, data augmentation is known to make neural networks robust only to the perturbations used in the augmentation process [Dodge2017b]. In other words, there is no guarantee that the *syn_rob* model is more robust than the *standard* one to unforeseen synthetic corruptions.

We conduct a second experiment to verify this last point. We train several ResNet-18, each augmented with all corruptions of Figure 3.2 but one. For instance, the *no-gaussian* model is trained on all the considered synthetic corruptions but the Gaussian noise. Formally, the *no- ϕ* model has been submitted to all the considered synthetic corruptions but ϕ . We compute the robustness of each *no- ϕ* model on samples corrupted with the ϕ perturbation. We compare each robustness score found this way with the one obtained using the *standard* model : results are presented in Table 3.3.

| Model \ ϕ | gaus | art | obstr | blur | contr | bright | color | trans | rot | flip |
|----------------|------|------|-------|------|-------|--------|-------|-------|------|------|
| standard | 0.81 | 0.81 | 0.95 | 0.78 | 0.87 | 0.94 | 0.72 | 0.89 | 0.94 | 0.78 |
| no- ϕ | 0.83 | 0.84 | 0.96 | 0.82 | 0.94 | 0.97 | 0.76 | 0.92 | 0.95 | 0.78 |

Table 3.3 – APP scores of the *standard* and *no- ϕ* models computed with unforeseen corruptions. Each score of the second line refers to the robustness of the *no- ϕ* model towards the ϕ perturbation.

It appears that even if each *no- ϕ* model has never seen the ϕ corruption, it is slightly more

robust to it than the *standard* model. It is true that the robustnesses to two very different corruptions are generally not correlated : robustness to obstructions does not imply robustness to Gaussian noise. Yet, here, it appears that data augmentation using a sufficiently large and diverse set of synthetic corruptions, can increase robustness to unforeseen ones. The *syn_rob* model is trained on even more corruptions than the *no- ϕ* models. Then, just as the *no- ϕ* models, *syn_rob* should be more robust to unforeseen synthetic corruptions than the *standard* model.

Now, we can tackle our original issue, i.e., we want to verify whether being more robust to synthetic corruptions, make also the *syn_rob* model more robust than the *standard* model to adversarial examples. To answer this question, we measure the robustness of the two models to all the adversarial examples presented in Section 3.2.2. The obtained results are shown in Table 3.1. We see that the *syn_rob* model is not more robust than the standard model to adversarial examples : their robustness scores are almost equal for all considered attacks. Then, it appears that increasing robustness to synthetic corruptions does not imply increasing robustness to adversarial examples.

Considering both this result and the one obtained in Section 3.2.3, we conclude that adversarial and synthetic corruption robustnesses are largely independent.

3.2.5 Concurrent Works

Some investigations studying the links between synthetic and adversarial robustnesses have been conducted at the same time or after our works. Some of the results of these studies are interesting to contrast with ours and enable to refine the conclusion drawn at the end of Section 3.2.4.

Consistently with our studies, recent experiments carried out in [Gulshad2021], show that adversarial robustness does not imply synthetic corruption robustness. However, it is also argued that the reciprocal can be true : data augmentation with synthetic corruptions can slightly improve robustness to adversarial examples. We believe their results differ from ours because the severity of the adversarial attacks used in their study is from one to two orders of magnitude lower than the one we used. In other words, these attacks are barely harmful, and we think that they do not correspond to the sever adversarial perturbations an attacker may use in practical cases. Besides, they only used very small datasets : one order of magnitude smaller than the one we used. Further investigations are required to better understand why their results partially contrast with ours.

Another interesting result is that it has been shown that corruptions damaging high-frequency information in images (according to the Fourier frequency domain) are correlated in terms of robustness [Yin2019]. These conclusions are consistent with the results obtained in Table 3.2. Indeed, adversarial training, Gaussian noise and blur alter high-frequency information in images, and we have observed in our study that these perturbations are indeed correlated in terms of robustness. Yin et al. also argue that robustness to corruptions damaging high-frequency features in images does not transfer to corruptions damaging low-frequency information. This is consistent with the results found in Table 3.2 : adversarial training does not improve robustness to color distortions or contrast losses.

These observations also explain why Kireev et al. obtain results that apparently contrast with ours : they argue that adversarial training increases robustness to synthetic corruptions [Kireev2021]. But, the synthetic corruptions used in their study are the ones found in ImageNet-C [Hendrycks2019a], which means that twelve out of fifteen used corruptions are image transformations that damage high-frequency information in images [Yin2019]. Then, it is not surprising to observe a link between robustness to the ImageNet-C corruptions and adversarial attacks that also modify high-frequency features in images. Conversely, in our study, only three out of ten corruptions damage high-frequency features in images.

On the whole, we think that a more nuanced picture of the correlation between synthetic and adversarial robustnesses, could be drawn by adding to the study we carried out, adversarial examples that affect low-frequency features in images, such as the ones proposed in [Guo2020]

and illustrated in Figure 3.3.

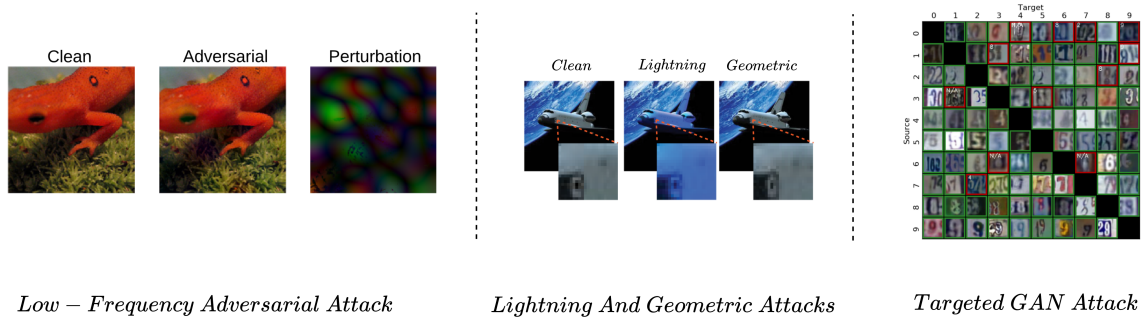


Figure 3.3 – Illustrations of the low-frequency adversarial attacks presented in [Guo2020] (on the left), of adversarial examples that are not bounded pixel-wise [Liu2019] (at the center) and GAN based attacks [Song2018b] (on the right).

We think it would be also interesting to study the robustness to corruptions that are at the boundary between synthetic corruptions and adversarial ones. Such corruptions have been proposed to enlarge the scope of adversarial perturbations in [Xiao2018] and [Liu2019]. These attacks include small well-chosen rotations, translations or brightness variations into the adversarial attack scope (see Figure 3.3). Besides, the adversarial training paradigm of GAN can be leveraged to generate adversarial examples that damage other features than traditional adversarial attacks (not only high-frequency information) [Song2018b]. Generators used in such GAN, target the weaknesses of some trained classifiers and automatically find relevant training examples to increase their robustness. The advantage of this strategy is that GAN generators can create original attacks that are not restricted to the traditionally use L_p bounded adversarial examples : an example is illustrated in Figure 3.3. All these corruptions could be used to erase the discrepancy between adversarial and synthetic corruptions, and enable to better understand how to address robustness in a broad sense.

3.2.6 Conclusion

Adversarial examples have been widely studied these last years. Which is legitimate because these attacks are a security threat and also raise conceptually interesting questions. But adversarial samples robustness is just a part of neural network robustness. Indeed, our study shows that robustness to these attacks is largely orthogonal with robustness to synthetic corruptions. In addition to that, some experimental results suggest that robustness to adversarial examples is not either correlated with robustness to natural corruptions [Orhan2019], [Hendrycks2021b]. Yet, we notice that there are more papers tackling the problem of adversarial robustness than papers addressing robustness to non-adversarial distribution shifts. However, we think that non-adversarial corruptions such as natural corruptions, are even more an obstacle to the deployment of neural networks in practical applications than adversarial examples.

Consequently, we advocate for research that consider both adversarial and non-adversarial robustness. While this can be more challenging than addressing robustness to a narrower range of corruptions, studying both aspects could help to get new insights on how neural networks work and why they are not robust to distribution shifts in general. Besides, we think that industrials should consider including both adversarial and non-adversarial corruptions in their robustness specifications, to make sure that they do not omit any important aspect of neural network robustness. New defenses are needed to achieve robustness to such wider ranges of corruptions. Therefore, we propose in the next Section (3.3), a first attempt to meet such complex robustness requirements.

3.3 Addressing Both Adversarial and Synthetic Corruption Robustnesses using M-TLAT.

3.3.1 Introduction

We demonstrated that adversarial robustness and robustness to synthetic corruptions are largely independent. Then, it is not surprising to see that these two aspects of neural network robustness are most of the time studied separately. They appear to be different issues that bring into play very different concepts. However, in some critical industrial applications (for instance autonomous vehicles), we want neural networks to be robust to as much corruptions as possible, including both adversarial examples and synthetic corruptions. Unfortunately, we showed in Table 3.2, that making models more robust to adversarial attacks, can cause a drop of accuracy for some specific synthetic corruptions such as contrast loss or translations. Similarly, some other studies show that data augmentation on some synthetic corruptions such as Gaussian noise, can significantly reduce robustness to other synthetic corruptions such as fog modeling [Yin2019].

In production context, an increase in robustness to a specific corruption may not be valuable if it reduces the robustness to other unforeseen corruptions. Why an industrial would spend resources increasing robustness to some corruptions if he or she suspects that it could also reduce performances on unforeseen corruptions that may be encountered by deployed models? In some cases, it is preferable to use robustness interventions that increase slightly the robustness to a large number of corruptions, rather than interventions that largely increase robustness to a narrow range of corruptions but also reduce robustness to other kinds of corruptions.

Unfortunately, existing robustness interventions often address a very specific kind of corruptions (see Section 2.2). Moreover, these defenses may have harmful effects on unforeseen corruptions, which can make deployed models get even lower performances than models trained without any robustness intervention. To make this situation less likely to occur, we need to enlarge the range of corruptions used to estimate robustness. The more a proposed robustness intervention is shown to be efficient for a wide diversity of corruptions, the less this defense is likely to be full of holes.

In addition to that, we also note that some robustness interventions, especially the ones that increase adversarial robustness, may reduce accuracy on clean samples [Su2018], [Zhang2019], [Tsipras2019]. Again, in some cases, we would prefer to deploy models not robust to adversarial examples to preserve their high accuracy on clean samples.

Consequently, it appears that proposed defenses are largely disconnected from what is required in some industrial applications. Then, as a first step to address these issues, we propose in this Section (3.3), one of the first defense shown to increase robustness to a large diversity of both synthetic corruptions and adversarial attacks, without reducing accuracy on clean samples.

Our defense is a data augmentation approach called M-TLAT. It is a combination of Mixup [Zhang2018] and a new kind of adversarial training called Targeted Labeling Adversarial Training (TLAT). In a nutshell, the idea of this adversarial training is to label target adversarial examples with soft labels that contain information about the used target. We will show that M-TLAT can increase robustness of image classifiers to nineteen unforeseen synthetic corruptions and five adversarial attacks, without reducing accuracy on clean samples. Interestingly, this shows that making models robust to a few well-chosen corruptions can make them robust to a wide range of diverse corruptions.

3.3.2 Combining Mixup with Targeted Labeling Adversarial Training : M-TLAT

3.3.2.1 Mixup

M-TLAT is based on a first component called Mixup [Zhang2018]. Let us consider the couples (x_i, y_i) and (x_j, y_j) , where x_i and x_j are images of the training set and y_i and y_j are their associated

one-hot encoding labels. Mixup [Zhang2018] is a data augmentation process (illustrated in Figure 3.4) that interpolates linearly samples and labels using the following equation :

$$\begin{aligned} x_{mix} &= \lambda * x_i + (1 - \lambda) * x_j \\ y_{mix} &= \lambda * y_i + (1 - \lambda) * y_j \end{aligned} \quad (3.1)$$

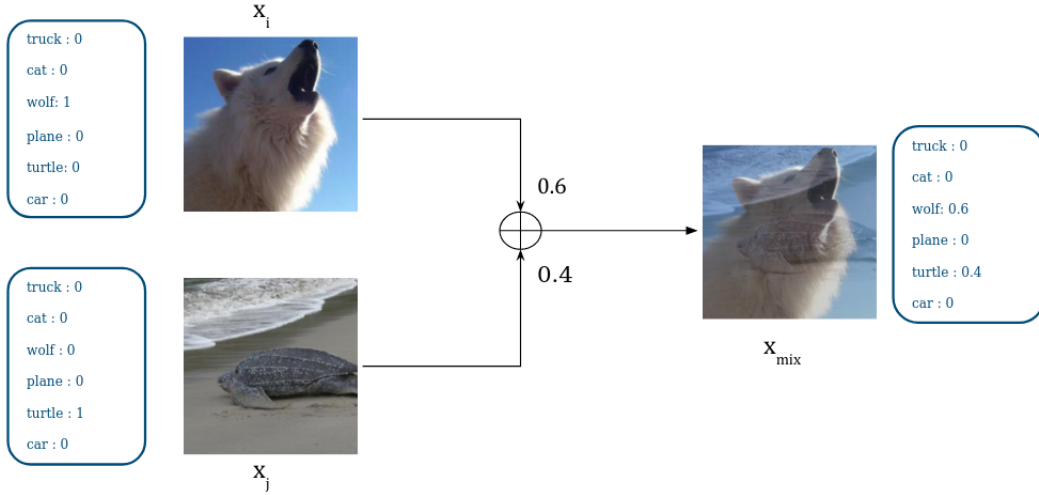


Figure 3.4 – Illustration of the mixup augmentation process.

where λ is drawn from a Beta distribution defined by a hyperparameter α : $\lambda \sim Beta(\alpha, \alpha)$. This augmentation strategy encourages trained models to have a linear behavior in-between training samples [Zhang2018], [Verma2019]. In theory, it makes neural network outputs vary more smoothly across a large range of out-of-distribution samples. In practice, Mixup increases clean accuracy, makes neural networks less sensitive to corrupted labels in training sets and slightly improves robustness to adversarial examples. In the ablation study carried out in Section 3.3.4.2, we show that Mixup also has a significant influence on neural network robustness.

Augmenting datasets by interpolating samples of training sets have been largely studied [Tokozume2018], [Inoue2018]. Among the most successful methods that use this idea, we note Manifold Mixup that interpolates hidden representations instead of interpolating only at the input layer [Verma2019]. Mixup Inference [Pang2020] proposes to use Mixup during inference phases to degrade perturbations that may corrupt input images. Directional Adversarial Training and Untied Mixup are alternative policies to get interpolation ratios when mixing samples and labels [Archambault2019]. The proposed M-TLAT algorithm uses the standard Mixup, but it is not incompatible with the other interpolation strategies mentioned in this paragraph.

3.3.2.2 TLAT

M-TLAT relies on a second data augmentation procedure : adversarial training, which consists in adding adversarial examples into training sets [Goodfellow2015], [Madry2018]. More precisely, we propose a new adversarial training paradigm, which is presented in this Section. We consider an unperturbed sample x_{clean} of size S and its associated label y_{clean} . We can get a new training adversarial couple (x_{adv}, y_{adv}) , by corrupting x_{clean} solving the following optimization problem :

$$\begin{aligned} x_{adv} &= x_{clean} + \underset{\delta \in [-\epsilon, \epsilon]^S}{\text{arg max}} \{L(x_{clean} + \delta, y_{clean}, \theta)\} \\ y_{adv} &= y_{clean} \end{aligned} \quad (3.2)$$

Where θ are the parameters of the attacked model. L is a cost function like the cross-entropy function. The value ϵ defines the amount of the introduced adversarial perturbation. As suggested in [Kurakin2017a], adversarial training is even more efficient when it uses adversarial examples that target a specific class y_{target} . The augmentation strategy becomes :

$$\begin{aligned} x_{adv} &= x_{clean} - \underset{\delta \in [-\epsilon, \epsilon]^S}{\operatorname{arg\,max}} \{L(x_{clean} + \delta, y_{target}, \theta)\} \\ y_{adv} &= y_{clean} \end{aligned} \quad (3.3)$$

One advantage to use target adversarial examples during training is to prevent label leaking [Kurakin2017b], i.e., trained models cannot assess ground truth labels from introduced adversarial corruptions. We propose to improve this augmentation strategy by introducing y_{target} in the labeling of the adversarial examples. In particular, we propose to mix the one-hot encoding ground-truth labels of the original samples with the one-hot encoding target labels used to craft the adversarial examples :

$$\begin{aligned} x_{adv} &= x_{clean} - \underset{\delta \in [-\epsilon, \epsilon]^S}{\operatorname{arg\,max}} \{L(x_{clean} + \delta, y_{target}, \theta)\} \\ y_{adv} &= (1 - \epsilon) * y_{clean} + \epsilon * y_{target} \end{aligned} \quad (3.4)$$

We call this augmentation strategy Targeted Labeling Adversarial Training (TLAT). The *arg max* part can be approximated using various methods of adversarial example generations. We show in our study that TLAT is efficient even when using a simple attack such as target FGSM [Kurakin2017a], which is computed with the following expression :

$$x_{adv} = x_{clean} - \epsilon * \operatorname{sign}(\nabla_{x_{clean}} L(x_{clean}, y_{target}, \theta)) \quad (3.5)$$

As for traditional adversarial trainings, models trained with TLAT are trained on both clean samples and adversarial samples [Kurakin2017a]. We show in Section 3.3.4.3, that the advantage of TLAT is to make models have a high clean accuracy compared to the models trained with a standard adversarial training algorithm.

TLAT uses soft labels instead of one-hot encoding labels. Using soft labels is a recurrent idea in the field of neural network robustness. As mentioned above, Mixup interpolates training labels to generate soft labels [Zhang2018]. Label smoothing replaces the zeros of one-hot encoding labels by a smoothing parameter $s > 0$ and normalizes the high value so that the distribution still sums to one [Szegedy2016]. Distillation learning uses the logits of a trained neural network to train a second neural network [Papernot2016]. The second network is enforced to make smooth predictions by learning on soft labels. Bilateral training generates soft labels by adversarially changing training labels [Wang2019b]. It uses the gradient of the cost function of attacked models to generate adversarial labels. Models trained on both adversarial examples and adversarial labels are encouraged to have a small gradient magnitude which makes them more robust to adversarial examples.

The originality of TLAT is to use target labels of adversarial attacks as a component of soft labels. We show Section 3.3.4.3 that this idea helps to increase accuracy of trained models on clean samples.

3.3.2.3 M-TLAT

The idea of M-TLAT is to generate new training couples by applying sequentially the TLAT perturbations (3.4) after the Mixup interpolations (3.1) :

$$\begin{aligned} x_{mtlat} &= x_{mix} - \underset{\delta \in [-\epsilon, \epsilon]^S}{\operatorname{arg\,max}} \{L(x_{mix} + \delta, y_{target}, \theta)\} \\ y_{mtlat} &= (1 - \epsilon) * y_{mix} + \epsilon * y_{target} \end{aligned} \quad (3.6)$$

As illustrated in Figure 3.5, $x_{m\text{tlat}}$ contains features that come from three distinct sources : two clean images and an adversarial perturbation that targets a specific class. The label $y_{m\text{tlat}}$ contains the class and the weight associated with each source of features. These weights are determined by the values λ and ϵ . A model trained with M-TLAT is not only constraint to predict the classes that correspond to the three sources. It also has to predict the weight of each source within the features of $x_{m\text{tlat}}$. We believe that being able to predict the class and the weighting of the three sources requires a subtle understanding of the features present in images. In practice, being trained with augmented couples $(x_{m\text{tlat}}, y_{m\text{tlat}})$ makes neural networks more robust.

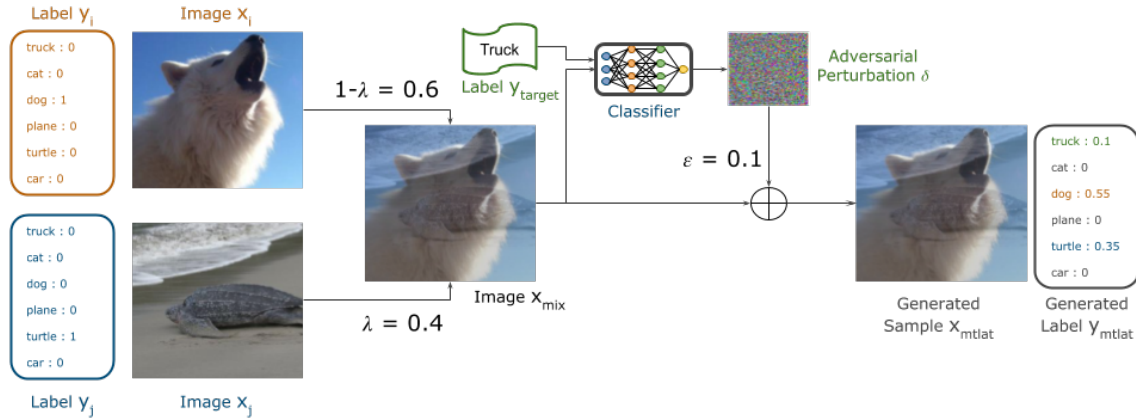


Figure 3.5 – Visualization of the generation of a new training couple using M-TLAT.

The expressions 3.6 are the essence of our augmentation strategy. The whole process of one training step with M-TLAT is provided in the Algorithm 1 description. As recommended in [Kurakin2017a], training minibatches contain both adversarial and non-adversarial samples. In our algorithm, non-adversarial samples are obtained using a standard Mixup interpolation, and adversarial samples are crafted by combining Mixup and TLAT.

An other approach combines Mixup and adversarial training [Lamb2019]. Their method called Interpolated Adversarial Training (IAT) is different from M-TLAT for two main reasons. Most importantly, they do not use the labeling strategy of TLAT. Basically, their adversarially corrupted samples are labeled using the standard Mixup interpolation while our labels contain information about the amount and the target of the used adversarial examples. Secondly, we interpolate images before adding the adversarial corruptions. On the contrary they adversarially corrupt images before mixing them up. In practice, we get better adversarial robustness when we proceed in our order. Besides, proceeding in their order doubles the number of adversarial perturbations to compute : it increases the training time. In Section 3.3.4.1, we compare the robustness of two models trained with these approaches.

3.3.3 Experimental Set-up

3.3.3.1 Robustness Benchmark

To estimate robustness of models, we constitute a benchmark of both synthetic and adversarial corruptions.

Synthetic Corruption Selection. The used synthetic corruption benchmark contains the fifteen ImageNet-C corruptions (more details about this benchmark can be found in Section 2.3.3). In addition to these perturbations, four other corruptions are included called : *rotation*, *translation*, *color distortion* and *occlusion*. These four corruptions affect features that are largely different from the ones altered by the ImageNet-C corruptions. More precisely, they damage low-frequency information in images on contrary to most of the ImageNet-C corruptions [Yin2019]. Then, to increase the diversity of the considered corruptions, we add those four corruptions to our pool. The

Algorithm 1 One training step of the M-TLAT algorithm.

Require: θ the parameters of the trained neural network

Require: L a cross entropy function

Require: $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4) \sim \text{Dataset}$

Require: $\lambda_1, \lambda_2 \sim \text{Beta}(\alpha, \alpha)$

Require: $\varepsilon \sim U[0, \varepsilon_{max}]$ where U is a uniform distribution and ε_{max} is the maximum perturbation allowed

Require: $y_{target} \sim U[0, N]$ where N is the number of classes

Require: $optim$ an optimizer like Adam or SGD

$$x_{mix1} = \lambda_1 * x_1 + (1 - \lambda_1) * x_2$$

$$y_{mix1} = \lambda_1 * y_1 + (1 - \lambda_1) * y_2$$

$$x_{mix2} = \lambda_2 * x_3 + (1 - \lambda_2) * x_4$$

$$y_{mix2} = \lambda_2 * y_3 + (1 - \lambda_2) * y_4$$

$$x_{mlat} = x_{mix2} - \varepsilon * \text{sign}(\nabla_{x_{mix2}} L(x_{mix2}, y_{target}, \theta))$$

$$y_{mlat} = (1 - \varepsilon) * y_{mix2} + \varepsilon * y_{target}$$

$$loss_1 = L(x_{mix1}, y_{mix1}, \theta)$$

$$loss_2 = L(x_{mlat}, y_{mlat}, \theta)$$

$$loss = loss_1 + loss_2$$

$$gradients = \nabla_{\theta} loss$$

$optim.update(gradients, \theta)$ The optimizer updates θ according to the computed gradients

implementation details along with the severity ranges of these corruptions are given in Figure 3.2.

Adversarial Attack Selection. Adversarial examples are added to the selected synthetic corruptions to complete the robustness benchmark. Following the recommendations in [Carlini2019], we carefully choose these adversarial examples. Firstly, we use adversarial examples in both the white-box and black-box settings. Secondly, to compute the bound of adversarial attacks, we use two different metrics : the L_{∞} and the L_2 norms. Thirdly, we employ targeted and untargeted attacks. Fourthly, several amounts of adversarial perturbations are used. Finally, the selected adversarial examples are not used during any training of our experiments. We build a set of adversarial examples to cover all the conditions mentioned above. Namely, we use PGD with $\varepsilon = 0.04$ as a white-box L_{∞} bounded attack [Madry2018]. We generate targeted adversarial attacks by using PGD_LL using the same corruption amount as PGD. We use MI_FGSM with $\varepsilon = 0.04$ and $\varepsilon = 0.08$ as black-box attacks [Dong2018]. The decay rate of accumulated gradients computed during the MI_FGSM algorithm is set to 1. PGD, PGD_LL and MI_FGSM are computed over 10 iterations. We use the Carlini-Wagner attack (CW_2) as a L_2 white-box bounded attack [Carlini2017]. We perform the optimization process of this attack with 40 iterations and a confidence score of 50. A VGG network [Simonyan2015] trained on the same training set and the exact same training hyperparameters as the attacked neural networks, is used to craft the black-box attacks. Formal definitions of these attacks can be found in Section 2.3.2.2.

3.3.3.2 Training Details

In the experimental Section 3.3.4, models are trained during 90 epochs with a batch size of 256. The Adam optimizer [Kingma2015] is used with a learning rate of 0.002 and a weight decay of 10^{-4} . At the end of the epoch number 30, 60 and 80, the learning rate is divided by 10. The cost function used is the cross entropy. These hyperparameters are used for all trainings.

All models are trained and tested using ImageNet. Because of a limited computational budget,

Table 3.4 – Effect on robustness of M-TLAT and comparison with IAT.

(a) APP scores computed using the ImageNet-C corruptions.

| | | Clean | Gauss | Shot | Impul | Defocus | Glass | Motion | Zoom | Snow | Fog | Frost | Bright | Contr | Elastic | Pixelate | Jpeg |
|---------|----------|-------------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------------|
| ResNet | standard | 73.3 | 0.17 | 0.17 | 0.12 | 0.25 | 0.35 | 0.41 | 0.47 | 0.31 | 0.48 | 0.34 | 0.78 | 0.34 | 0.73 | 0.65 | 0.80 |
| | IAT | 73.2 ⁻ | 0.47 ⁺ | 0.46 ⁺ | 0.42 ⁺ | 0.51 ⁺ | 0.65 ⁺ | 0.58 ⁺ | 0.68 ⁺ | 0.49 ⁺ | 0.78 ⁺ | 0.66 ⁺ | 0.79 ⁺ | 0.78 ⁺ | 0.84 ⁺ | 0.82 ⁺ | 0.63 ⁻ |
| | M-TLAT | 73.9⁺ | 0.56 ⁺ | 0.54 ⁺ | 0.52 ⁺ | 0.41 ⁺ | 0.61 ⁺ | 0.58 ⁺ | 0.63 ⁺ | 0.49 ⁺ | 0.59 ⁺ | 0.61 ⁺ | 0.82 ⁺ | 0.58 ⁺ | 0.85 ⁺ | 0.94 ⁺ | 0.95⁺ |
| ResNeXt | standard | 76.4 | 0.25 | 0.25 | 0.20 | 0.28 | 0.37 | 0.44 | 0.48 | 0.36 | 0.53 | 0.37 | 0.79 | 0.36 | 0.75 | 0.72 | 0.74 |
| | IAT | 74.7 ⁻ | 0.46 ⁺ | 0.44 ⁺ | 0.43 ⁺ | 0.53 ⁺ | 0.67 ⁺ | 0.62 ⁺ | 0.70 ⁺ | 0.51 ⁺ | 0.73 ⁺ | 0.69 ⁺ | 0.81 ⁺ | 0.80 ⁺ | 0.85 ⁺ | 0.83 ⁺ | 0.59 ⁻ |
| | M-TLAT | 76.5⁺ | 0.57 ⁺ | 0.55 ⁺ | 0.54 ⁺ | 0.44 ⁺ | 0.64 ⁺ | 0.60 ⁺ | 0.66 ⁺ | 0.52 ⁺ | 0.68 ⁺ | 0.67 ⁺ | 0.86 ⁺ | 0.70 ⁺ | 0.85 ⁺ | 0.95 ⁺ | 0.95⁺ |

(b) APP scores computed using the additional synthetic corruptions.

| | | Obstru | Color | Trans | Rot |
|---------|----------|-------------------------|-------------------|-------------------------|-------------------|
| ResNet | standard | 0.74 | 0.76 | 0.78 | 0.68 |
| | IAT | 0.71 ⁻ | 0.89 ⁺ | 0.75 ⁻ | 0.72 ⁺ |
| | M-TLAT | 0.75⁺ | 0.86 ⁺ | 0.79⁺ | 0.74 ⁺ |
| ResNeXt | standard | 0.75 | 0.82 | 0.82 | 0.72 |
| | IAT | 0.72 ⁻ | 0.90 ⁺ | 0.77 ⁻ | 0.71 ⁻ |
| | M-TLAT | 0.76⁺ | 0.89 ⁺ | 0.82 | 0.74 ⁺ |

| | | pgd $\epsilon=0.04$ | pgd_ll $\epsilon=0.04$ | cw_l2 | mi_fgsm $\epsilon=0.04$ | mi_fgsm $\epsilon=0.08$ |
|---------|----------|------------------------|---------------------------|-------------------------|----------------------------|----------------------------|
| ResNet | standard | 0.00 | 0.00 | 0.00 | 0.58 | 0.34 |
| | IAT | 0.01 ⁺ | 0.08 ⁺ | 0.84 ⁺ | 0.87 ⁺ | 0.78 ⁺ |
| | M-TLAT | 0.08 ⁺ | 0.45⁺ | 1.00⁺ | 0.96 ⁺ | 0.87 ⁺ |
| ResNeXt | standard | 0.00 | 0.00 | 0.00 | 0.58 | 0.33 |
| | IAT | 0.01 ⁺ | 0.11 ⁺ | 0.95 ⁺ | 0.87 ⁺ | 0.81 ⁺ |
| | M-TLAT | 0.09 ⁺ | 0.38⁺ | 0.99⁺ | 0.95 ⁺ | 0.87 ⁺ |

we used a subset of ImageNet built on 100 randomly chosen classes. The only pre-processing used is the resizing of images to the 224*224 format.

We call models trained without any data augmentation the *standard* models. We observed that the highest clean accuracy for the models trained with mixup is reached when $\alpha = 0.4$, so we use this value in all experiments. The adversarial examples used in trainings are crafted using FGSM with $\epsilon \sim U[0, 0.025]$. The range of pixel values of images in our experiments is $[0, 1]$.

3.3.4 Results

3.3.4.1 Robustness of Models trained with M-TLAT

In the first experiment, we respectively train two Resnet-50 [He2016], using the M-TLAT and IAT algorithm [Lamb2019]. To underline that the obtained results are consistent from one architecture to another, we also train two other models that have a different architecture using the same algorithms. This additional architecture is the ResNeXt-50 with the 32x4d template [Xie2017], which contains computational blocks very different from the ones of a standard ResNet-50.

The hyperparameters used are detailed in Section 3.3.3.2. The training of these models took two dozens of hours using a single GPU Nvidia Tesla V100. We also train one Resnet-50 and one ResNeXt-50 with the IAT algorithm [Lamb2019]. We compute the APP scores of the trained models towards all the perturbations of the benchmark constituted in Section 3.3.3.1. The obtained results are reported in Table 3.4.

In this table, the *Clean* column contains the accuracy of models on the not-corrupted test set. Each of the other columns contains an APP score computed using one perturbation of the benchmark. For the ImageNet-C corruptions, the displayed scores correspond to the mean APP score computed with the five severity levels associated to the considered corruption. To better visualize the effect of the augmentation algorithms, we use either an index "-" or an index "+", to signify if a model is less or more robust than the standard model to the considered corruption.

We observe in Table 3.4 that the models trained with M-TLAT are slightly more accurate than the standard models on clean images. They are also more robust than the standard models to every single tested synthetic corruption. We see that using M-TLAT makes neural networks much

more robust to the CW_2 and PGD_LL attacks. It also makes models less sensitive to black-box adversarial examples. We observe that the robustness gain for the PGD attack is less important. This can be explained by the fact that using FGSM during training is known to poorly defend models against iterative adversarial attacks such as PGD [Kurakin2017b]. The robustness of the M-TLAT models towards PGD can likely be increased by replacing FGSM by an iterative adversarial attack. But as a side effect, this would increase significantly the training time and would also reduce the accuracy on clean samples [Madry2018].

By comparing the performances of our algorithm to the ones of the IAT algorithm in Table 3.4, we note that IAT tends to reduce clean accuracy. Besides, it does not increase robustness to all the synthetic corruptions of the benchmark (noticeable results are in bold). In particular, it significantly decreases robustness towards the Jpeg perturbation. Besides, models trained with IAT are significantly less robust to adversarial examples than the M-TLAT models.

To our knowledge, M-TLAT is the first proposed data augmentation approach that is able to increase the robustness to every single synthetic corruption and adversarial example of a large set of diverse perturbations, without reducing clean accuracy.

3.3.4.2 Complementarity between Mixup and TLAT

To better understand the effect of each constituent of the M-TLAT algorithm, we proceed to an ablation study. Two ResNet-50 are respectively trained with the Mixup and TLAT data augmentations. We report their APP scores computed with the perturbations of our benchmark in Table 3.5.

First, we notice that Mixup causes an increase of clean accuracy, which is coherent with observations made in [Zhang2018]. On the contrary, TLAT makes trained models less accurate on clean data. But those two effects seem to cancel each other : the M-TLAT model and the standard model have comparable clean accuracies as observed in Table 3.4.

In Tables 3.5a and 3.5b, we observe that Mixup makes the trained model more robust than the standard model to all the tested synthetic corruptions but the *Motion Blur*, *Pixelate* and *Jpeg* corruptions. We observe in Table 3.5c that Mixup has a little influence on adversarial robustness, with either a slight increase or decrease of robustness depending on the considered attack.

On the other hand, TLAT makes models much more robust to adversarial attacks. Indeed, the TLAT model is much more difficult to attack with the tested black-box adversarial examples or with the CW_2 attack. It is also significantly more robust to PGD_LL and slightly more robust to PGD. For synthetic corruptions, the effect of TLAT is very contrasted. Concerning the noise and blur corruptions (the seven first corruptions of Table 3.5a), the TLAT model is much more robust than the standard model. For some other synthetic corruptions like *Fog* or *Contrast*, the TLAT augmentation decreases significantly the obtained robustness scores.

In a nutshell, models trained with M-TLAT are much more robust to adversarial examples thanks to the contribution of TLAT. However, for synthetic corruptions, Mixup and TLAT are remarkably complementary. Concerning the few corruptions for which Mixup has a very negative effect on robustness (*Jpeg* and *Pixelate*), TLAT has a strong positive effect. Similarly, for the *Fog* and *Contrast* corruptions, TLAT makes models less robust while Mixup makes them much more robust.

The ablation study indicates that both components are important to increase robustness to a large diversity of perturbations.

3.3.4.3 Ablation on Labeling Strategy in Adversarial Trainings

Adversarial trainings increase adversarial robustness of trained models, but they also reduce their accuracy on clean samples [Tsipras2019], [Kurakin2017b]. In this section, we want to show that TLAT decreases less clean accuracy of trained models than traditional adversarial trainings.

Table 3.5 – Influence on robustness of the Mixup and TLAT data augmentations.

(a) APP scores computed using the ImageNet-C corruptions.

| | | Clean | Gauss | Shot | Impul | Defocus | Glass | Motion | Zoom | Snow | Fog | Frost | Bright | Contr | Elastic | Pixelate | Jpeg |
|--------|----------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|--------|-------------------|-------------------|-------------------|-------------------|
| ResNet | standard | 73.3 | 0.17 | 0.17 | 0.12 | 0.25 | 0.35 | 0.41 | 0.47 | 0.31 | 0.48 | 0.34 | 0.78 | 0.34 | 0.73 | 0.65 | 0.80 |
| | Mixup | 74.9 ⁺ | 0.28 ⁺ | 0.28 ⁺ | 0.24 ⁺ | 0.25 | 0.38 ⁺ | 0.39 ⁻ | 0.53 ⁺ | 0.38 ⁺ | 0.79 ⁺ | 0.53 ⁺ | 0.78 | 0.75 ⁺ | 0.75 ⁺ | 0.58 ⁻ | 0.61 ⁻ |
| | TLAT | 69.4 ⁻ | 0.57 ⁺ | 0.54 ⁺ | 0.51 ⁺ | 0.43 ⁺ | 0.60 ⁺ | 0.56 ⁺ | 0.60 ⁺ | 0.41 ⁺ | 0.15 ⁻ | 0.41 ⁺ | 0.78 | 0.13 ⁻ | 0.84 ⁺ | 0.94 ⁺ | 0.97 ⁺ |

(b) APP scores computed using the additional synthetic corruptions.

| | | Obstru | Color | Trans | Rot |
|--------|----------|-------------------|-------------------|-------------------|-------------------|
| ResNet | standard | 0.74 | 0.76 | 0.78 | 0.68 |
| | Mixup | 0.75 ⁺ | 0.88 ⁺ | 0.79 ⁺ | 0.71 ⁺ |
| | TLAT | 0.69 ⁻ | 0.76 | 0.67 ⁻ | 0.66 ⁻ |

(c) APP scores computed using the adversarial examples.

| | | pgd $\epsilon=0.04$ | pgd_ll $\epsilon=0.04$ | cw_l2 | mi_fgsm $\epsilon=0.04$ | mi_fgsm $\epsilon=0.08$ |
|--------|----------|------------------------|---------------------------|--------------------|----------------------------|----------------------------|
| ResNet | standard | 0.00 | 0.00 | 0.00 | 0.58 | 0.34 |
| | Mixup | 0.00 | 0.00 | 0.202 ⁺ | 0.61 ⁺ | 0.22 ⁻ |
| | TLAT | 0.10 ⁺ | 0.74 ⁺ | 0.97 ⁺ | 0.98 ⁺ | 0.93 ⁺ |

To achieve it, we trained four ResNet-50 with different kinds of adversarial training algorithms. The first model is trained using untargeted FGSM and the second is trained using targeted FGSM with randomly chosen target. Both use adversarial examples labeled with ground-truth labels. We train another model with target FGSM, but regularized via label smoothing [Szegedy2016]. We call it the LS model. For this model, we use a smoothing parameter equal to ϵ , where ϵ is the amount of introduced FGSM perturbations. In other words, the one value of one-hot encoding vectors are replaced by $1 - \epsilon$ and zeros are replaced by ϵ/N , where N is the number of classes. The fourth model is trained using the TLAT algorithm. All models are trained with minibatches that contain both clean samples and adversarial examples. We measure the clean accuracy of those models : results are displayed in Table 3.6.

We see that TLAT is the adversarial training method that reduces the less the clean accuracy. This result shows that TLAT is important to preserve the clean accuracy of models trained with M-TLAT, all the while making them more robust to adversarial examples.

Using soft labels in trainings is known to help models to generalize [Papernot2016], [Szegedy2016]. Here, we want to make sure that the usage of soft labels is not the main reason of the high clean accuracy obtained when using TLAT. To achieve it, we compare the performances of the TLAT and LS models. Even if the LS model also uses soft labels during trainings, it performs worse than the TLAT model. Consequently, the good performances of TLAT are not only due to the usage of soft labels. We provide below an interpretation to better understand the high clean accuracies obtained using M-TLAT.

Table 3.6 – Comparison between the accuracy on clean samples of the TLAT model and the one of models trained with other kinds of adversarial trainings.

| | standard | FGSM | target-FGSM | LS | TLAT |
|--------------------|----------|------|-------------|------|------|
| Clean accuracy (%) | 73.3 | 65.8 | 68.3 | 67.1 | 69.4 |

Interpretation. The TLAT augmentation is motivated by the works of Ilyas et al [Ilyas2019]. In their study, they reveal the existence of brittle yet highly predictive features in the data. Neural networks largely rely on those features even if they are nearly invisible to human eye. They are called non-robust features. Interestingly, Ilyas et al. show that models using only non-robust features when completing tasks, can still generalize well to unseen data. This is illustrated in Figure 3.6.

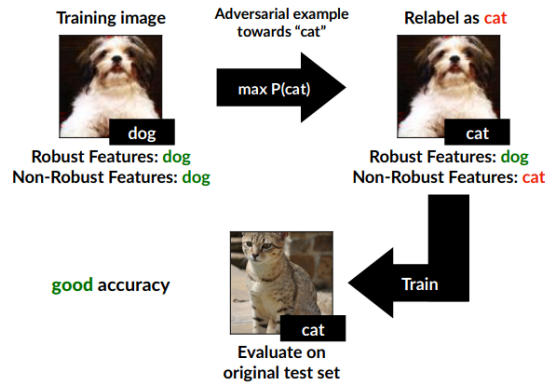


Figure 3.6 – Illustration of the experiment revealing the existence of non-robust predictive features (scheme extracted from [Ilyas2019]). A model trained only on images mislabeled according to human perception (dog images labeled with "cat"), can still make accurate predictions on unseen data (it can recognize cats). In other words, the image on the upper right corner contains features that are never entangled in natural images (dog appearance with brittle invisible features of cats).

They use this phenomenon to interpret adversarial vulnerability. Adversarial attacks are small perturbations, so they mainly affect non-robust features. They can especially make those brittle features anti-correlated with ground-truth labels. Since neural networks largely rely on non-robust features, their behavior is completely disturbed by adversarial attacks.

In adversarial trainings, neural networks encounter adversarial examples containing non-robust features uncorrelated with ground-truth labels. Trained models are then constrained to less rely on non-robust features. This can explain the success of adversarial training : adversarially trained models give less importance to non-robust features so they are much more difficult to attack with small perturbations.

Despite its efficiency, adversarial training generally causes a decrease in clean accuracy [Kurakin2017b], [Tsipras2019], [Zhang2019]. One possible reason could be that adversarial patterns in training adversarial examples are not coherent with their associated ground-truth label. Consequently, an adversarially trained model encounters samples for which features are not completely coherent with labeling.

The proposed method tries to make labeling of target adversarial examples more correlated with their non-robust features. Targeted adversarial attacks make non-robust features of samples correlated with their associated target classes. So, instead of labeling attacked samples only with ground-truth labels, M-TLAT introduces a part related to target classes. Therefore, trained models still learn the true original class of corrupted samples, but labels used for learning are more correlated with the non-robust features of these adversarial examples. We believe this could be the reason why our method has better performances than traditional target adversarial training in practice.

3.3.5 Concurrent Works

In this section, we present the few methods that have also been shown to increase robustness to adversarial examples and synthetic corruptions without reducing clean accuracy.

The first kind of succeeding approaches consists in using self-supervised learning on a very large amount of data to increase robustness [Mahajan2018], [Hendrycks2019b], [Xie2020b]. While these methods increase drastically clean accuracy and robustness to synthetic corruptions, the gain on adversarial examples is not significant. We note that the computational cost of these methods is extremely high : in some practical applications, such a high computational power may not be

available.

Another successful method is Manifold Mixup [Verma2019]. This data augmentation process consists in interpolating samples in batches at randomly chosen layer in neural networks. When an interpolation occurs at the input layer, Manifold Mixup is equivalent to a standard mixup. We note that Manifold Mixup has only been tested on small datasets : cifar and svhn, and we do not know to what extent it would work for a large dataset such as ImageNet.

Adversarial Logit Pairing [Kannan2018] is also known to increase robustness to both adversarial examples and synthetic corruptions. It encourages trained models to output similar logits for adversarial examples and their clean counterparts. However, this method reduces the accuracy of trained models on clean samples.

Neural networks have been shown to overly rely on high-frequency information (texture) in images [Geirhos2019]. Some methods have thus been proposed to make neural networks rely more on low-frequency information (shapes, positions of objects, etc.). In particular, a recently proposed approach changes texture of training images, and labels them consistently with the new texture introduced [Li2021b]. For example, a chimpanzee with a lemon texture is labeled with an interpolation of the chimpanzee and lemon labels (see Figure 3.7). This method increases robustness to both adversarial and synthetic corruptions, but also raises clean accuracy.

Similarly to M-TLAT, this method generates images that contain features that are generally uncorrelated (here a chimpanzee shape with a lemon texture), and it labels these images coherently with the unusual combinations of features they contain. We think it would be interesting to investigate to understand more deeply why such label and image combination strategies are particularly efficient to increase neural network robustness.

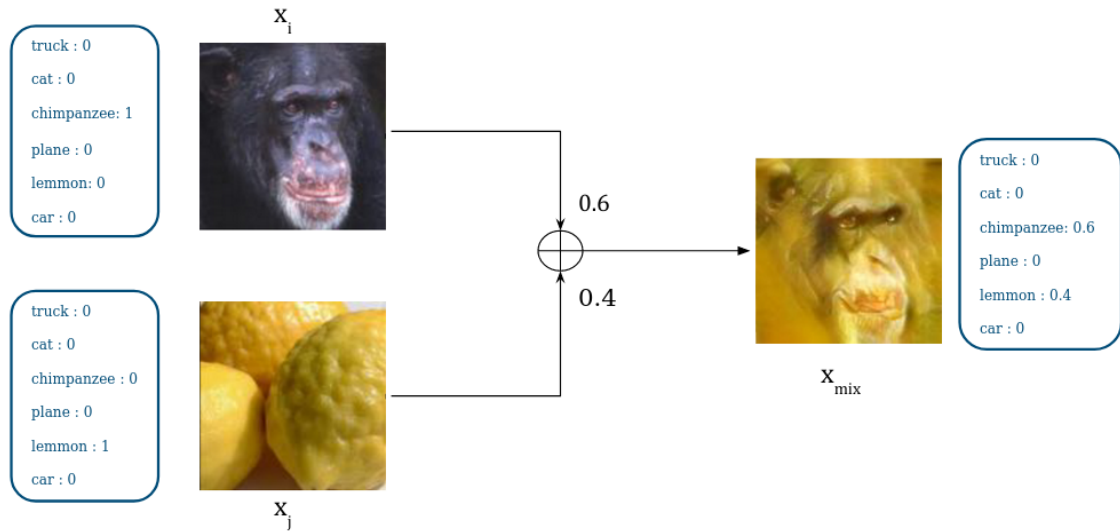


Figure 3.7 – Illustration of the shape-texture debiased training process proposed in [Li2021b].

3.3.6 Conclusion

We propose a new adversarial augmentation strategy called TLAT. We showed that models trained with TLAT have a better accuracy on clean samples than models trained with a standard adversarial training algorithm. The idea of TLAT is to interpolate the target labels of adversarial examples with the ground-truth labels. This operation is computationally negligible and can be used in any trainings with target adversarial examples in order to improve clean accuracy of trained models.

By combining TLAT with Mixup, we get a data augmentation strategy that increases robustness of neural networks to a large set of unforeseen synthetic corruptions and adversarial examples.

The experiments carried out suggest that the effect of M-TLAT is always positive : it increases the robustness to any tested corruption without reducing clean accuracy.

By proposing M-TLAT, we show that it is possible to address robustness in a broader sense. Instead of focusing on a specific kind of corruptions, we tested the robustness of trained models on very diverse and numerous corruptions. Benchmarking neural networks this way remains relatively rare in current robustness studies, yet, we think that doing so reduces the chances of encountering harmful unforeseen corruptions in real-world applications.

3.4 Conclusion

In this Chapter, we showed that robustness to adversarial examples generally does not transfer to synthetic corruptions and conversely. This result means that both aspects should be considered when trying to make neural networks robust to any kind of corruptions. In particular, when designing models in some practical applications (as described in diagram 1.3), we should make sure that established robustness specifications include the two kinds of corruptions.

Overall, our results are in line with several recent studies emphasizing that robustness has multiple faces that should all be considered when studying neural network robustness [Geirhos2020], [Hendrycks2020a], [Taori2020].

Some of these studies recommend to use natural corruptions (defined in Section 2.1.3) when estimating robustness. Indeed, while synthetic and adversarial corruptions form a convenient proxy to study robustness to unforeseen corruptions, most of the distribution shifts that industrial currently face when trying to deploy neural networks are natural corruptions. These corruptions are thus to be considered when establishing robustness specifications to be met when designing models to be deployed in the real-world. Robustness to natural corruptions is then studied afterwards in the manuscript, in Section 4.3.

Obviously, including more diverse corruptions in robustness specifications makes them harder to meet. But, we showed in Section 3.3 that it is possible to build a robustness intervention (M-TLAT) that increases robustness to very diverse corruptions, including both adversarial and synthetic ones, which is rather encouraging.

As detailed in Section 3.3.4.2, one component of our defense mostly increases robustness to synthetic corruptions (Mixup) while the other component mostly increases robustness to adversarial attacks (TLAT). We note that this is consistent with the idea that adversarial and synthetic robustnesses are independent attributes, which can then hardly be addressed by using a single simple idea. Yet, it would be interesting to see if standalone methods could be proposed to address these two aspects of neural network robustness simultaneously. Such innovative defenses could bring new ideas on how to make neural networks robust to distribution shifts in general.

4

Measuring Robustness Using Synthetic Corruptions

Contents

| | | |
|------------|--|-----------|
| 4.1 | Introduction | 48 |
| 4.1.1 | Synthetic Corruptions : Convenient Proxies to Study Robustness to Unforeseen Distribution Shifts | 48 |
| 4.1.2 | Synthetic Corruption Selections and Biases in Benchmarks | 48 |
| 4.1.3 | Organization | 49 |
| 4.2 | Using the Overlapping Score to Reveal Flaws in Synthetic Corruption Benchmarks | 49 |
| 4.2.1 | Introduction | 49 |
| 4.2.2 | The Corruption Overlapping Score | 50 |
| 4.2.2.1 | Definition | 50 |
| 4.2.2.2 | How to Compute an Overlapping Score ? | 51 |
| 4.2.3 | Experimental Set-up | 51 |
| 4.2.4 | Overlappings in ImageNet-C | 52 |
| 4.2.5 | Corruption Overlappings and Benchmark Balance | 53 |
| 4.2.6 | Corruption Overlappings and Benchmark Coverage | 53 |
| 4.2.7 | Concurrent Works | 54 |
| 4.2.8 | Conclusion | 56 |
| 4.3 | Building Synthetic Corruption Benchmarks More Predictive of Robustness to Natural Corruptions | 57 |
| 4.3.1 | Introduction | 57 |
| 4.3.2 | Experimental Set-up | 58 |
| 4.3.3 | Corruption Categories | 60 |
| 4.3.3.1 | Corruption Category Building Process | 60 |
| 4.3.3.2 | Empirical Evaluation of the Built Categories | 61 |
| 4.3.4 | Synthetic Corruption Selection Criteria | 63 |
| 4.3.4.1 | Number of Corruption Categories Represented in Benchmarks | 63 |
| 4.3.4.2 | Balance Among Categories | 64 |
| 4.3.4.3 | Corruption Benchmark Size | 65 |
| 4.3.5 | Comparisons with Existing Synthetic Corruption Benchmarks | 66 |
| 4.3.6 | Discussions | 66 |
| 4.3.7 | Conclusion | 67 |
| 4.4 | Conclusion | 68 |

4.1 Introduction

4.1.1 Synthetic Corruptions : Convenient Proxies to Study Robustness to Unforeseen Distribution Shifts

Some distribution shifts encountered by neural networks when deployed in the real-world, are extremely hard to model. For instance, viewpoint changes, which are recurrently faced in video surveillance applications, cannot be easily implemented with a hand-coded image transformation. Then, to study robustness to viewpoint changes, instead of modeling this corruption, we generally gather and label a lot of images captured from different viewpoints. These images can then be used to train some neural networks or estimate their robustness.

Unfortunately, viewpoint changes represent only one example of the numerous distribution shifts that may be faced by video surveillance systems (sensors changes, weather variations, etc.). In other words, it is prohibitively expensive to directly address neural network robustness to the diverse and complex distribution shifts they may encounter. Basically, this would require too many human annotations.

Synthetic corruptions can be seen as a solution to address this issue. Indeed, on contrary to naturally arising corruptions, synthetic ones are largely available through various libraries ([Jung2020], [Buslaev2020], [Alexander2021]). These libraries contain dozens of optimized image corruption implementations, ranging from motion blurs and brightness variations to shear transformations. Moreover, benchmarks containing images corrupted with synthetic corruptions are relatively easy and cheap to build.

Consequently, in academic research, studying robustness to synthetic corruptions is often seen as a convenient way to study robustness to unforeseen distribution shifts in general [Temel2017], [Djolonga2021]. The idea is that being able to make neural networks robust to unforeseen synthetic corruptions, should also provide insights on how to make them robust to other unforeseen distribution shifts. For example, finding a specific optimizer that increases robustness to various unforeseen synthetic corruptions, can mean that this optimizer helps neural networks to better generalize on new data in general (not only synthetically corrupted ones). Such findings would suggest investigations to better understand why, and thus get new knowledge about neural network robustness.

In production context, synthetic corruptions are also often used. Indeed, as explained above, it is prohibitively expensive to build a benchmark representative of the diversity of natural corruptions that can be encountered by neural networks to be deployed. Then, robustness specifications based on synthetic corruptions are used instead. Models to be deployed are selected depending on their performances on these specifications (see diagram 1.3). The intuition behind this approach is that a model robust to various synthetic corruptions it has not seen during training, should also be robust to the naturally arising corruptions it will encounter.

In both contexts synthetic corruption benchmarks are then used. A question that naturally arises then is how to constitute such benchmarks ?

4.1.2 Synthetic Corruption Selections and Biases in Benchmarks

A benchmark built on four digital noises, obviously does not give the same robustness estimations as a benchmark built on geometric transformations (rotations, translations, etc.). Then, why choosing some corruptions instead of others to estimate robustness? While this interrogation appears to be legitimate, it is barely or not discussed in submissions of new synthetic benchmarks [Karahana2016], [Temel2017], [Geirhos2019], i.e., corruptions are widely arbitrarily picked.

When it comes to autonomous car for instance, a benchmark built on geometric transformations, brightness variations and weather modelings, can be seen as a way to guarantee that vehicles are respectively robust to pedestrian positions, daylight variations and weather fluctuations. Unfortunately, it turns out that such predictions can be completely wrong because common sense is

quite limited in this matter. Indeed, we show in this Chapter that sets of arbitrarily chosen synthetic corruptions can provide very biased estimations of neural network robustness.

Interestingly, we note that this problematic is known and much more studied in the field of adversarial examples. Indeed, it has been demonstrated that non-discussed adversarial example selections, lead to benchmarks that make flawed robustness estimations [Athalye2018a]. By extension, robustness intervention conception processes that use such benchmarks to estimate robustness, lead to defenses that are also flawed. To tackle this issue, clear guidelines have been provided to constitute unbiased selections of adversarial examples to estimate robustness [Carlini2019].

In a similar fashion, we argue that recommendations to go beyond arbitrary selections when designing synthetic corruption benchmarks, should also be established. The aim of this Chapter is precisely to tackle this issue.

4.1.3 Organization

In this Chapter, we propose to follow the organization described here.

First, in Section 4.2, we demonstrate that several synthetic benchmark weaknesses can be revealed by studying overlappings between their corruptions. We define overlapping corruptions, as corruptions correlated in terms of robustness : being robust to one of them implies being robust to the other one and conversely. In Section 4.2.2, we propose a metric called the overlapping score able to estimate to what extent two corruptions overlap. Using this metric in Section 4.2.4, we study the overlappings in a currently used synthetic corruption selection. In Section 4.2.5 and 4.2.6, we show that overlapping scores can be used to reveal important biases in benchmarks.

To reduce these biases, we present in Section 4.3, a new methodology to select the synthetic corruptions to include in benchmarks. This method is based on the establishment of several synthetic corruption categories presented in Section 4.3.3. Each of these categories is shown to be representative of a different aspect of neural network robustness. In Section 4.3.4, we provide precise recommendations based on the established corruption categories, that enable to build less biased synthetic benchmarks. Finally, we show in Section 4.3.5, that benchmarks built following our criteria make robustness estimations more predictive of robustness to naturally arising corruptions.

4.2 Using the Overlapping Score to Reveal Flaws in Synthetic Corruption Benchmarks

4.2.1 Introduction

Robustness benchmarks built on synthetic corruptions have been proposed in various computer vision tasks such as face recognition [Karahan2016], traffic sign recognition [Temel2017], scene classification [Tadros2019], object recognition [Hendrycks2019a], object detection [Michaelis2019], image segmentation [Kamann2020] and saliency region detection [Che2020].

In all these studies, the way of selecting the synthetic corruptions to be included in benchmarks is barely or not discussed. Unfortunately, we show in this Section 4.2, that such arbitrary selections can make benchmarks have two important biases.

First, they can be unbalanced, i.e., they give much more importance to the robustness to some of their corruptions compared to others. For example a benchmark built on Gaussian noise, salt-pepper noise, shot noise, and rotations, is unbalanced because it rewards more noise robustness than robustness to geometric transformations.

The second important bias we identify, is that benchmarks do not cover some synthetic cor-

ruptions, i.e., being robust to these benchmarks provides no guarantee on the robustness to some synthetic corruptions. For instance, the benchmark given as example in the previous paragraph, does not cover the corruptions that change illumination in images.

In this Section 4.2, we demonstrate that benchmarks can be shown to have these two biases by studying the overlappings between their corruptions. Two corruptions overlap when the robustnesses of neural networks towards these corruptions are correlated. For instance, camera shake blur and defocus blur have been shown to be correlated in terms of robustness [Vasiljevic2016], so by definition they overlap. We propose in Section 4.2.2, a metric called the corruption overlapping score, designed to estimate to what extent two corruptions overlap.

We use this metric in Section 4.2.4, to study the overlappings between the corruptions of the widely used synthetic benchmark ImageNet-C [Hendrycks2019a]. Leveraging the obtained results, we demonstrate in Section 4.2.5 and 4.2.6, that our overlapping metric can be used to determine if a benchmark is unbalanced and if it covers some specific synthetic corruptions.

Overall, our results encourage to put into perspective the robustness gains obtained on some currently used synthetic benchmarks. Besides, it also questions the insights on neural network robustness got by studying these biased synthetic corruption benchmarks.

4.2.2 The Corruption Overlapping Score

4.2.2.1 Definition

To reveal some biases in benchmarks we first need to consider the notion of corruption overlapping. Two corruptions are said to overlap when robustness to one of these corruptions is correlated with robustness to the other. In this section, we propose a method to estimate to what extent two corruptions overlap.

We consider two neural networks m_1 and m_2 and two corruptions c_1 and c_2 . m_1 and m_2 are identical, and trained with exactly the same settings except that their training sets are respectively augmented with the corruptions c_1 and c_2 . A *standard* model is trained the same way but only with non-corrupted samples. We propose a method to measure to what extent c_1 and c_2 overlap. The idea of the method is to see whether a data augmentation with c_1 makes a model more robust to c_2 and conversely. To determine this, m_1 , m_2 , the *standard* model and a test set are used to compute the following expression :

$$(R_{c_1}^{m_2} - R_{c_1}^{standard}) + (R_{c_2}^{m_1} - R_{c_2}^{standard}) \quad (4.1)$$

R_c^m corresponds to the Accuracy Percentage Preserved (APP) score of m computed with c , i.e., it estimates the robustness of m to c (more details about this robustness metric are provided in Section 2.3.1).

Then, the first term of (4.1) measures if a model that fits exactly c_2 is more robust to c_1 than the *standard* model. Symmetrically, the second term measures if a model that fits exactly c_1 is more robust than the *standard* model to c_2 . The more fitting exactly c_1 implies being robust to c_2 and reciprocally, and the more we can suppose that the robustnesses to c_1 and c_2 are correlated in practice. In other words, the expression (4.1) gives an estimation of the overlapping between c_1 and c_2 .

To be more convenient, we would like to build a corruption overlapping score equal to 1 when $c_1 = c_2$, and equal to 0 when the robustnesses to c_1 and c_2 are not correlated at all. We propose a new expression that respects both conditions :

$$O_{c_1, c_2} = \max\left\{0, \frac{1}{2} * \left(\frac{R_{c_2}^{m_1} - R_{c_2}^{standard}}{R_{c_2}^{m_2} - R_{c_2}^{standard}} + \frac{R_{c_1}^{m_2} - R_{c_1}^{standard}}{R_{c_1}^{m_1} - R_{c_1}^{standard}} \right)\right\} \quad (4.2)$$

The expression (4.2) is a normalized version of (4.1). It measures the overlapping between two corruptions while respecting the conditions mentioned above. Indeed, if a data augmentation with

c_1 does not increase the robustness to c_2 at all and conversely, then the ratios in (4.2) are null or negative, so the whole overlapping score is maximized to zero. Besides, when $c_1 = c_2$, $R_{c_2}^{m_1} = R_{c_2}^{m_2}$ and $R_{c_1}^{m_2} = R_{c_1}^{m_1}$, so both ratios of (4.2) are equal to 1. Then, $O_{c_1, c_2} = 1$ when c_1 and c_2 completely overlap.

4.2.2.2 How to Compute an Overlapping Score ?

To get the overlapping score between c_1 and c_2 , we follow the method illustrated in Figure 4.1. This method has six steps, and requires to have a training set, a test set and three untrained models that share the same architecture (m_1 , m_2 and *standard*). Step (1), consists in using the corruptions c_1 and c_2 to get two training sets, each corrupted with one corruption. Then, the obtained corrupted sets are used to train the models m_1 and m_2 in step (2). The *standard* model is also trained during this step but only with non-corrupted samples. In step (3), similarly to step (1), we use c_1 and c_2 to get two corrupted versions of the test set. The accuracies of the three models on the three test sets are computed in step (4). The scores obtained are used in step (5), to get the robustness scores of each model for the corruptions c_1 and c_2 . The results obtained are used to compute the overlapping score between c_1 and c_2 in step (6). To be meaningful, an overlapping score should be computed with models that converged adequately on their training sets. In all experiments, we verify that all trained models converge.

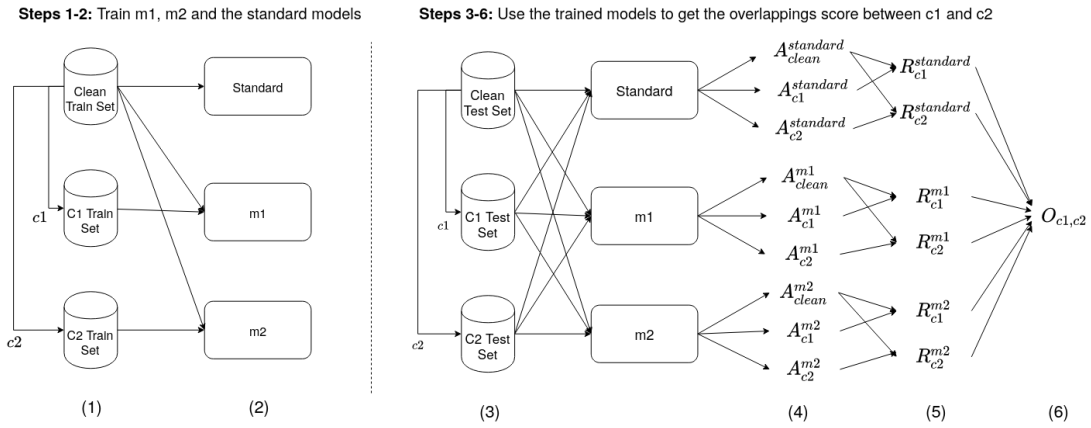


Figure 4.1 – Methodology used to compute the overlapping score between two corruptions c_1 and c_2 .

4.2.3 Experimental Set-up

Training Details. Models trained in our experiments fit ImageNet-100 : a subset of ImageNet that contains every tenth ImageNet class by WordNetID order [Deng2009]. In every training, we use the following parameters. The used optimizer is SGD with a momentum of 0.9. The used cost function is a cross-entropy function, and the weight decay is set to 10^{-4} . Models are trained for 40 epochs with a batch size of 256. The initial learning rate is set to 0.1 and is divided by 10 at epoch 20 and 30. In all training inferences, images are resized to the 224x224 format, and randomly horizontally flipped with a probability of 0.5. When a model is trained with data augmentation, half of the images of each training batch are transformed with a corruption, while the other half is not corrupted.

Robustness metric. We use the residual robustness metric in the overlapping score computation methodology described in Section 4.2.2.2. This metric is a simple and direct way of measuring robustness, which meets the needs of the method proposed to compute an overlapping score. We note that comparing the residual robustness scores of two models, requires to check that the ac-

curacies on clean samples of the two models are comparable. This condition is verified in all our experiments.

We also use the relative Corruption Error (CE) metric [Hendrycks2019a] in Sections 4.2.5 and 4.2.6. This robustness metric is used because in these sections, we compare models whose robustness has been stated with the CE score. So, to be consistent with previous studies, we measure robustness of those models using this metric.

Obviously, we never compare a CE score with a residual robustness score in any experiment. In other words, the usage of the two metrics is clearly separated, which avoids unfair comparisons. More details about the two metrics have been provided in Section 2.3.1.

4.2.4 Overlappings in ImageNet-C

We propose to use the corruption overlapping score to study ImageNet-C [Hendrycks2019a]: the most widely used synthetic corruption benchmark in the field of image classification. More precisely, we apply the methodology presented in Figure 4.1, with the training parameters defined in Section 4.2.3, to compute the overlapping scores between all the ImageNet-C corruptions. In Figure 4.2, we display the scores obtained using the ResNet-18 [He2016] and DenseNet-121 [Huang2017] architectures. Computing all these overlapping scores required to train thirty models and took two weeks with a single GPU Nvidia Tesla V100.

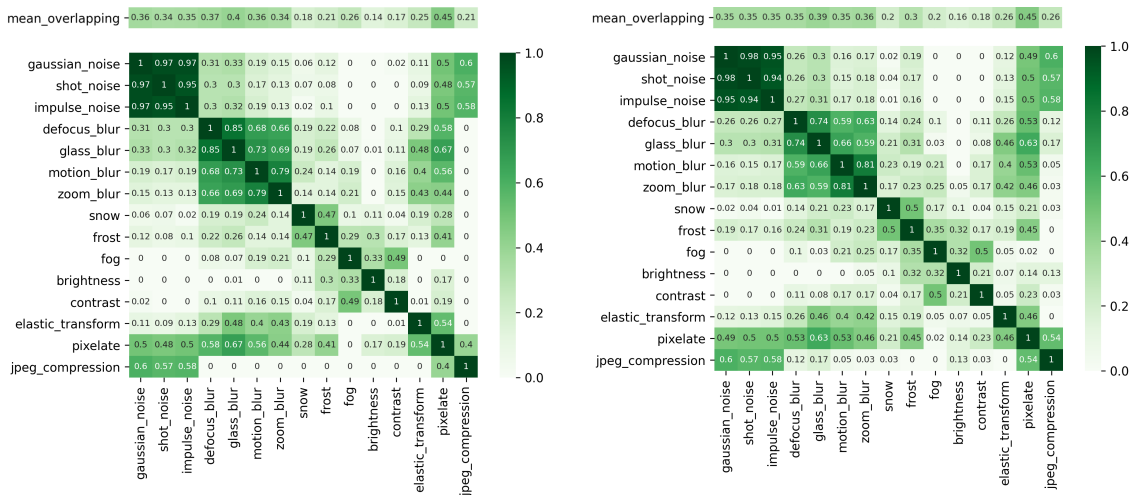


Figure 4.2 – Overlapping scores of ImageNet-C computed with the ResNet-18 (left) and DenseNet-121 (right) architectures.

We observe that the two matrices look very similar, i.e., for each corruption couple, the computed overlapping score displayed in the left matrix, is close to the one displayed in the right matrix. This suggests that our overlapping metric is network architecture independent.

The overlapping metric predicts that several groups of ImageNet-C corruptions such as noises are very correlated in terms of robustness. In other words, the metric predicts that being robust to one of the ImageNet-C noises implies being robust to the other ones. Interestingly, we observe that the corruptions that damage textures in images (blurs, noises, pixelate and jpeg) significantly overlap. This is consistent with the experiments carried out in [Yin2019], where it is argued that the robustnesses of neural networks to corruptions that alter high-frequency information in images are correlated.

Moreover, the obtained overlapping scores predict that there is redundancy in ImageNet-C because several of its corruptions appear to measure the same aspect of neural network robustness. Intuitively, such redundancy is not desirable because it makes the benchmark less informative. Indeed, knowing that a neural network is robust to shot noise, is not very valuable if you already

know that this model is robust to Gaussian noise and salt-pepper noise. In the following Section 4.2.5 and 4.2.6, we show that our metric is predictive, and confirms that in practice ImageNet-C gives poor estimations of robustness to synthetic corruptions.

4.2.5 Corruption Overlappings and Benchmark Balance

We consider (c_1, c_2) any corruption couple in a benchmark *bench*. We say that *bench* is unbalanced, when the robustness to *bench* (the mean robustness computed with the corruptions in *bench*) is more correlated with the robustness to c_1 than with the robustness to c_2 or vice versa. For instance, we show in this section that ImageNet-C is unbalanced because robustness to this benchmark is more correlated with robustness to blurs than with robustness to brightness variations. Yet, we think that in both academic and production contexts, being robust to different kinds of blurs is not more valuable than being robust to lighting condition variations. So, being unbalanced is in general not a desirable property because it makes benchmarks give biased estimations of neural network robustness.

We propose a simple method based on the overlapping score to determine whether a benchmark is unbalanced. If c_1 overlaps more with the other corruptions of *bench* than c_2 , then being robust to the corruptions of *bench* should be more correlated with being robust to c_1 than being robust to c_2 . So, if we show that a corruption contributes more to the total overlapping of a benchmark than another corruption, then this benchmark should be unbalanced. Following this idea, we compute the mean overlapping associated with each ImageNet-C corruption. The results are displayed in Figure 4.2. It appears that some corruptions contribute more to the total overlapping than others. For example, *pixelate* has a much higher mean overlapping score than *brightness*. In other words, the computed overlapping scores predict that ImageNet-C is unbalanced.

To verify this, we proceed to the following experiment : we consider a first group set_1 of eight corruptions, constituted with the three ImageNet-C noises, its four blurs and its *pixelate* corruption. This group corresponds to the corruptions that contribute the most to the overlappings in ImageNet-C. set_2 corresponds to the group containing the remaining ImageNet-C corruptions.

We compute the mean CE scores of the standard pretrained ResNet-50 of the torchvision library [Marcel2010], for both set_1 and set_2 . In Table 4.1, we compare these scores with the ones obtained with four models that have been shown to be robust to ImageNet-C called *SIN+IN* [Geirhos2019], *ANT^{3x3}* [Rusak2020], *Augmix* [Hendrycks2020b] and *DeepAugment* [Hendrycks2020a]. Indeed, these four models are more robust to the considered corruptions than the standard ResNet-50, but we observe that the robustness gain is much higher for set_1 than set_2 . *ANT^{3x3}*, *Augmix* and *DeepAugment* have been designed to be robust to synthetic corruptions, and ImageNet-C has been used as a reference by these models to estimate their robustness. The overlapping scores predict that ImageNet-C gives more importance to the robustness to some corruptions such as blurs than to the robustness to other corruptions such as brightness variations. So, it is not surprising to see that the studied models are more robust to blurs than to brightness variations : this is a direct consequence of the unbalance of ImageNet-C.

4.2.6 Corruption Overlappings and Benchmark Coverage

We consider that a corruption c is covered by a benchmark when being robust to this benchmark implies being robust to c . In practice, we want benchmarks to cover as many corruptions as possible. For instance, if a contrast loss corruption is not covered by a benchmark, then a model could be robust to this benchmark without being robust to contrast losses. So this benchmark omits an important aspect of neural network robustness. Therefore, finding corruptions that are not covered by a benchmark reveals weaknesses of this benchmark and suggests improvements.

We propose a simple method based on the overlapping score to show that a corruption c is not covered by a benchmark *bench*. The idea is to compute all the overlapping scores between c and

Table 4.1 – The CE scores of various models computed with the *border*, *obstruction*, *set₁* and *set₂* corruptions. The scores in parentheses indicate the CE gains compared to the standard ResNet-50.

| | mean_CE_Set1 | mean_CE_Set2 | Border | Obstruction |
|--------------------|--------------|--------------|--------|-------------|
| Standard | 81 (0) | 73 (0) | 53 | 63 |
| SIN+IN | 71 (-10) | 68 (-5) | 56 | 69 |
| Augmix | 66 (-15) | 65 (-8) | 50 | 63 |
| ANT ^{3x3} | 60 (-21) | 68 (-5) | 58 | 71 |
| DeepAugment | 59 (-22) | 63 (-10) | 60 | 72 |

the *bench* corruptions. If these scores are all equal to zero, then being robust to *bench* should not imply being robust to *c*, that is to say *c* is not covered by *bench*.

To confirm the validity of our approach, we propose to use the overlapping metric to find corruptions that are not covered by ImageNet-C. To achieve it, we gather several corruptions (displayed in Figure 4.3) that are not in ImageNet-C. Then, we compute the overlapping scores between all these corruptions and all the ImageNet-C corruptions. We found two corruptions that do not overlap at all with any of the ImageNet-C corruptions : *border* and *obstruction* (all the overlapping scores computed with these two corruptions are strictly equal to zero).

Therefore, by definition, the overlapping scores predict that a model can be robust to all the ImageNet-C corruptions, without being robust to the *border* and *obstruction* corruptions.

To verify this, we compute the CE scores of the four models robust to ImageNet-C with these two corruptions, and we compare their results with the ones obtained using the standard ResNet-50 in Table 4.1. We see that *SIN+IN*, *ANT^{3x3}* and *DeepAugment* are less robust than the standard ResNet-50 to the two corruptions. Then, in practice, being robust to ImageNet-C does not imply being robust to *border* and *obstruction*, i.e., these two corruptions are not covered by ImageNet-C.

In concrete terms, this means that ImageNet-C does not provide any robustness guarantee towards these two corruptions. Therefore, the predictions made using the overlapping score are verified, which confirms our hypothesis that this metric can be used to find corruptions not covered by benchmarks.

4.2.7 Concurrent Works

An other metric called the Minimal Sample Distance (MSD) has been proposed to measure to what extent two corruptions are correlated in terms of robustness [Mintun2021]. To explain how this metric works, we consider two synthetic corruptions c_1 and c_2 . The idea of MSD, consists in computing the mean of the feature vectors extracted by a neural network on samples corrupted with c_1 . This mean vector is noted mv_1 . Then, the feature vectors extracted by the same neural network, on the same images but corrupted with c_2 , are gathered. Among these feature vectors, the closest to mv_1 is picked and noted v_2 . The computed MSD score, corresponds to the L_2 distance between mv_1 and v_2 .

We note that on contrary to the overlapping score, the MSD metric is not symmetric, i.e., $MSD(c_1, c_2) \neq MSD(c_2, c_1)$. Therefore, their metric cannot be considered as a distance metric. This makes the predictions made using MSD less interpretable and less usable than the ones of our metric. In particular, not being a distance metric makes it formally harder to classify or cluster corruptions using the MSD score.

It is also important to note that they only apply their metric on small images such as the ones












| Corruption Name | Illustration | Description | Severity Range |
|------------------|---|---|--|
| Quantization |  | Quantize the pixel values into a few quantization levels | Number of levels: 4 to 9 |
| Thumbnail Resize |  | Apply a bilinear interpolation twice. Firstly to reduce the size of the image. Secondly to get back to its original size. | Reduction factor: 1.1 to 3.25 |
| Artifacts |  | Add artifacts into the image. The artifacts are made with a small dotted lines | Number of artifacts: 15 to 170 |
| Obstruction |  | One randomly chosen square area is filled with a unique pixel value | Size of the square 47 to 125 |
| Border |  | Fill the border of images with a unique pixel value | Thickness of the corrupted area: 9 to 46 pixels |
| Translation |  | One horizontal and one vertical translation | Number of pixels translated for both translations: 15 to 62 |
| Shear |  | Horizontal Shear | Maximum pixel translation due to the shear: 0.39 |
| Rotation |  | Clockwise or anticlockwise rotation | Degrees of the rotation: 7 to 50 |
| Backlight |  | Split the image into two areas and select a value. Add this value to all pixel of one of the area and subtract the value for the other. | Value added or subtracted: 0.11 to 0.44 |
| Color Distortion |  | Add a unique value to all pixels of one of the RGB channel | Value added: 0.09 to 0.40 |
| Gray Scale |  | Interpolate the image with its gray scale version | Interpolation factor associated with the gray scale image: 0.49 to 1 |

Figure 4.3 – Illustrations and implementation details of various image transformations not included in ImageNet-C.

in CIFAR-10 [Krizhevsky2009]). For such images, textures play a lesser role in model decisions compared to the large images we use. Yet, texture damaging corruptions are used in both our experiments and theirs. Therefore, robustness correlations between corruptions measured using our metric can hardly be compared to the ones measured using MSD.

Despite all that, we note that the experimental results obtained using MSD, indicate that corruptions close in terms of human perception, are correlated in terms of robustness. This is consistent with our results displayed in Figure 4.2 : the couples of overlapping corruptions, for instance Gaussian noise and salt-pepper noise or motion blur and zoom blur, are made of corruptions fairly similar according to human perception. Consequently, the results of the MSD metric along with ours, are in line with previous works [Heusel2017], that show that neural networks are quite successful at making similarity estimations between images that align with human perception.

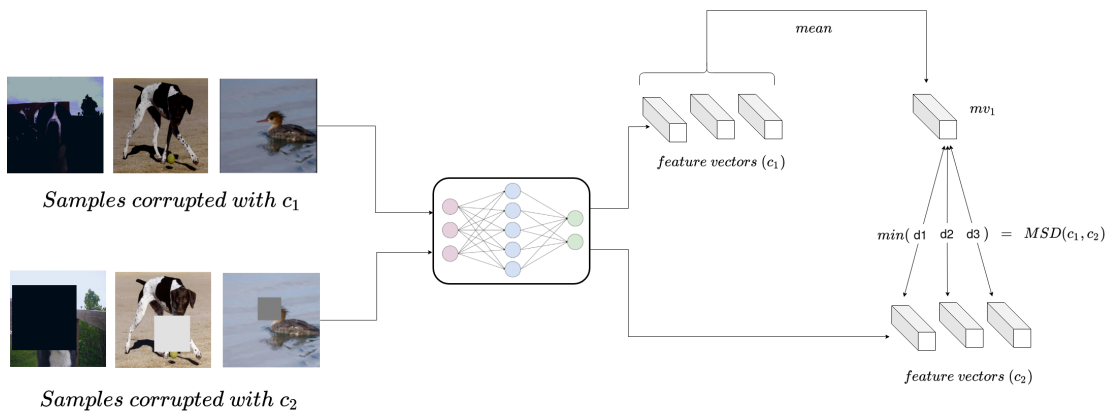


Figure 4.4 – Diagram describing how to compute the Minimal Samples Distance between two samples.

4.2.8 Conclusion

In Section 4.2.2, we proposed a metric called the overlapping score, which estimates to what extent two corruptions are correlated in terms of robustness. We showed that this metric is predictive and can be used to reveal flaws in synthetic corruption benchmarks. More precisely, it can be used to determine whether a benchmark is unbalanced, i.e., whether it gives too much importance to some corruptions it contains compared to others. It can also be used to show that a corruption is not covered by a benchmark, i.e., it does not provide any robustness guarantee towards this corruption.

By studying the overlappings in ImageNet-C in Section 4.2.4, we showed that this benchmark is unbalanced. More precisely, it gives much more importance to the robustness to texture damaging corruptions to the detriment of other kinds of synthetic corruptions. We showed in Sections 4.2.5 and 4.2.6, that these flaws can have prejudicial consequences on models we build, which is all the more harmful as ImageNet-C is widely used. Indeed, we demonstrated that several image classifiers, that are considered robust and used ImageNet-C as a reference to estimate their robustness, are biased towards the same corruptions as ImageNet-C.

In addition, we note that there are several ImageNet-C derivatives in other computer vision fields such as object detection [Michaelis2019] or segmentation [Kamann2020]. These benchmarks contain exactly the same corruptions as ImageNet-C. Then, it is very likely that these benchmarks have the same biases as ImageNet-C, even if they are not used in the same kinds of application. Therefore, good performances of models on such benchmarks should also be taken

with a grain of salt. Further investigations would enable to precisely determine to what extent these ImageNet-C derivatives are biased.

Overall, it appears that arbitrary corruption selections lead to biased benchmarks and should be avoided. In the following Section 4.3, we provide concrete recommendations to build less flawed benchmarks .

4.3 Building Synthetic Corruption Benchmarks More Predictive of Robustness to Natural Corruptions

4.3.1 Introduction

Synthetic corruptions are rarely unforeseen in practical applications. Indeed, the synthetic corruptions that may be encountered by deployed neural networks are introduced by digital image processings, which are often known in advance. For instance, it is recurrent that images in production context are compressed, resized or quantized. Moreover, synthetic corruptions that may be encountered are not so diverse, and we just have to make sure that models are robust to these corruptions before deploying them. Therefore, robustness to synthetic corruptions is barely an issue compared to robustness to natural corruptions.

Unfortunately, as detailed in Section 4.1.1, naturally corrupted samples are very expensive to gather compared to samples corrupted using synthetic image transformations. Consequently, we cannot easily measure robustness of models towards all the natural corruptions they may face, because we cannot reasonably gather the excessive amount of labeled images that would be required to achieve it. Then, we need alternative methods to be able to estimate natural corruption robustness.

One appealing strategy is based on the consideration that similarly to natural distribution shifts, synthetic ones are made of corruptions that can largely make performances of neural networks shrink [Dodge2016], [Vasiljevic2016], [Azulay2019]. Therefore, one can assume that neural network sensitivities to synthetic corruptions and natural ones have common causes. So, intuitively, if we ensure that a model is robust to several unforeseen synthetic corruptions, then it is likely that it will also be robust to various natural corruptions it may encounter. The advantage of this approach is that easily accessible synthetic corruptions can be leveraged to estimate natural corruption robustness. In other words, this makes the issue cheaper to address.

In some specific cases, this strategy has been shown to be relevant. For instance, robustness to synthetic blurs is known to transfer to real-world blurs [Hendrycks2020a]. On the other hand, other experiments demonstrate that the robustness to some synthetic corruptions is completely uncorrelated with the robustness to some natural corruption benchmarks [Taori2020]. In summary, the circumstances under which robustness to synthetic corruptions transfers to natural corruptions are not really understood. Therefore, we do not really know whether using synthetic corruption benchmarks as a proxy to natural corruptions is a relevant approach.

We propose here to investigate this matter. Namely, we provide several experimental results throughout this Section 4.3, demonstrating that some corruption selections are more predictive of robustness to natural corruptions than others. Besides, we present a methodology to find these more predictive synthetic benchmarks. Our methodology is based on two steps respectively described in Sections 4.3.3 and 4.3.4.

Firstly, given an initial set of synthetic corruptions, we split this set into categories. These categories are built such as the corruptions belonging to the same category overlap (they are correlated in terms of robustness), while the corruptions belonging to different categories do not. Secondly, based on these categories, we identify 3 parameters to take into account while building a synthetic

corruption benchmark : (1) the number of represented corruption categories (2) the balance among categories (3) the size of benchmarks.

Using the proposed methodology, we build synthetic corruption benchmarks, that we show in Section 4.3.5 are particularly predictive of robustness to natural corruptions. Overall, benchmarks built using our guidelines, appear to be less biased than benchmarks made of arbitrarily selected corruptions.

4.3.2 Experimental Set-up

We provide here the implementation details of the experiments carried out in Sections 4.3.3, 4.3.4 and 4.3.5.

Training Details. All models trained in Section 4.3.3 fit the ImageNet-100 dataset : a subset of ImageNet that contains every tenth ImageNet class by WordNetID order [Deng2009].

These trainings are completed with the following hyperparameters. The used optimizer is SGD with a momentum of 0.9. The used cost function is a cross-entropy function associated to a weight decay set to 10^{-4} . Models are trained for 40 epochs with a batch size of 256. The initial learning rate is set to 0.1 and is divided by 10 at epochs 20 and 30. In all training inferences, images are resized to the 224x224 format, and randomly horizontally flipped with a probability of 0.5. When a model is trained with data augmentation, half of the images of each training batch are transformed with a corruption, while the other half is not corrupted; which is the standard procedure when training with data augmentation [Dodge2017b], [Lamb2019], [Rusak2020].

Natural Corruption Benchmarks. The experiments carried out in Sections 4.3.4 and 4.3.5, involve several natural corruption benchmarks, which we propose to describe here. ImageNet-A is a challenging benchmark, constructed by selecting images that are misclassified by various ResNet-50 architecture-based models [Hendrycks2021b]. ImageNet-Sketch [Wang2019a] is an alternative ImageNet validation set containing hand-drawn sketches. ImageNet-V2 [Recht2019] has been built by replicating the ImageNet construction process. Because of some statistical biases in the image selection [Engstrom2020], a distribution shift is observed between ImageNet and ImageNet-V2. ObjectNet [Barbu2019] is a set of images that contains objects that have been randomly rotated or taken with various backgrounds and viewpoints. ImageNet-R contains artistic renditions of ImageNet object classes [Hendrycks2020a]. Samples extracted from all these benchmarks are displayed in Figure 4.5.

Synthetic Corruptions. In addition to the natural corruption benchmarks, we use in our experiments two synthetic benchmarks : ImageNet-C and ImageNet-P. More information about both benchmarks is provided in Section 2.3.3.

In Section 4.3.3, some experiments also involve synthetic corruptions that are not in ImageNet-C nor ImageNet-P. We model them using the albumentations library [Buslaev2020]. The functions and parameters used to model these corruptions are provided in Table 4.2.

Robustness Metric. In all experiments, robustness is measured using the residual robustness metric. We note that comparing the residual robustness scores of two models, requires to check that the accuracies on clean samples of the two models are comparable. This condition is verified in all our experiments. More information about this metric can be found in Section 2.3.1.

Neural Network Selection. In the experiments of Sections 4.3.3, 4.3.4 and 4.3.5, we evaluate correlations in terms of robustness between benchmarks. To achieve this, we use a set of models that cover various neural network architectures, sizes and training methodologies. The idea is to obtain a set of models representative of the diversity of currently used image classifiers. Some of the selected models have been trained with data augmentation on either adversarial examples [Madry2018], [Engstrom2019], [Rusak2020] or synthetic corruptions [Yun2019], [Lim2019], [Geirhos2019], [Hendrycks2020b], [Hendrycks2020a]. Some others have trained leveraging a large amount of unlabeled data using self-supervised learning [Mahajan2018], [Yalniz2019], [Xie2020b], [Li2021a]. We also select models having non-standard architectures [Xie2020a],

4.3. Building Synthetic Corruption Benchmarks More Predictive of Robustness to Natural Corruptions

| Corruption Name | Albumentations Function | Parameters |
|---------------------|------------------------------------|---|
| Affines | IAAAffine | scale=(0.6, 0.8) |
| Blur | Blur | blur_limit=(5, 9) |
| Border | Compose(CenterCrop, PadIfNeeded) | (height=130, width=130),(height=224, width=224) |
| ChannelDropout | ChannelDropout | - |
| CoarseDropout | CoarseDropout | min_holes=16, max_holes=48 |
| ColorJitter | ColorJitter | brightness=0.7, contrast=0.7, saturation=0.7, hue=0.7 |
| Cutout | Cutout | max_h_size=150, max_w_size=150 |
| Downscale | Downscale | scale_min=0.45, scale_max=0.45 |
| ElasticTransform | ElasticTransform | border_mode=1, value=0 |
| Emboss | IAAEmboss | alpha=(1.0, 1.0) |
| Flip | Flip | - |
| GaussianBlur | GaussianBlur | blur_limit=(9, 13) |
| GaussNoise | GaussNoise | var_limit=(500.0, 1000.0), mean=0 |
| GridDistortion | GridDistortion | distort_limit=1 |
| HueSaturationValue | HueSaturationValue | hue_shift_limit=100, sat_shift_limit=150, val_shift_limit=100 |
| ImageCompression | ImageCompression | quality_lower=5, quality_upper=15 |
| InvertImg | InvertImg | - |
| ISONoise | ISONoise | color_shift=(0.2, 0.5), intensity=(0.5, 1.0) |
| JpegCompression | JpegCompression | quality_lower=5, quality_upper=15 |
| MedianBlur | MedianBlur | blur_limit=(5, 11) |
| MotionBlur | MotionBlur | blur_limit=(11, 17) |
| MultiplicativeNoise | MultiplicativeNoise | multiplier=(0.7, 1.3), elementwise=True |
| OpticalDistortion | OpticalDistortion | distort_limit=4, shift_limit=4 |
| Perspective | IAAPerspective | scale=(0.1, 0.3) |
| Posterize | Posterize | num_bits=2 |
| RandomBrightness | RandomBrightness | limit=0.9 |
| RandomContrast | RandomContrast | limit=1.5 |
| RandomFog | RandomFog | fog_coef_lower=0.8 |
| RandomGamma | RandomGamma | gamma_limit=(300, 1000) |
| RandomGridShuffle | RandomGridShuffle | grid=(4, 4) |
| RandomRain | RandomRain | shadow_roi=(0, 0, 1, 1), num_shadows_lower=2, num_shadows_upper=4, shadow_dimension=8 |
| RandomShadow | RandomShadow | - |
| RandomSnow | RandomSnow | - |
| RandomSunFlare | RandomSunFlare | src_radius=200 |
| RGBShift | RGBShift | r_shift_limit=200, g_shift_limit=200, b_shift_limit=200 |
| Rotate | Rotate | limit=180 |
| Sharpen | IAASharpen | alpha=(0.3, 0.6), lightness=(0.6, 1.0) |
| Solarize | Solarize | threshold=200 |
| ToSepia | ToSepia | - |
| Transpose | ShiftScaleRotate | shift_limit=0.7, scale_limit=0.0, rotate_limit=0.0 |

Table 4.2 – Albumentations functions and their associated parameters used to get the overlapping scores displayed Figure 4.7.

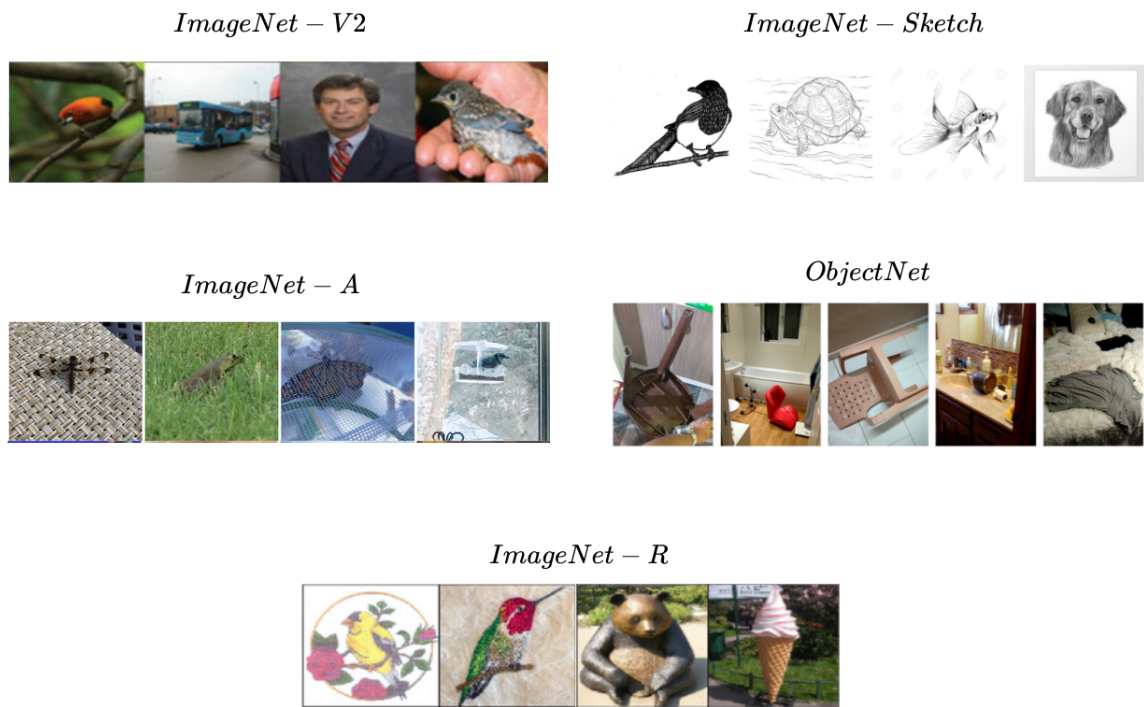


Figure 4.5 – Images extracted from ImageNet-V2 [Recht2019], ImageNet-Sketch [Wang2019a], ImageNet-A [Hendrycks2021b], ObjectNet [Barbu2019] and ImageNet-R [Hendrycks2020a].

[Vasconcelos2020]. The gathered models are displayed in Table 4.3. We also use the plain counterparts (trained without any robustness intervention) of these models.

4.3.3 Corruption Categories

4.3.3.1 Corruption Category Building Process

There are a lot of possible synthetic corruptions that can be included in a benchmark. Constructing a corruption benchmark requires to pick some corruptions among all possible candidates. Here, we consider a list of 40 candidate corruptions that are illustrated in Figure 4.6. An other list of corruptions could have been selected, but most of the corruptions usually included in existing benchmarks [Karahan2016], [Temel2017], [Hendrycks2019a] can be found in these candidates (blurs, noises, contrast loss, etc.). The implementation details of these functions can be found in Table 4.2.

The 40 considered corruptions form a heterogeneous set. It is difficult to determine the number and the kinds of corruptions to be included in a robustness benchmark a priori. Here, we propose a method to select groups of corruptions that make robustness estimations more correlated with robustness to natural corruptions.

The first step of our method is to compute the overlapping scores between candidate corruptions : here we use the 40 candidates described above. The overlapping scores are computed using the methodology described in Section 4.2.2.2, using the ImageNet-100 dataset, the ResNet-18 architecture [He2016] and the training hyperparameters described in Section 4.3.2.

Each corruption c is then associated with a vector that contains all the overlapping scores computed using c and any other corruptions. The second step is to split the candidate corruptions into categories, such as the overlapping score vectors of the corruptions belonging to the same category are correlated ; while the overlapping score vectors of the corruptions belonging to different



Figure 4.6 – Candidate corruptions displayed in the same order as the corruption names of Figure 4.7.

categories are not.

To achieve it, we cluster our candidates using their associated 40-dimensional vector of overlapping scores. We use the K-means algorithm increasing progressively the number of centroids K . We note that increasing K , raises on average the correlations between the overlapping vectors of the Same Category Corruptions (SCC), which is consistent with our goal ; but it also raises the correlations between the vectors of Different Category Corruptions (DCC), which is not desired. We choose to stop increasing K at $K = 6$, when the mean of all the Pearson correlation coefficients computed using SCC overlapping vectors becomes higher than 0.5. The obtained categories can be seen in Figure 4.7.

Interestingly, we observe that most of the obtained categories can be associated with a human visual perception interpretation. Indeed, *Category 2* to *6* in Figure 4.7, could be respectively called *geometric transformations*, *blurs*, *lightning condition variations*, *fine-grain artifacts* and *color distortions*. *Category 1* is harder to name because it contains heterogeneous corruptions according to human perception (see the first six images displayed in Figure 4.6). This category is quite different from the others and deserves its own discussion, which is provided in 4.3.6.

4.3.3.2 Empirical Evaluation of the Built Categories

To verify the relevance of the built corruption categories, we propose to conduct the following experiment. For each corruption c among the candidate corruptions displayed in Figure 4.6, we compute the residual robustness of the twenty-one models displayed in Table 4.3 with the ImageNet validation set corrupted with c . Each candidate corruption c is now associated with a vector that contains the twenty-one robustness scores computed using c . For each possible couple of candidate corruptions, we compute the Pearson correlation coefficient using the two robustness score vectors associated with the corruptions of the considered couple. The mean correlation obtained using SCC is 0.67, while the one obtained using DCC is 0.10. This experiment confirms the relevance of the built corruption categories : SCC are in practice correlated in terms of robustness while DCC are not.

By the way, it also adds up to the experimental results of Section 4.2, confirming that predictions made using the overlapping score are quite accurate.

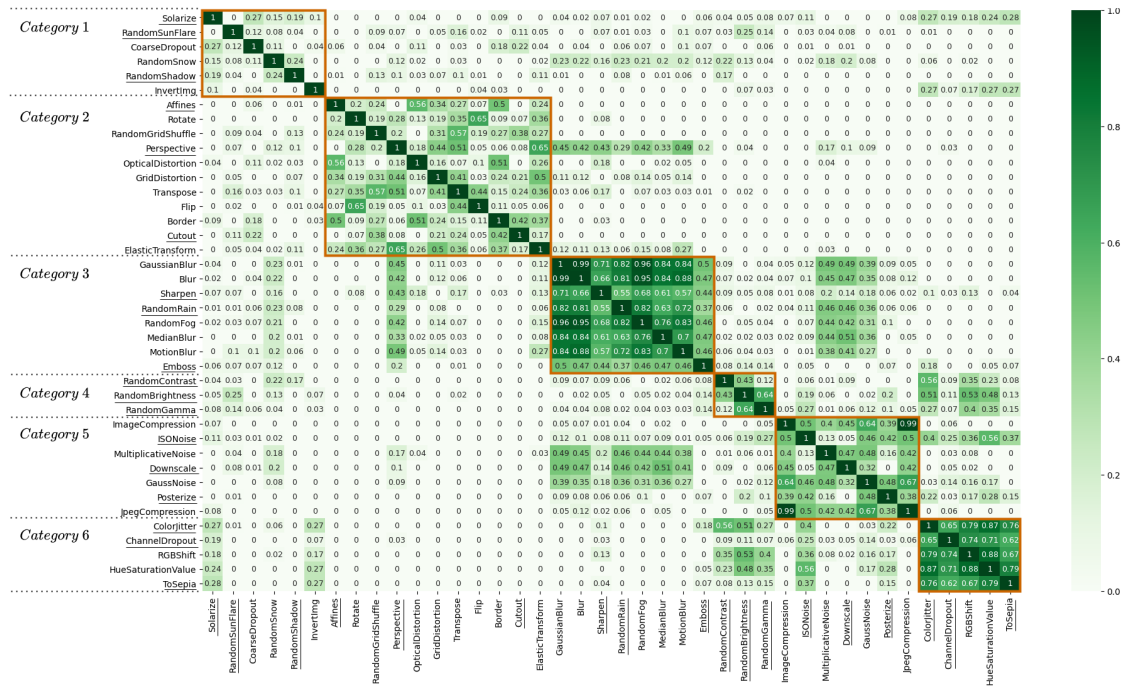


Figure 4.7 – Overlapping scores computed using all the possible candidate corruption couples. The scores computed with SCC are in the 6 orange squares : one square for each of the 6 categories.

| Models Trained with a Robustness Intervention | Plain Counterpart |
|---|--------------------|
| FastAutoAugment [Lim2019]; Worst-of-10 spatial data augmentation using the following transformation space : ± 3 pixels ± 30 degrees [Engstrom2019] | ResNet18 |
| ANT ^{3x3} [Rusak2020]; SIN Augmentation [Geirhos2019]; Augmix [Hendrycks2020b]; DeepAugment [Hendrycks2020a]; MoPro [Li2021a]; RSC [Huang2020]; Adversarial Training : $L_{inf}, \epsilon = 4/255$ [Madry2018] | ResNet-50 |
| Noisy Student Training [Xie2020b]; AdvProp [Xie2020a] | EfficientNet-0 |
| Anti-Aliased [Vasconcelos2020] | DenseNet-121 |
| Cutmix [Yun2019] | ResNet-152 |
| Weakly Supervised Pretraining [Mahajan2018]; Semi-Supervised Pretraining [Yalniz2019] | ResNeXt-101-32x16d |

Table 4.3 – Selected models trained with a robustness intervention. Their plain counterpart are also included in this set of models used to compute robustness correlations between benchmarks.

4.3.4 Synthetic Corruption Selection Criteria

We introduce the definition of some terms used in this section. The size of a benchmark is the number of corruptions this benchmark contains. Each time a benchmark *bench*, contains a corruption *c* that belongs to the corruption category *CC*, we say that *CC* is represented in *bench*; and *c* is called a representative of *CC* in *bench*. In this section, we identify three parameters of synthetic corruption benchmarks that influence the way robustness to these benchmarks is correlated with robustness to natural corruptions. These parameters are : (1) the number of corruption categories represented (2) the balance among categories (3) the size of benchmarks. We make an ablation study in each of the three following sections to demonstrate the importance of each parameter.

4.3.4.1 Number of Corruption Categories Represented in Benchmarks

Each corruption category displayed in Figure 4.7 contains image transformations that modify different attributes in images. For instance, *Category 6* contains essentially corruptions that modify colorimetry; while *Category 4* contains corruptions that modify contrast and brightness. As a consequence, the features modified in one category, are mostly different from the ones modified in the other categories. Then, we make the assumption that the more corruption categories are represented in a benchmark, the more this benchmark takes into account a large diversity of attribute modifications.

Now, as shown in Figure 2.3, distribution shifts due to natural corruptions generally change a lot of attributes at the same time : background, resolution, viewpoint, etc. Then, intuitively, the largest the number of represented categories in a benchmark is, the more this benchmark is likely to make robustness estimations predictive of robustness to natural corruptions. To verify this intuition, we propose to use Algorithm 2 to build several benchmarks that have various numbers of represented categories.

Algorithm 2 Corruption Benchmark Generation Algorithm

Require: A group of candidate corruptions split into several corruption categories *cat*
Require: *n* : the number of categories represented in the returned benchmark
Require: *k* : the number of representatives of the categories represented in the returned benchmark
 Randomly select *n* categories in *cat*
 For each selected category, randomly select *k* distinct corruptions of this category
return A benchmark that contains the selected corruptions

Using the corruption categories displayed in Figure 4.7, we run Algorithm 2 for several *n, k* couples : (2,3), (3,2), (6,1), (4,3), (6,2), (6,3). We repeat this process until we obtain a group of 1000 different benchmarks for each of the considered *n, k* couples. We note that a benchmark generated using *n = 4, k = 3* contains 3 representatives of 4 out of 6 categories.

We want to measure whether increasing the number of represented categories *n* makes robustness estimations of benchmarks more correlated with robustness to natural corruptions. To verify this, we propose the Algorithm 3, that measures to what extent the robustness estimations made by a group of synthetic corruption benchmarks, are on average correlated with the robustness to one natural corruption benchmark.

For each of the benchmark groups generated using Algorithm 2, we run Algorithm 3 for each of the natural corruption benchmarks introduced in Section 4.3.2 (ImageNet-A, ImageNet-R, ImageNet-V2, ImageNet-Sketch and ObjectNet). The group of neural networks used to run Algorithm 3 is the set of models presented in Table 4.3. The obtained scores are displayed in Table 4.4 (*n, k* columns). We also compute the mean p-value associated with each of these scores : they are all lower than 0.02, which means that these correlations are statistically significant.

The higher the score of a group of synthetic corruption benchmarks of Table 4.4 is, the more the considered group makes on average robustness estimations correlated with the robustness to the

Algorithm 3 Estimates the average correlation between the robustness to one natural corruption benchmark and the robustness to synthetic corruption benchmarks

Require: SB a group of synthetic corruption benchmarks

Require: TNN a group of trained neural networks

Require: NB a benchmark of naturally corrupted samples

$scores_1 \leftarrow$ the residual robustness of all models in TNN computed with NB

For each benchmark sb in SB :

$scores_2 \leftarrow$ the residual robustness of all models in TNN computed with sb

 Get the Pearson correlation coefficient between $score_1$ and $score_2$

return the mean of the correlation coefficients computed in the loop

| Natural \ Synthetic | n, k | | | n, k | | | INet-C | INet-P | INet-S2N |
|---------------------|--------|-------|-------|--------|-------|--------------|--------|--------|--------------|
| | 2,3 | 3,2 | 6,1 | 4,3 | 6,2 | 6,3 | | | |
| ImageNet-A | 0.667 | 0.680 | 0.701 | 0.707 | 0.721 | 0.729 | 0.642 | 0.561 | 0.726 |
| ImageNet-R | 0.635 | 0.653 | 0.678 | 0.682 | 0.697 | 0.707 | 0.565 | 0.437 | 0.691 |
| ImageNet-V2 | 0.691 | 0.733 | 0.751 | 0.757 | 0.777 | 0.785 | 0.857 | 0.920 | 0.794 |
| ObjectNet | 0.695 | 0.732 | 0.757 | 0.763 | 0.789 | 0.796 | 0.807 | 0.823 | 0.814 |
| ImageNet-Sketch | 0.651 | 0.674 | 0.695 | 0.701 | 0.717 | 0.725 | 0.641 | 0.511 | 0.713 |
| Mean | 0.668 | 0.694 | 0.716 | 0.720 | 0.740 | 0.748 | 0.702 | 0.650 | 0.748 |

Table 4.4 – Correlation scores computed using Algorithm 3 between natural and synthetic corruption benchmarks. The n, k couples define the benchmarks generated using Algorithm 2.

natural corruption benchmark used to compute this score. To only study the effect of the number of represented categories n in benchmarks, we only compare the scores of Table 4.4 obtained using benchmarks that have the same size (the influence of the benchmark size parameter is studied in Section 4.3.4.3). So, we compare the benchmarks of 6 corruptions generated using the (n, k) couples (2, 3), (3, 2) and (6, 1). We see that the obtained scores increase with n for all the tested natural corruption benchmarks. Similarly, for the benchmarks of 12 corruptions generated using the (n, k) couples (4, 3) and (6, 2), the obtained scores are higher for $n = 6$ than for $n = 4$.

This experiment confirms the idea that increasing the number of categories represented in synthetic corruption benchmarks, makes robustness to these benchmarks more predictive of robustness to natural corruptions.

4.3.4.2 Balance Among Categories

We consider that the balance among categories represented in a benchmark is preserved, when all represented categories of this benchmark have the same number of representatives. For instance, the balance among categories of a benchmark *bench* that contains the Gaussian noise, iso-noise, multiplicative noise and color-jitter corruptions is not preserved : *bench* contains three representatives of *Category 5* and one representative of *Category 6* (see Figure 4.7). Obviously *bench* is biased towards texture damaging robustness rather than towards colorimetry variation robustness.

Intuitively, the robustness to a benchmark biased towards a few kinds of feature modifications, is not likely to be predictive of robustness to natural corruptions that change a large diversity of features in images. Consequently, preserving the balance among categories, should help to build benchmarks that make robustness estimations more correlated with robustness to natural corruptions.

We conduct an experiment to verify this intuition. We note *std*, the standard deviation compu-

ted using all the numbers of representatives of the categories represented in a benchmark. The *std* of benchmarks generated using Algorithm 2 are null : their balance among categories is preserved. We propose to get new benchmarks that have higher *std* by using the substitution operation.

A substitution randomly removes a corruption c_1 from a benchmark *bench* and adds to it a corruption c_2 randomly selected in the set of candidates. But, c_1 and c_2 are selected such as three conditions are respected : (1) the represented categories of *bench* do not change (2) *std* of *bench* strictly increases (3) c_2 is not already in *bench*.

We consider *group* : a set of 1000 benchmarks that have been generated using Algorithm 2 with $n = 6, k = 2$. We get 5000 new benchmarks, by substituting from 1 to 5 corruptions of each benchmark in *group*. The obtained benchmarks have various *std* : (0.6, 0.8, 1.0, 1.2, 1.4, 1.5, 1.8, 2.2). We group together all the benchmarks with the same *std*, the obtained groups contain all more than 200 benchmarks. For each of these groups, we run Algorithm 3 for each of the following natural corruption benchmarks : ImageNet-A, ImageNet-R, ImageNet-V2, ImageNet-Sketch and ObjectNet. The group of neural networks used to run Algorithm 3 is the set of models presented in Table 4.3. The obtained results are displayed in Table 4.5. We also compute the mean p-value associated with each score of this table, they are all lower than 0.02.

We observe that the scores of Table 4.5 decrease as the *std* of corruption benchmarks increases. This experiment confirms the intuition that benchmarks for which balance among represented categories is preserved, make robustness estimations more correlated with robustness to natural corruptions.

| Syn Bench Std \ Natural Bench | 0.0 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.5 | 1.8 | 2.2 |
|-------------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| ImageNet-A | 0.721 | 0.709 | 0.711 | 0.712 | 0.710 | 0.695 | 0.691 | 0.685 | 0.685 |
| ImageNet-R | 0.697 | 0.698 | 0.694 | 0.682 | 0.689 | 0.676 | 0.662 | 0.659 | 0.658 |
| ImageNet-V2 | 0.777 | 0.777 | 0.782 | 0.771 | 0.767 | 0.749 | 0.750 | 0.710 | 0.687 |
| ObjectNet | 0.789 | 0.798 | 0.781 | 0.786 | 0.777 | 0.761 | 0.744 | 0.713 | 0.698 |
| ImageNet-Sketch | 0.717 | 0.708 | 0.708 | 0.715 | 0.703 | 0.699 | 0.688 | 0.675 | 0.673 |
| Mean | 0.740 | 0.738 | 0.735 | 0.733 | 0.729 | 0.716 | 0.707 | 0.688 | 0.680 |

Table 4.5 – Correlations computed with Algorithm 3 using natural corruption benchmarks (lines) and groups of synthetic corruption benchmarks that have various *std* values (columns).

4.3.4.3 Corruption Benchmark Size

In Table 4.4, we observe that the scores obtained with the benchmarks generated using the n, k couples (6, 1), (6, 2) and (6, 3), increase with k . We notice that rising k for a fixed n when using Algorithm 2, is equivalent to increasing the size of generated benchmarks while conserving the balance among categories and the number of represented categories. Then, it appears that increasing the size of synthetic corruption benchmarks also helps to make robustness estimations more correlated with natural corruptions.

To explain the influence of this parameter, we note in Figure 4.7, that SCC are not completely correlated in terms of robustness. In other words, the representatives of the same category do not make exactly the same feature modifications. So, having more representatives in each category makes benchmarks measure the robustness to a larger range of feature modifications. Since natural corruptions modify a wide diversity of features in images, having more representatives per cate-

gory (increasing k for a fixed n) should make robustness to corruption benchmarks more predictive of robustness to natural corruptions, which can explain the obtained results.

4.3.5 Comparisons with Existing Synthetic Corruption Benchmarks

We want to set the three parameters identified in the previous sections to get the corruption selections that are the most correlated as possible in terms of robustness with natural corruptions. In others words, we want to build the largest benchmarks, that represent all the categories displayed in Figure 4.7 and have their balance among categories preserved. These benchmarks can be obtained by running the Algorithm 2, with $n = 6$ and k the largest as possible. In our case, the largest possible k is three because the smallest category displayed in Figure 4.7 contains only three corruptions.

Now, we want to determine if the benchmarks generated using $n = 6$, $k = 3$ are more predictive of robustness to natural corruptions than two existing synthetic corruption benchmarks : ImageNet-C and ImageNet-P [Hendrycks2019a]. In the same way as in the previous Section 4.3.4, we run Algorithm 3 using the models displayed in Table 4.3 with ImageNet-C, to estimate its correlation in terms of robustness with several natural corruption benchmarks. We repeat this process with ImageNet-P, but since this benchmark is not meant to be used with the residual robustness metric, we use its associated metric mean Flip Rate (mFR) instead to measure robustness towards this benchmark. More information about this metric can be found in Section 2.3.1. The obtained results are displayed in Table 4.4. The p-values associated with the scores computed using ImageNet-P and ImageNet-C are all lower than 0.05.

We observe that the benchmarks generated using $n = 6$, $k = 3$ are on average much more correlated in terms of robustness with ImageNet-A, ImageNet-Sketch and ImageNet-R than ImageNet-C and ImageNet-P. The contrary is observed for ImageNet-V2. For ObjectNet, the obtained scores are relatively close. The last line of Table 4.4 shows that the benchmarks generated using our methodology are on average more predictive of robustness to natural corruptions than ImageNet-C and ImageNet-P. It would be interesting to identify the reasons why the results obtained with ImageNet-V2 contrast with the general tendency.

For convenience, we provide an example of corruption selection picked from the benchmarks obtained using $n = 6$; $k = 3$ that we call ImageNet-Syn2Nat. It corresponds to the generated benchmark with the SCC that overlap the least which each other. The idea is to avoid getting a benchmark that contains corruptions that are almost equivalent. Obviously, other ways to pick a single benchmark could be used and further investigation would help to find the best ones. The correlations with natural corruption benchmarks of ImageNet-Syn2Nat are displayed in Table 4.4, and the names of its corruptions are underlined in Figure 4.7. We hope that ImageNet-Syn2Nat will help the community to make better robustness estimations when studying ImageNet classifiers.

4.3.6 Discussions

To our knowledge, these works constitute the first attempt to provide a methodology to build less biased synthetic corruption benchmarks. Despite its promising results, we think that this method would require further investigations to be fully mature.

Firstly, we think that the corruption category building process described in Section 4.3.3 can be improved. Namely, the overlapping scores of the *Category1* displayed in Figure 4.7, indicate that this category can be seen as a default category. Indeed, the corruptions it contains are quite isolated : they do not really overlap with any of the other candidate corruptions. Consequently, on contrary to the other categories, *Category1* does not really represent a precise aspect of neural network robustness. It is more like each of its corruptions could be considered as a distinct category.

This suggests improvements because the intuitions behind the high-level criteria identified in Section 4.3.4 are based on the idea that the used corruption categories contain overlapping

corruptions. *Category1* does not really meet this requirement. To overcome this issue, we could find ways to build more refined categories, in which would be split the *Category1* corruptions. This could probably be achieved by considering more candidate corruptions, to make it less likely to obtain isolated corruptions (that do not overlap with any of the other candidates).

Secondly, it could be also interesting to use a more complex clustering strategy to better capture the complexity of the overlappings between candidate corruptions. Rethinking our method using a hierarchical clustering strategy (such as HDBSCAN [Campello2013]) appears to be a good track to pursue in order to achieve this goal.

Finally, we note that the benchmark construction methodology presented in this Section 4.3 has been designed for image classification applications. Obviously, it could be adapted to other computer vision tasks. Especially, we think it would be valuable to adapt our method to image segmentation or 3D vision, where natural corruption benchmarks are particularly expensive to constitute. Synthetic corruption benchmarks that are predictive of robustness to natural corruptions would be extremely helpful in such applications.

4.3.7 Conclusion

In this Section 4.3, we demonstrated that some synthetic corruption selections are more predictive of robustness to natural corruptions than others. More precisely, in Section 4.3.3, we identified the key concept of corruption category that helps to better understand the links between synthetic and natural corruption robustnesses. These corruption categories are built such as the same category corruptions overlap while the others do not. In other words, the built categories reveal distinct aspects of neural network robustness. Namely, robustnesses to texture damagings, object shape distortions, color changes and lightning condition variations, appear to be largely independent attributes that should all be considered whether we address robustness to synthetic corruptions or robustness to natural ones.

Interestingly, we note that these attributes are indirectly considered in the most successful robustness interventions based on data augmentation on synthetic corruptions [Cubuk2019], [Hendrycks2020b], [Wang2021]. Indeed, they involve image transformations that can clearly be classified into the corruption categories we identified.

In Section 4.3.4, we leveraged the established corruption categories, by identifying three parameters that are important to consider when building corruption benchmarks : the number of categories represented, the balance among categories and the size. We showed that taking into account these parameters helps to build corruption benchmarks that make robustness estimations more correlated with robustness to natural corruptions. In particular, we built ImageNet-Syn2Nat in Section 4.3.5, a benchmark that is shown to be more predictive of natural corruption robustness than the two widely used synthetic benchmarks : ImageNet-C and ImageNet-P.

It is interesting to note that some previous results, suggest that natural corruption robustness and robustness to synthetic corruptions are not correlated [Taori2020], while some other studies indicate the opposite [Hendrycks2020a]. The experiments of this Section 4.3 show that the picture is more nuanced, and provide elements to better understand the circumstances under which these two aspects of neural network robustness are correlated.

Overall, our results might help to better estimate robustness using synthetic corruptions when natural corruption benchmarks are too expensive to constitute.

4.4 Conclusion

In this Chapter, we studied synthetic corruption benchmarks and we questioned the idea of arbitrarily selecting the corruptions to be included in them. Indeed, in Section 4.2, we showed that this bad practice can lead to benchmarks that are clearly biased towards some corruptions.

This is problematic considering that robustness to unforeseen synthetic corruptions is often seen as a way to better understand robustness to distribution shifts in general. Indeed, we demonstrated that robustness interventions designed to achieve robustness to biased synthetic benchmarks, are biased towards the same corruptions as these benchmarks. In particular, we identified a current tendency to provide excessive importance to texture damaging corruptions compared to other kinds of corruptions. In summary, insights gained by studying robustness to currently used synthetic corruption benchmarks are likely to be misleading, which reduces the chances of finding the way to design truly efficient robustness interventions.

As a consequence, we proposed in Section 4.3, a methodology to build less biased benchmarks. Basically, the aim of this strategy is to build benchmarks containing synthetic corruptions diverse enough to represent equally numerous aspects of neural network robustness.

We showed that our approach is particularly interesting when using synthetic corruptions to predict robustness to naturally arising distribution shifts. Indeed, we found that robustness to benchmarks built using our methodology transfers more to natural corruptions, than robustness to currently used synthetic benchmarks.

We hope that this new way of selecting corruptions to be included in benchmarks will help industrials to better estimate robustness of their neural networks, and then will reduce the chances of having unexpected performance reductions because of real-world distribution shifts. Concerning the academic community, we think that studying less biased benchmarks built using our guidelines, could help to get better insights on how to address neural network robustness to distribution shifts in general. In particular, we think it should help to build less specialized robustness interventions.

Overall, we think that synthetic corruptions can indeed form a convenient proxy to study robustness to other distribution shifts, as long as they are handled with care to avoid drawing misleading conclusions.

5

Conclusion

Contents

| | | |
|------------|--|-----------|
| 5.1 | Summary and Contributions | 71 |
| 5.2 | Further Works | 72 |
| 5.3 | Publications | 73 |

5.1 Summary and Contributions

The starting point of this thesis is the assumption regularly made in both academic and production contexts, that meeting some robustness specifications, provides a robustness guarantee on corruptions encountered after being deployed (this idea is illustrated in Figure 1.3). In other words, it is assumed that making sure that a model is robust to various corruptions that were not known during its conception, make it more likely to be robust to other corruptions it may face. In this thesis, we tried to better understand if and when this assumption holds.

We know that increasing a model robustness to one corruption c , also increases its robustness to a set of corruptions set . However, given c , we are not able to determine precisely set . Therefore, we do not know towards which corruptions a model is provided a robustness guarantee, given that it meets some robustness specifications. Roughly speaking, robustness specifications are generally arbitrarily established, crossing fingers that meeting them will make models robust to the unforeseen corruptions they will encounter.

We need to be able to determine more precisely, which corruptions overlap, i.e., are correlated in terms of robustness. In the example mentioned above, we want to know towards which corruptions a model is likely to be robust, given that it is robust to c . In other words, we need to clearly understand the circumstances under which neural network robustness to one distribution shift is likely to transfer to another. This should enable to better predict the situations where a model is reliable, depending on the robustness specifications it meets.

We think that the works of this thesis constitute a first important step towards this aim. Indeed, leveraging our results, we are now able to make very concrete assessments on neural network robustness, depending on their performances on some robustness benchmarks. For instance, we can now directly ensure that a model only guaranteed to be robust to adversarial examples, is not guaranteed to be robust to colorimetry changes nor unexpected image resolutions.

Indeed, we have shown in Section 3.2 that adversarial and synthetic corruption robustnesses are largely orthogonal. This means that robustness specifications based on adversarial examples only, are not relevant if deployed neural networks are likely to encounter synthetic corruptions. We have also shown that the reciprocal is true.

Another example of interesting prediction that can directly be made using our results, is that a model robust to Gaussian noise and shot noise, is very likely to be also robust to jpeg-compression artifacts. We can also assess that obstructions and brightness changes could still significantly harm the performances of this model.

Such predictions have been made possible thanks to the knowledge acquired in Sections 4.2 and 4.3, where we estimated the correlations in terms of robustness between some of the most currently used synthetic corruptions in the field of image classification. Moreover, we established great categories of synthetic corruptions that are all representative of an aspect of neural network robustness. Namely, we have shown that sensitivity to colorimetry changes, texture damagings, geometric transformations and lightning condition variations, are different faces of neural network robustness, which should all be considered when establishing robustness specifications.

Having these corruption categories in mind, is shown to be even more useful in Section 4.3, where we demonstrated that making neural networks robust to all these kinds of synthetic corruptions, make them more likely to be robust to naturally arising corruptions. This result in particular, should help to build more reliable models in production context, where natural corruptions form a major obstacle to neural network deployment.

Taken all together, we note that our works encourage to build more demanding robustness specifications. Namely, to maximize the chances of models to be robust to unforeseen corruptions, used benchmarks should contain representatives of all the kinds synthetic corruptions we have identified, in addition to adversarial attacks that should also be diverse and numerous.

Obviously, designing models robust to such diverse unforeseen corruptions is challenging. Unfortunately, we note that most of existing robustness interventions are very specialized, i.e.,

they provide robustness guarantees on very narrow ranges of corruptions, such as robustness to L_p bounded adversarial examples. Such robustness interventions are too limited regarding the challenging robustness benchmarks our results encourage to constitute.

Consequently, methods that improve robustness to wider ranges of corruptions are missing. That being the case, we presented in Section 3.3 M-TLAT, an example of defense able to increase robustness towards diverse and numerous kinds of corruptions. Obviously, M-TLAT is far from achieving robustness in a broad sense, but it shows that it is possible to provide new kinds of defenses that take into account more aspects of neural network robustness.

Overall, our works encourage new practices when measuring robustness in both academic and production contexts. We believe that this is essential to the establishment of new generation robustness interventions that achieve robustness in a broader sense; which are absolutely required to enable the deployment of neural networks in a lot of practical cases.

5.2 Further Works

While useful, the results of this thesis do not enable to completely understand the circumstances under which robustness to one kind of corruptions transfers to an other. Further investigations would be valuable.

Firstly, it is important to remind that all the experiments conducted this thesis only involve image classifiers. However, the issue tackled in our works, is important as well to other computer vision tasks. Obviously, we also need to establish robustness specifications when designing image segmentation models for instance. Consequently, we think that conducting similar studies in other computer vision fields would be valuable.

Secondly, considering that we studied correlations between adversarial and synthetic corruption robustnesses in Section 3.2, and we also established links between synthetic and natural corruption robustnesses in Section 4.3; investigations about correlations in terms of robustness between adversarial and natural corruption robustnesses would enable to close the loop.

Thirdly, we note that new kinds of adversarial attacks have emerged these recent years. In particular, we note the arrival of low-frequency adversarial examples [Guo2020], non-gradient based attacks [Brendel2018] and not-bounded pixel-wise attacks [Engstrom2019]. We think that it would be interesting to consider the robustness correlations between these attacks and other non-adversarial corruptions to refine the results found in Section 3.2.

In the same fashion, the range of synthetic corruptions considered in all our experiments could be extended. In particular, in Section 4.3, we have shown that natural corruption robustness estimations got by using synthetic corruptions, largely depend on the diversity of the used synthetic corruptions. Then, one could display the benefit of using synthetic corruptions in an even clearer way, by considering more of them when measuring robustness. Using additional image transformation libraries [Jung2020], or including synthetic corruptions based on generative models [Dunn2020], [Hendrycks2020a], appear to be relevant tracks towards this aim.

Most importantly, we note that the results of this thesis, are in line with existing works suggesting that neural network robustness has multiple faces [Carlini2019], [Hendrycks2020a]. We identified some of these faces by establishing synthetic corruption categories in Section 4.3. They are all representative of a different aspect of neural network robustness (robustness to texture damagings, to illuminations changes, etc.). An even subtler understanding of neural network robustness, could be acquired by defining more refined and numerous corruption categories. Therefore, we believe that this idea of identifying more and more faces of neural network robustness, is a relevant way to progressively get closer and closer to address robustness in a broad sense.

5.3 Publications

- Alfred Laugros, Alice Caplier, and Matthieu Ospici. Using Synthetic Corruptions to Measure Robustness to Natural Distribution Shifts. In *British Machine Vision Conference*. 2021.
- Alfred Laugros, Alice Caplier, and Matthieu Ospici. Using the Overlapping Score to Improve Corruption Benchmarks. In *International Conference on Image Processing*. 2021.
- Alfred Laugros, Alice Caplier, and Matthieu Ospici. Addressing Neural Network Robustness with Mixup and Targeted Labeling Adversarial Training. In *European Conference on Computer Vision RLQ Workshop*. 2020.
- Alfred Laugros, Alice Caplier, and Matthieu Ospici. Are Adversarial Robustness and Common Perturbation Robustness Independent Attributes? In *International Conference on Computer Vision RLQ Workshop*. 2019.

List of Figures

| | | |
|------|--|----|
| 1.1 | Illustration of the distribution shift phenomenon. Scenes arising in the real-world can be drastically different from the ones seen during trainings of neural networks. Upper images are extracted from [Cordts2016]. | 2 |
| 1.2 | Illustration of the data augmentation process. A duplicate of a model training set is corrupted with a corruption c (here a ninety degrees rotation). The considered neural network is then trained on both a clean training set and a corrupted one. Once trained, the model is robust to c | 3 |
| 1.3 | Diagram illustrating the standard procedure followed when trying to build robust neural networks. | 5 |
| 2.1 | Scheme of a traditional adversarial example crafting (extracted form [Goodfellow2015]). | 10 |
| 2.2 | Examples of synthetic corruptions. | 11 |
| 2.3 | Illustration of natural corruption effects : training images (extracted from the VIPeR dataset [Gray2007]) have features that are statistically different from features of naturally corrupted images. | 12 |
| 2.4 | Illustrations of complex corruptions generated using DeepAugment (images extracted from [Hendrycks2020a]). | 14 |
| 2.5 | Illustration of the self-supervised training paradigm. Here the complementary task to carry out in addition to the traditional classification task, consists in predicting rotation angles of images (scheme extracted from [Zhai2019]). | 15 |
| 2.6 | Illustration of the empirically measured linear correlation between accuracy on clean samples and accuracy on corrupted samples. Each point of this figure provides two accuracies of one model, respectively computed with clean samples and samples extracted from the natural corruption benchmarks ImageNet-V2 (left) and ObjectNet (right) (scheme extracted from [Taori2020]). | 17 |
| 2.7 | Sequences of similar images used to measure neural network prediction stability (images extracted from [Gu2019]). | 18 |
| 2.8 | Scheme of an adversarial example crafting using the FGSM algorithm. | 20 |
| 2.9 | Illustration of the fifteen ImageNet-C corruptions (Scheme extracted from [Hendrycks2019a]). The five severity levels associated with each of these corruptions are illustrated at the bottom of the Figure (using the salt-pepper noise corruption as an example). | 22 |
| 2.10 | Illustration of the two kinds of natural distribution shifts identified in [Ye2021]. The images on the left are extracted from [Rusak2021]. | 23 |
| 3.1 | Illustration of computed gradients for two models, of their loss L with respect to three input images x . Models robust to adversarial examples extract features that align better with human perception. (Scheme adapted from [Tsipras2019]). | 27 |
| 3.2 | Illustration of various image transformations used to estimate robustness to synthetic corruptions. | 30 |

| | | |
|-----|---|----|
| 3.3 | Illustrations of the low-frequency adversarial attacks presented in [Guo2020] (on the left) , of adversarial examples that are not bounded pixel-wise [Liu2019] (at the center) and GAN based attacks [Song2018b] (on the right). | 34 |
| 3.4 | Illustration of the mixup augmentation process. | 36 |
| 3.5 | Visualization of the generation of a new training couple using M-TLAT. | 38 |
| 3.6 | Illustration of the experiment revealing the existence of non-robust predictive features (scheme extracted from [Ilyas2019]). A model trained only on images mislabeled according to human perception (dog images labeled with "cat"), can still make accurate predictions on unseen data (it can recognize cats). In other words, the image on the upper right corner contains features that are never entangled in natural images (dog appearance with brittle invisible features of cats). | 43 |
| 3.7 | Illustration of the shape-texture debiased training process proposed in [Li2021b]. | 44 |
| 4.1 | Methodology used to compute the overlapping score between two corruptions c_1 and c_2 | 51 |
| 4.2 | Overlapping scores of ImageNet-C computed with the ResNet-18 (left) and DenseNet-121 (right) architectures. | 52 |
| 4.3 | Illustrations and implementation details of various image transformations not included in ImageNet-C. | 55 |
| 4.4 | Diagram describing how to compute the Minimal Samples Distance between two samples. | 56 |
| 4.5 | Images extracted from ImageNet-V2 [Recht2019], ImageNet-Sketch [Wang2019a], ImageNet-A [Hendrycks2021b], ObjectNet [Barbu2019] and ImageNet-R [Hendrycks2020a]. | 60 |
| 4.6 | Candidate corruptions displayed in the same order as the corruption names of Figure 4.7. | 61 |
| 4.7 | Overlapping scores computed using all the possible candidate corruption couples. The scores computed with SCC are in the 6 orange squares : one square for each of the 6 categories. | 62 |

List of Tables

| | | |
|-----|---|----|
| 3.1 | Robustness of models to adversarial examples, estimated by using the APP metric, in both the black-box (on the left) and white-box (on the right) settings. The column entitled "Clean", contains the accuracy of tested models on clean samples. | 31 |
| 3.2 | APP scores of adversarially trained models computed using the synthetic corruptions displayed in Figure 3.2. | 32 |
| 3.3 | APP scores of the standard and $no-\phi$ models computed with unforeseen corruptions. Each score of the second line refers to the robustness of the $no-\phi$ model towards the ϕ perturbation. | 32 |
| 3.4 | Effect on robustness of M-TLAT and comparison with IAT. | 40 |
| 3.5 | Influence on robustness of the Mixup and TLAT data augmentations. | 42 |
| 3.6 | Comparison between the accuracy on clean samples of the TLAT model and the one of models trained with other kinds of adversarial trainings. | 42 |
| 4.1 | The CE scores of various models computed with the <i>border</i> , <i>obstruction</i> , <i>set₁</i> and <i>set₂</i> corruptions. The scores in parentheses indicate the CE gains compared to the standard ResNet-50. | 54 |
| 4.2 | Albumentations functions and their associated parameters used to get the overlapping scores displayed Figure 4.7. | 59 |
| 4.3 | Selected models trained with a robustness intervention. Their plain counterpart are also included in this set of models used to compute robustness correlations between benchmarks. | 62 |
| 4.4 | Correlation scores computed using Algorithm 3 between natural and synthetic corruption benchmarks. The n, k couples define the benchmarks generated using Algorithm 2. | 64 |
| 4.5 | Correlations computed with Algorithm 3 using natural corruption benchmarks (lines) and groups of synthetic corruption benchmarks that have various <i>std</i> values (columns). | 65 |

Bibliography

- [Alexander2021] Alekhin Alexander et more than a thousand of contributors. *OpenCV*. <https://docs.opencv.org/master/>, 2021.
- [Amodei2016] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman et Dan Mané. *Concrete Problems in AI Safety*. CoRR, vol. abs/1606.06565, 2016.
- [Archambault2019] Guillaume P. Archambault, Yongyi Mao, Hongyu Guo et Richong Zhang. *MixUp as Directional Adversarial Training*. arXiv preprint arXiv :1906.06875, 2019.
- [Arjovsky2019] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani et David Lopez-Paz. *Invariant Risk Minimization*. arXiv preprint arXiv :1907.02893, 2019.
- [Athalye2018a] Anish Athalye, Nicholas Carlini et David Wagner. *Obfuscated Gradients Give a False Sense of Security : Circumventing Defenses to Adversarial Examples*. In Proceedings of the 35th International Conference on Machine Learning, pages 274–283. PMLR, 2018.
- [Athalye2018b] Anish Athalye, Logan Engstrom, Andrew Ilyas et Kevin Kwok. *Synthesizing Robust Adversarial Examples*. In Proceedings of the 35th International Conference on Machine Learning, 2018.
- [Azulay2019] Aharon Azulay et Yair Weiss. *Why do deep convolutional networks generalize so poorly to small image transformations?* Journal of Machine Learning Research, 2019.
- [Barbu2019] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum et Boris Katz. *ObjectNet : A large-scale bias-controlled dataset for pushing the limits of object recognition models*. In Advances in Neural Information Processing Systems, 2019.
- [Beery2018] Sara Beery, Grant Van Horn et Pietro Perona. *Recognition in Terra Incognita*. In Proceedings of the European Conference on Computer Vision (ECCV), September 2018.
- [Biggio2013] Battista Biggio, Iginio Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto et Fabio Roli. *Evasion Attacks against Machine Learning at Test Time*. In Machine Learning and Knowledge Discovery in Databases, 2013.
- [Brendel2018] Wieland Brendel, Jonas Rauber et Matthias Bethge. *Decision-Based Adversarial Attacks : Reliable Attacks Against Black-Box Machine Learning Models*. In International Conference on Learning Representations, 2018.
- [Buslaev2020] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin et Alexandr A. Kalinin. *Albumentations : Fast and Flexible Image Augmentations*. Information, 2020.

- [Campello2013] Ricardo JGB Campello, Davoud Moulavi et Jörg Sander. *Density-based clustering based on hierarchical density estimates*. In Pacific-Asia conference on knowledge discovery and data mining, pages 160–172. Springer, 2013.
- [Carlini2017] N. Carlini et D. Wagner. *Towards Evaluating the Robustness of Neural Networks*. In 2017 IEEE Symposium on Security and Privacy (SP), pages 39–57, May 2017.
- [Carlini2019] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry et Alexey Kurakin. *On Evaluating Adversarial Robustness*. arXiv preprint arXiv :1902.06705, 2019.
- [Che2020] Zhaohui Che, Ali Borji, Guangtao Zhai, Xiongkuo Min, Guodong Guo et Patrick Le Callet. *How is Gaze Influenced by Image Transformations? Dataset and Model*. Trans. Img. Proc., 2020.
- [Cordts2016] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth et Bernt Schiele. *The cityscapes dataset for semantic urban scene understanding*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3213–3223, 2016.
- [Cubuk2019] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan et Quoc V. Le. *AutoAugment : Learning Augmentation Strategies From Data*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.
- [Deng2009] J. Deng, W. Dong, R. Socher, L. Li, Kai Li et Li Fei-Fei. *ImageNet : A large-scale hierarchical image database*. In IEEE Conference on Computer Vision and Pattern Recognition, 2009.
- [Dhillon2018] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton et Animashree Anandkumar. *Stochastic activation pruning for robust adversarial defense*. In International Conference on Learning Representations, 2018.
- [Djolonga2021] Josip Djolonga, Jessica Yung, Michael Tschannen, Rob Romijnders, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Matthias Minderer, Alexander Nicholas D’Amour, Dan Moldovan, Sylvain Gelly, Neil Houlsby, Xiaohua Zhai et Mario Lucic. *On Robustness and Transferability of Convolutional Neural Networks*. In Conference on Computer Vision and Pattern Recognition, 2021.
- [Dodge2016] S. Dodge et L. Karam. *Understanding how image quality affects deep neural networks*. Eighth International Conference on Quality of Multimedia Experience (QoMEX), 2016.
- [Dodge2017a] Samuel F. Dodge et Lina J. Karam. *Quality Resilient Deep Neural Networks*. CoRR, vol. abs/1703.08119, 2017.
- [Dodge2017b] Samuel F. Dodge et Lina J. Karam. *Quality Resilient Deep Neural Networks*. CoRR, vol. abs/1703.08119, 2017.
- [Dong2018] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu et Jianguo Li. *Boosting Adversarial Attacks With Momentum*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

-
- [Dunn2020] Isaac Dunn, Laura Hanu, Hadrien Pouget, Daniel Kroening et Tom Melham. *Evaluating Robustness to Context-Sensitive Feature Perturbations of Different Granularities*. arXiv preprint arXiv :2001.11055, 2020.
- [Engstrom2019] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt et Aleksander Madry. *Exploring the Landscape of Spatial Robustness*. In Kamalika Chaudhuri et Ruslan Salakhutdinov, éditeurs, Proceedings of the 36th International Conference on Machine Learning, volume 97 of *Proceedings of Machine Learning Research*, pages 1802–1811. PMLR, 2019.
- [Engstrom2020] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Jacob Steinhardt et Aleksander Madry. *Identifying Statistical Bias in Dataset Replication*. In Proceedings of the 37th International Conference on Machine Learning, 2020.
- [Eykholt2018] Kevin Eykholt, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno et Dawn Song. *Robust Physical-World Attacks on Deep Learning Visual Classification*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.
- [Fawzi2016] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli et Pascal Frossard. *Robustness of classifiers : from adversarial to random noise*. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon et R. Garnett, éditeurs, Advances in Neural Information Processing Systems 29, pages 1632–1640. Curran Associates, Inc., 2016.
- [Franceschi2018] Jean-Yves Franceschi, Alhussein Fawzi et Omar Fawzi. *Robustness of classifiers to uniform L_p and Gaussian noise*. International Conference on Artificial Intelligence and Statistics, 2018.
- [Geirhos2018] Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schütt, Matthias Bethge et Felix A. Wichmann. *Generalisation in humans and deep neural networks*. In Advances in Neural Information Processing Systems, 2018.
- [Geirhos2019] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann et Wieland Brendel. *ImageNet-trained CNNs are biased towards texture ; increasing shape bias improves accuracy and robustness*. In International Conference on Learning Representations, 2019.
- [Geirhos2020] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge et Felix A. Wichmann. *Shortcut learning in deep neural networks*. Nature Machine Intelligence, 2020.
- [Ghosh2018] Sanjukta Ghosh, Rohan Shet, Peter Amon, Andreas Hutter et André Kaup. *Robustness of Deep Convolutional Neural Networks for Image Degradations*. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.
- [Goodfellow2015] Ian J. Goodfellow, Jonathon Shlens et Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. CoRR, vol. abs/1412.6572, 2015.
- [Goodfellow2016] Ian Goodfellow, Yoshua Bengio et Aaron Courville. Deep learning. MIT Press, 2016.
- [Gray2007] Douglas Gray, S. Brennan et H. Tao. *Evaluating Appearance Models for Recognition, Reacquisition, and Tracking*. In 10th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), 09/2007 2007.
-

- [Gu2019] Keren Gu, Brandon Yang, Jiquan Ngiam, Quoc V. Le et Jonathon Shlens. *Using Videos to Evaluate Image Model Robustness*. CoRR, vol. abs/1904.10076, 2019.
- [Gulrajani2021] Ishaan Gulrajani et David Lopez-Paz. *In Search of Lost Domain Generalization*. In International Conference on Learning Representations, 2021.
- [Gulshad2021] Sadaf Gulshad et Arnold W. M. Smeulders. *Natural Perturbed Training for General Robustness of Neural Network Classifiers*. CoRR, vol. abs/2103.11372, 2021.
- [Guo2018] Chuan Guo, Mayank Rana, Moustapha Cisse et Laurens van der Maaten. *Countering Adversarial Images using Input Transformations*. In International Conference on Learning Representations, 2018.
- [Guo2020] Chuan Guo, Jared S. Frank et Kilian Q. Weinberger. *Low Frequency Adversarial Perturbation*. In Ryan P. Adams et Vibhav Gogate, éditeurs, Proceedings of The 35th Uncertainty in Artificial Intelligence Conference, volume 115 of *Proceedings of Machine Learning Research*, pages 1127–1137. PMLR, 22–25 Jul 2020.
- [He2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. *Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification*. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), December 2015.
- [He2016] K. He, X. Zhang, S. Ren et J. Sun. *Deep Residual Learning for Image Recognition*. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, June 2016.
- [He2021] Yue He, Zheyang Shen et Peng Cui. *Towards Non-I.I.D. image classification : A dataset and baselines*. Pattern Recognition, vol. 110, page 107383, 2021.
- [Hendrycks2019a] Dan Hendrycks et Thomas Dietterich. *Benchmarking Neural Network Robustness to Common Corruptions and Perturbations*. Proceedings of the International Conference on Learning Representations, 2019.
- [Hendrycks2019b] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath et Dawn Song. *Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty*. Advances in Neural Information Processing Systems (NeurIPS), 2019.
- [Hendrycks2020a] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt et Justin Gilmer. *The Many Faces of Robustness : A Critical Analysis of Out-of-Distribution Generalization*. arXiv preprint arXiv :2006.16241, 2020.
- [Hendrycks2020b] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer et Balaji Lakshminarayanan. *AugMix : A Simple Method to Improve Robustness and Uncertainty under Data Shift*. In International Conference on Learning Representations, 2020.
- [Hendrycks2021a] Dan Hendrycks, Nicholas Carlini, John Schulman et Jacob Steinhardt. *Unsolved Problems in ML Safety*. arXiv preprint arXiv :2109.13916, 2021.
- [Hendrycks2021b] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt et Dawn Song. *Natural Adversarial Examples*. CVPR, 2021.
- [Heusel2017] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler et Sepp Hochreiter. *Gans trained by a two time-scale update rule converge*

-
- to a local nash equilibrium. *Advances in neural information processing systems*, vol. 30, 2017.
- [Huang2017] G. Huang, Z. Liu, L. Van Der Maaten et K. Q. Weinberger. *Densely Connected Convolutional Networks*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- [Huang2020] Zeyi Huang, Haohan Wang, Eric P. Xing et Dong Huang. *Self-Challenging Improves Cross-Domain Generalization*. In ECCV, 2020.
- [Huval2015] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, Fernando A. Mujica, Adam Coates et Andrew Y. Ng. *An Empirical Evaluation of Deep Learning on Highway Driving*. CoRR, 2015.
- [Ilyas2019] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran et Aleksander Madry. *Adversarial Examples Are Not Bugs, They Are Features*. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.
- [Inoue2018] Hiroshi Inoue. *Data Augmentation by Pairing Samples for Images Classification*. arXiv preprint arXiv :1801.02929, 2018.
- [Işın2016] Ali Işın, Cem Direkoğlu et Melike Şah. *Review of MRI-based Brain Tumor Image Segmentation Using Deep Learning Methods*. *Procedia Computer Science*, vol. 102, pages 317–324, 2016. 12th International Conference on Application of Fuzzy Systems and Soft Computing, ICAFS 2016, 29-30 August 2016, Vienna, Austria.
- [Jung2020] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte et al. *imgaug*. <https://github.com/aleju/imgaug>, 2020. Online ; accessed 01-Feb-2020.
- [Kamann2020] Christoph Kamann et Carsten Rother. *Benchmarking the Robustness of Semantic Segmentation Models*. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [Kannan2018] Harini Kannan, Alexey Kurakin et Ian Goodfellow. *Adversarial Logit Pairing*. arXiv preprint arXiv :1803.06373, 2018.
- [Karahan2016] S. Karahan, M. Kilinc Yildirim, K. Kirtac, F. S. Rende, G. Butun et H. K. Ekenel. *How Image Degradations Affect Deep CNN-Based Face Recognition ?* In 2016 International Conference of the Biometrics Special Interest Group (BIOSIG), pages 1–5, Sep. 2016.
- [Kingma2015] Diederik P. Kingma et Jimmy Ba. *Adam : A Method for Stochastic Optimization*. In Yoshua Bengio et Yann LeCun, editeurs, 3rd International Conference on Learning Representations, ICLR, 2015.
- [Kireev2021] Klim Kireev, Maksym Andriushchenko et Nicolas Flammarion. *On the effectiveness of adversarial training against common corruptions*. CoRR, vol. abs/2103.02325, 2021.
- [Koh2021] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanus Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness,
-

- Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn et Percy Liang. *WILDS : A Benchmark of in-the-Wild Distribution Shifts*. In International Conference on Machine Learning (ICML), 2021.
- [Kornblith2020] Simon Kornblith, Honglak Lee, Ting Chen et Mohammad Norouzi. *What's in a Loss Function for Image Classification?* CoRR, vol. abs/2010.16402, 2020.
- [Koziański2017] Michał Koziański et Bogusław Cyganek. *Image recognition with deep neural networks in presence of noise – Dealing with and taking advantage of distortions*. Integrated Computer-Aided Engineering, vol. 24, pages 1–13, 08 2017.
- [Krizhevsky2009] Alex Krizhevsky, Geoffrey Hinton et al. *Learning multiple layers of features from tiny images*. 2009.
- [Krizhevsky2012] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. In F. Pereira, C. J. C. Burges, L. Bottou et K. Q. Weinberger, éditeurs, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012.
- [Kurakin2017a] Alexey Kurakin, Ian J. Goodfellow et Samy Bengio. *Adversarial examples in the physical world*. In International Conference on Learning Representations, 2017.
- [Kurakin2017b] Alexey Kurakin, Ian J. Goodfellow et Samy Bengio. *Adversarial Machine Learning at Scale*. In International Conference on Learning Representations, 2017.
- [Lamb2019] Alex Lamb, Vikas Verma, Juho Kannala et Yoshua Bengio. *Interpolated Adversarial Training : Achieving Robust Neural Networks Without Sacrificing Too Much Accuracy*. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, page 95–103, 2019.
- [Li2017] Da Li, Yongxin Yang, Yi-Zhe Song et Timothy M. Hospedales. *Deeper, Broader and Artier Domain Generalization*. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017.
- [Li2021a] Junnan Li, Caiming Xiong et Steven Hoi. *MoPro : Webly Supervised Learning with Momentum Prototypes*. In International Conference on Learning Representations, 2021.
- [Li2021b] Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille et cihang xie. *Shape-Texture Debaised Neural Network Training*. In International Conference on Learning Representations, 2021.
- [Lim2019] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim et Sungwoong Kim. *Fast AutoAugment*. In Advances in Neural Information Processing Systems, 2019.
- [Liu2018] Xuanqing Liu, Minhao Cheng, Huan Zhang et Cho-Jui Hsieh. *Towards Robust Neural Networks via Random Self-ensemble*. In The European Conference on Computer Vision (ECCV), September 2018.
- [Liu2019] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai et Alec Jacobson. *Beyond Pixel Norm-Balls : Parametric Adversaries using an Analytically Differentiable Renderer*. In International Conference on Learning Representations, 2019.
- [Lu2017] Jiajun Lu, Hussein Sibai, Evan Fabry et David Alexander Forsyth. *NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles*. ArXiv, vol. abs/1707.03501, 2017.

-
- [Machado2020] Gabriel Resende Machado, Eugênio Silva et Ronaldo Ribeiro Goldschmidt. *Adversarial Machine Learning in Image Classification : A Survey Towards the Defender's Perspective*. CoRR, vol. abs/2009.03728, 2020.
- [Madry2018] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras et Adrian Vladu. *Towards Deep Learning Models Resistant to Adversarial Attacks*. In International Conference on Learning Representations, 2018.
- [Mahajan2018] Dhruv Kumar Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe et Laurens van der Maaten. *Exploring the Limits of Weakly Supervised Pretraining*. In ECCV, 2018.
- [Marcel2010] Sébastien Marcel et Yann Rodriguez. *Torchvision the Machine-Vision Package of Torch*. In Proceedings of the 18th ACM International Conference on Multimedia, MM '10, page 1485–1488. Association for Computing Machinery, 2010.
- [Metz2019] Luke Metz, Niru Maheswaranathan, Jonathon Shlens, Jascha Sohl-Dickstein et Ekin D. Cubuk. *Using learned optimizers to make models robust to input noise*. CoRR, vol. abs/1906.03367, 2019.
- [Michaelis2019] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander Ecker, Matthias Bethge et Wieland Brendel. *Benchmarking Robustness in Object Detection : Autonomous Driving when Winter is Coming*. arXiv preprint arXiv :1907.07484, 2019.
- [Mintun2021] Eric Mintun, Alexander Kirillov et Saining Xie. *On Interaction Between Augmentations and Corruptions in Natural Corruption Robustness*. CoRR, vol. abs/2102.11273, 2021.
- [Na2018] Taesik Na, Jong Hwan Ko et Saibal Mukhopadhyay. *Cascade Adversarial Machine Learning Regularized with a Unified Embedding*. In International Conference on Learning Representations, 2018.
- [Orhan2019] A. Emin Orhan. *Robustness properties of Facebook's ResNeXt WSL models*. CoRR, vol. abs/1907.07640, 2019.
- [Pang2020] Tianyu Pang, Kun Xu et Jun Zhu. *Mixup Inference : Better Exploiting Mixup to Defend Adversarial Attacks*. In International Conference on Learning Representations, 2020.
- [Papernot2016] N. Papernot, P. McDaniel, X. Wu, S. Jha et A. Swami. *Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks*. In 2016 IEEE Symposium on Security and Privacy (SP), 2016.
- [Papernot2017] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik et Ananthram Swami. *Practical Black-Box Attacks against Machine Learning*. In ACM on Asia Conference on Computer and Communications Security, 2017.
- [Recht2019] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt et Vaishaal Shankar. *Do ImageNet Classifiers Generalize to ImageNet ?* In Proceedings of the 36th International Conference on Machine Learning, 2019.
- [Ren2015] Shaoqing Ren, Kaiming He, Ross Girshick et Jian Sun. *Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, 06 2015.
-

- [RichardWebster2019] Brandon RichardWebster, Samuel E. Anthony et Walter J. Scheirer. *Psy-Phy : A Psychophysics Driven Evaluation Framework for Visual Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019.
- [Ross2018] Andrew Ross et Finale Doshi-Velez. *Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing Their Input Gradients*. In AAAI Conference on Artificial Intelligence, 2018.
- [Rusak2020] Evgenia Rusak, Lukas Schott, Roland S. Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge et Wieland Brendel. *A simple way to make neural networks robust against diverse image corruptions*. ECCV, 2020.
- [Rusak2021] Evgenia. Rusak, Steffen Schneider, Peter Gehler, Oliver Bringmann, Wieland Brendel et Matthias Bethge. *Adapting ImageNet-scale models to complex distribution shifts with self-learning*. CoRR, vol. abs/2104.12928, 2021.
- [Sagawa2020] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto et Percy Liang. *Distributionally Robust Neural Networks*. In International Conference on Learning Representations, 2020.
- [Samangouei2018] Pouya Samangouei, Maya Kabkab et Rama Chellappa. *Defense-GAN : Protecting Classifiers Against Adversarial Attacks Using Generative Models*. In International Conference on Learning Representations, 2018.
- [Santhanam2018] Gokula Krishnan Santhanam et Paulina Grnarova. *Defending Against Adversarial Attacks by Leveraging an Entire GAN*. arXiv preprint arXiv :1805.10652, 2018.
- [Schmidt2018] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar et Aleksander Madry. *Adversarially Robust Generalization Requires More Data*. In Advances in Neural Information Processing Systems 31, pages 5014–5026. Curran Associates, Inc., 2018.
- [Shafahi2019] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor et Tom Goldstein. *Adversarial training for free!* In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox et R. Garnett, éditeurs, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.
- [Shankar2019] Vaishaal Shankar, Achal Dave, Rebecca Roelofs, Deva Ramanan, Benjamin Recht et Ludwig Schmidt. *A Systematic Framework for Natural Perturbations from Videos*. In ICML Workshop on Identifying and Understanding Deep Learning Phenomena, 2019.
- [Simonyan2015] K. Simonyan et A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. In International Conference on Learning Representations, 2015.
- [Song2018a] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon et Nate Kushman. *PixelDefend : Leveraging Generative Models to Understand and Defend against Adversarial Examples*. In International Conference on Learning Representations, 2018.
- [Song2018b] Yang Song, Rui Shu, Nate Kushman et Stefano Ermon. *Constructing Unrestricted Adversarial Examples with Generative Models*. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi et R. Garnett,

-
- editeurs, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [Su2018] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen et Yupeng Gao. *Is Robustness the Cost of Accuracy? – A Comprehensive Study on the Robustness of 18 Deep Image Classification Models*. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [Szegedy2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow et Rob Fergus. *Intriguing properties of neural networks*. In *International Conference on Learning Representations*, 2014.
- [Szegedy2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens et Zbigniew Wojna. *Rethinking the Inception Architecture for Computer Vision*. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [Tadros2019] Timothy Tadros, Nicholas C. Cullen, Michelle R. Greene et Emily A. Cooper. *Assessing Neural Network Scene Classification from Degraded Images*. *ACM Trans. Appl. Percept.*, 2019.
- [Taigman2014] Y. Taigman, M. Yang, M. Ranzato et L. Wolf. *DeepFace : Closing the Gap to Human-Level Performance in Face Verification*. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, June 2014.
- [Talby2020] David Talby. *Three Insights From Google’s Failed Test To Use AI For Medical Diagnosis*. <https://www.forbes.com/sites/forbestechcouncil/2020/06/09/three-insights-from-googles-failed-field-test-to-use-ai-for-medical-dia-2020>.
- [Tan2019] Mingxing Tan et Quoc Le. *EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks*. In *Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research*. PMLR, 09–15 Jun 2019.
- [Taori2020] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht et Ludwig Schmidt. *Measuring Robustness to Natural Distribution Shifts in Image Classification*. In *NeurIPS*, 2020.
- [Temel2017] Dogancan Temel, Gukyeong Kwon, Mohit Prabhushankar et Ghassan Al-Regib. *Cure-tsrl : Challenging unreal and real environments for traffic sign recognition*. *NIPS Workshop*, 2017.
- [Tokozume2018] Y. Tokozume, Y. Ushiku et T. Harada. *Between-Class Learning for Image Classification*. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5486–5494, June 2018.
- [Tramèr2018] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh et Patrick McDaniel. *Ensemble Adversarial Training : Attacks and Defenses*. In *International Conference on Learning Representations*, 2018.
- [Tsipras2019] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner et Aleksander Madry. *Robustness May Be at Odds with Accuracy*. In *International Conference on Learning Representations*, 2019.
- [Vasconcelos2020] Cristina Vasconcelos, Hugo Larochelle, Vincent Dumoulin, Nicolas Le Roux et Ross Goroshin. *An Effective Anti-Aliasing Approach for Residual Networks*. *ArXiv*, abs/2011.10675, 2020.
-

- [Vasiljevic2016] Igor Vasiljevic, Ayan Chakrabarti et Gregory Shakhnarovich. *Examining the Impact of Blur on Recognition by Convolutional Networks*. arXiv preprint arXiv :1611.05760, 2016.
- [Verma2019] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz et Yoshua Bengio. *Manifold Mixup : Better Representations by Interpolating Hidden States*. In Proceedings of the 36th International Conference on Machine Learning, pages 6438–6447. PMLR, 2019.
- [Wang2019a] Haohan Wang, Songwei Ge, Zachary Lipton et Eric P Xing. *Learning Robust Global Representations by Penalizing Local Predictive Power*. In Advances in Neural Information Processing Systems, pages 10506–10518, 2019.
- [Wang2019b] Jianyu Wang et Haichao Zhang. *Bilateral Adversarial Training : Towards Fast Training of More Robust Models Against Adversarial Attacks*. In The IEEE International Conference on Computer Vision (ICCV), October 2019.
- [Wang2021] Haotao Wang, Chaowei Xiao, Jean Kossaifi, Zhiding Yu, Anima Anandkumar et Zhangyang Wang. *AugMax : Adversarial Composition of Random Augmentations for Robust Training*. arXiv preprint arXiv :2110.13771, 2021.
- [Wong2020] Eric Wong, Leslie Rice et J. Zico Kolter. *Fast is better than free : Revisiting adversarial training*. In International Conference on Learning Representations, 2020.
- [Xiao2018] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu et Dawn Song. *Spatially Transformed Adversarial Examples*. In International Conference on Learning Representations, 2018.
- [Xie2017] S. Xie, R. Girshick, P. Dollár, Z. Tu et K. He. *Aggregated Residual Transformations for Deep Neural Networks*. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- [Xie2020a] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille et Quoc V. Le. *Adversarial Examples Improve Image Recognition*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.
- [Xie2020b] Qizhe Xie, Minh-Thang Luong, Eduard Hovy et Quoc V. Le. *Self-Training With Noisy Student Improves ImageNet Classification*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.
- [Xu2018] Weilin Xu, David Evans et Yanjun Qi. *Feature Squeezing : Detecting Adversarial Examples in Deep Neural Networks*. In 25th Annual Network and Distributed System Security Symposium, 2018.
- [Yalniz2019] I. Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri et Dhruv Mahajan. *Billion-scale semi-supervised learning for image classification*. arXiv preprint arXiv :1905.00546, 2019.
- [Ye2021] Nanyang Ye, Kaican Li, Lanqing Hong, Haoyue Bai, Yiting Chen, Fengwei Zhou et Zhenguo Li. *OoD-Bench : Benchmarking and Understanding Out-of-Distribution Generalization Datasets and Algorithms*. CoRR, vol. abs/2106.03721, 2021.

- [Yin2019] Dong Yin, Raphael Gontijo Lopes, Jonathon Shlens, Ekin D. Cubuk et Justin Gilmer. *A Fourier Perspective on Model Robustness in Computer Vision*. ICML Workshop on Uncertainty and Robustness in Deep Learning, 2019.
- [Yun2019] Sangdoon Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe et Youngjoon Yoo. *CutMix : Regularization Strategy to Train Strong Classifiers With Localizable Features*. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.
- [Zech2018] John R. Zech, Marcus A. Badgeley, Manway Liu, Anthony B. Costa, Joseph J. Titano et Eric Karl Oermann. *Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs : A cross-sectional study*. PLOS Medicine, pages 1–17, 11 2018.
- [Zhai2019] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov et Lucas Beyer. *S4L : Self-Supervised Semi-Supervised Learning*. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.
- [Zhang2018] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin et David Lopez-Paz. *mixup : Beyond Empirical Risk Minimization*. In International Conference on Learning Representations, 2018.
- [Zhang2019] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui et Michael Jordan. *Theoretically Principled Trade-off between Robustness and Accuracy*. In Proceedings of the 36th International Conference on Machine Learning, pages 7472–7482, 2019.
- [Zhou2017] Y. Zhou, S. Song et N. Cheung. *On classification of distorted images with deep convolutional neural networks*. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1213–1217, March 2017.

