ADVERSARIAL ROBUSTNESS IN DATA AUGMENTATION

Hamid Eghbal-zadeh[1,2], **Khaled Koutini**[1], **Verena Haunschmid**[1],
Paul Primus[1], **Michal Lewandowski**[3], **Werner Zellinger**[3], **Gerhard Widmer**[1,2]
[1] LIT Artificial Intelligence Lab, Johannes Kepler University, Linz, Austria
[2] Institute of Computational Perception, Johannes Kepler University, Linz, Austria
[3] Software Competence Center Hagenberg GmbH, Hagenberg, Austria
hamid.eghbal-zadeh@jku.at

ABSTRACT

Data augmentation has become a standard technique in deep learning, as it has been shown to greatly improve the generalisation abilities of models. In addition to human-designed augmentation operations such as geometric transformations (e.g., on images), recently some methods were proposed that generate new samples from the training data (e.g. using Mixup or GANs). In this paper, we empirically assess the effect of these kinds of data augmentation, regarding both classification accuracy and adversarial vulnerability. We find that 'classical' augmentation improves performance and robustness the most. However, we also find that while GAN-based augmentation and Mixup can improve prediction, they cause significant adversarial vulnerabilities when applied alone. Analyzing the smoothness of the models' decision boundaries, we can relate smoothness to robustness, and find that classical augmentation results in smoother boundaries than Mixup and GAN augmentation. Finally, using influence functions we show that, when asked to predict on adversarial test examples, vulnerable models rely more on augmented samples than on real ones. Taken together, our results suggest that general-purpose data augmentations that do not take into the account the characteristics of the data and the task, must be applied with care.

1 INTRODUCTION

Data augmentation is one of the fundamental building blocks of deep learning, and it has been shown that without it, deep neural networks suffer from different problems such as lack of generalisation (Perez & Wang, 2017) and adversarial vulnerability (Zhang et al., 2017). By affecting a model's behavior outside of the given training data, any data augmentation strategy introduces a certain *bias* (Battaglia et al., 2018). Some augmentation methods incorporate inductive bias into the model by transformations designed by domain experts, while others rely on sampling from a proxy distribution (i.e., vicinity distribution) or a learned distribution (i.e., generative models).

In this work, we empirically analyze three classes of data augmentation in the visual domain: 'classical' augmentation via geometric transformations (which can be viewed as expert knowledge), Mixup (Zhang et al., 2017), and GAN augmentation, looking at both classification performance and adversarial robustness. We propose a theoretical formulation of data augmentation in a probabilistic setup, which permits us to express combination of different data augmentations as a composition of functions (a general concept that we believe will be useful beyond the current paper). Extensive experiments with data augmentation methods used independently and in combination, and with systematic adversarial attacks, reveal that only the expert-defined biases improve the model performance if applied alone. More importantly, we observe that models with expert-defined augmentations are much more adversarially robust, compared to the other models. In contrast to the common belief that Mixup and GAN augmentation improve both generalization and robustness, our experiments suggest that this happens only when these methods are applied in combination with classical data augmentation. To reveal the underlying reasons behind, we propose a specific definition of the *smoothness* of decision boundaries and show that this property is systematically related to the adversarial vulnerability of models. While we find that Mixup and GAN augmentation causes models

to have smooth boundaries around the training data, we observe less smoothness around the unseen test examples in such models, compared to models trained with classical data augmentation. Also, by calculating *influence functions* (Koh & Liang, 2017) on the trained models, we demonstrate that models trained with GAN augmentation rely more on GAN-augmented samples when predicting on adversarial examples, than when predicting on real examples. All in all, these results suggest that general-purpose data augmentation based on vicinity distributions of training data or generative models must be used with care, and requires deeper analysis.

2 Formalization of Data Augmentation

In the Appendix A, we formulate data augmentation in a probabilistic setup, as a random function that produces a new observation from a given sample $\mathbf{x}$ in such a way that the most probable class label remains unchanged (see Definition 2); this random function can be sampled from, to obtain an arbitrary number of augmented vectors $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_s$. Some examples of how to construct simple standard data augmentation functions under this definition are given in A. More complex augmentation strategies can be constructed by function composition (Lemma 1 in Appendix A). Examples, including how to formulate augmentation via Conditional GAN or Mixup, are also provided in the Appendix A. The specific augmentation strategies that will be experimentally evaluated here are precisely the three functions defined as *Examples 4, 5, and 6* in A: a composition of horizontal flip, random rotation and noise addition (what will be called 'classical' augmentation in the following); a conditional GAN; and Mixup.

3 Empirical Analysis of Data Augmentation

Our experimental test domain is image classification, using CIFAR10 (Krizhevsky et al., 2009). The empirical analysis of the augmentation functions defined above is structured into three parts: (1) *performance analysis*, where we look at the effect of data augmentation on classification performance and adversarial robustness; (2) *boundary analysis*, where we analyse how the decision boundary of a model is affected by the augmentation; and (3) *influence analysis*, where we look at how much a model relies on augmented training samples when predicting on the real test examples and their adversarial counterparts.

The following subsections summarise the main points and observations. The precise details of experimental setup and quantitative results are described in Appendix B.

3.1 Performance analysis

3.1.1 Experimental Setup

Two criteria for performance evaluation are considered: 1) classification performance, in terms of classification accuracy on a test set; and 2) adversarial robustness. To assess the latter, we carry out *Projected Gradient Descent (PGD)* attacks (Kurakin et al., 2016; Madry et al., 2017) on each trained model. Having full access to the model, we create adversarial examples for each test example, and compute the accuracy for the adversarial test set. These experiments are detailed in Appendix B.1.

We compare two augmentation scenarios: in the first, each data augmentation function is investigated independently from the others (see Section B.1.1). In the second (Section B.1.2), combinations of data augmentations are studied. For a more detailed explanation of our performance analysis, see Appendix B.1.

3.1.2 Summary of Results

(1) Independent Data Augmentation Functions: As can be seen in Figure 1, both GAN and Mixup data augmentation improve test accuracy compared to non-augmented classifiers (0% augmented), while being outperformed by classical data augmentation. Hence, all data augmentations are *useful*. However, comparing GAN and Mixup with non-augmented classifiers, we observe that the adversarial vulnerability increases with the amount of augmentation in the majority of the cases. This is in contrast to models augmented with classical augmentation, which increases the robustness.

This observation suggests that the inductive bias introduced by well-understood image transformations is more helpful, in terms of classification performance and adversarial robustness, than other augmentation methods.

It can be additionally observed that while the classification performance in both Mixup and GAN data augmentation does not change significantly when varying the amount of augmentation p (except for the extreme case of $p = 1$ in GANs), adversarial vulnerability increases significantly with p in all the attacks on the classifiers that used Mixup and GAN data augmentation. This difference is more apparent in the weaker attacks (e.g., PGD 10 iterations), which suggests that using Mixup and GAN augmentation alone increases adversarial vulnerability. With stronger attacks (PGD with 100 iterations), the difference between GAN and classical augmentation shrinks, while Mixup remains the most vulnerable method. Therefore, GAN and Mixup can be considered *non-robust* augmentations, while classical augmentation can be considered a *robust* data augmentation.

(2) Composition of Data Augmentation Functions: We observe a different pattern in the function composition experiments. As can be seen in Figure 4 in the Appendix, the classification performance of Mixup and GAN data augmentation models reaches the performance of the classical data augmentation. Mixup seems to achieve better classification performances in all cases, when combined with classical data augmentation. While the performance of GAN-augmented classifiers has been significantly improved by the composition with classic data augmentation, in the extreme case of $p = 1$ the performance degradation is apparent.

Similarly to the independent experiments, classical augmentation remains the most robust classifier in the majority of the cases. Looking at the adversarial accuracies, we can see two different patterns for the Mixup and the GAN-augmented classifiers. The Mixup classifiers show more robustness to weaker attacks (PGD with 10 iterations), compared to GAN-augmented classifiers. However, in stronger attacks (PGD with 100 iterations), GAN and Mixup augmentation achieve similar results. Overall, in the majority of cases classical augmentation achieves better robustness compared to GAN and Mixup augmentation.

3.2 Boundary analysis

3.2.1 Experimental Setup

To obtain a measurable property of robustness for models that we can relate to (and that may partly explain) the observed adversarial vulnerability, we propose the following measure of *smoothness* for functions.

Definition 1 (Smoothness)**.** Let X be a random variable on $\mathcal{X}$. We define the *ϵ-smoothness* of some function $f : \mathcal{X} \to \mathcal{Y}$ as:

$$\text{Smoothness}(f) := P\left(f(X) = f(\mathbf{x}) \mid \mathbf{x} \in \partial B_\epsilon(X)\right), \tag{1}$$

where $\partial B_\epsilon(X)$ is the surface of a ball $B_\epsilon(X)$ around X with radius $\epsilon > 0$. In other words, for a given input $\mathbf{x}$ and its predicted label $f(\mathbf{x})$, smoothness relates to the probability that a random neighbor from the ϵ-sphere of $\mathbf{x}$ will be assigned the same label by the model.

We now define the *smoothness curve* as a curve that represents *the average of the smoothness values per percentile*, for a given set of samples. The higher the percentile values, the more robust the model.

Our smoothness measure can be related to adversarial robustness, both intuitively and experimentally: if the boundary around a sample is smooth, then it is less likely that the prediction of the model can be flipped by an example in that neighborhood. We empirically show that if a model has smoother decision boundaries around unseen test examples, this model is more adversarially robust. Based on these observations, if the majority of samples have smooth boundaries around them, then the model is more robust. Hence, adversarial robustness increases with smoothness.

In practice, we sample a number of points from the surface of the ϵ-sphere around every $\mathbf{x}_i$, and calculate the ratio of sampled points that have the same class prediction as the label $f(\mathbf{x}_i)$, where f is the classifier model we analyse. This is our estimate of the *smoothness* around $\mathbf{x}_i$. We then compare the distribution of smoothness measures for different models in Figure 2. The details of the experimental setup are provided in Appendix D.

3.2.2 SUMMARY OF EXPERIMENTAL RESULTS

Figure 2 compares the smoothness of different classifiers with different augmentation methods and augmentation probabilities on the train and the test sets, as explained in Sections 3.2 and D.6. We observe that, overall, the higher the augmentation rate is, the less smooth the classifier will become around the original training samples. However, different augmentation methods result in different amount of smoothness around *test* examples (see Figure 7). For example, Mixup augmentation, which encourages a linear relationship between the training samples and the labels, results in the smoothest boundaries around training samples, while having the least smooth boundaries around the test examples. This observation is in line with the adversarial vulnerability of Mixup, which is the most vulnerable among all methods. GAN augmentations have smoother boundaries around test examples compared to Mixup, but less smooth compared to classical, which is also in agreement with their adversarial vulnerability as reported in Figure 1. Please see Appendix C for an extended results.

3.3 INFLUENCE ANALYSIS

3.3.1 EXPERIMENTAL SETUP

We use *influence functions* (Koh & Liang, 2017) to analyse the importance of normal and GAN-augmented training data in relation to the adversarial vulnerability of the resulting classifiers. Our influence analysis is explained in detail in Appendix B.3. We measure the influence values for GAN-augmented training examples, as well as for normal training examples, for two sets of test examples: the first set is the original test set, and the second set is their adversarial counterparts. Hence the influence of GAN samples over the original test set, and over the adversarial test set can now be compared using these influence values.

3.3.2 SUMMARY OF EXPERIMENTAL RESULTS

The results provided in Figure 3 compare classifiers that were trained with different amounts of GAN augmentation. We look at the performance of GAN-augmented models using their influence values on normal and GAN-augmented training data. As can be seen, all influence values for adversarial examples (dotted lines) are higher than the influence values of real test examples (solid lines). This means that the decisions of a model on adversarial examples have been influenced by GAN-augmented examples more than the decisions on normal test examples. This shows that when a model that used GAN augmentation wants to make a (wrong) decision on an adversarial example, it relies on GAN-augmented training examples. However, when the same model wants to predict on a real test example, then the GAN-augmented examples are not as important for this decision compared to the previous case. We can additionally observe that this influence of GAN examples increases with the amount of augmentation used in the training of the models. As the models with a higher amount of GAN augmentation are more vulnerable, we can relate adversarial vulnerability to the influence of the *non-robust augmented data*.

4 CONCLUSION

In this paper, we defined data augmentation as random functions and provided the ground for further theoretical analysis. We empirically studied the relation between adversarial robustness and data augmentation, as well as the relation between model performance and data augmentation. We proposed a new measure to analyse the decision boundaries of deep neural networks, and showed that there is a strong connection between adversarial robustness and the smoothness of boundaries around unseen examples. Finally, using influence functions we showed how GAN-augmented models rely more on augmented data when predicting on adversarial examples. In our future work, we will investigate *robust data augmentation* methods that preserve adversarial robustness in models, while being a *useful data augmentation*, in which help to improve classification accuracy.

ACKNOWLEDGMENTS

We thank Johannes Brandstetter, Philip Winter, and Lukas Gruber from the Machine Learning Institute at the Johannes Kepler University of Linz for their feedback on this paper. We also gratefully acknowledge the support of the NVIDIA Corporation with the donation of a Titan X GPU used for this research. This work has been supported by the LCM – K2 Center within the framework of the Austrian COMET-K2 program, and has been partly funded by BMK, BMDW, and the Province of Upper Austria in the frame of the COMET Programme managed by FFG and the COMET Module S3AI.

REFERENCES

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Rüdiger Busche. Incense (python library), 2019. URL `https://github.com/JarnoRFB/incense`.

R. Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, 42(1): 65–68, 1977. doi: 10.1080/00401706.2000.10485981. URL `https://doi.org/10.1080/00401706.2000.10485981`.

R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL `https://github.com/MadryLab/robustness`.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Klaus Greff, Aaron Klein, Martin Chovanec, Frank Hutter, and Jürgen Schmidhuber. The sacred infrastructure for computational research. In *Proceedings of the Python in Science Conferences-SciPy Conferences*, 2017.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pp. 6626–6637, 2017.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 1885–1894, 2017. URL `http://proceedings.mlr.press/v70/koh17a.html`.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Appendix

Table of Contents

A Formalization of Data Augmentation (extended)

In the following we formulate data augmentation in a probabilistic setup. Let $(\mathcal{X}, \mathcal{F}_1, P_1)$ be a probability space with a state space $\mathcal{X} \subseteq \mathbb{R}^n$ of inputs, e.g. images, and let $(\mathcal{Y}, \mathcal{F}_2, P_2)$ be a probability space with a state space $\mathcal{Y} \subset \mathbb{N}$ of labels, e.g. classes $1, \dots, l$. Following Vapnik (2013), an s-sized sample is a sequence

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_s, y_s)$$

of input-label pairs which are observations of independent and identically distributed random variables on the product space $(\mathcal{X} \times \mathcal{Y}, \mathcal{F}_1 \otimes \mathcal{F}_2, P := P_1 \times P_2)$.

Definition 2 (Augmentation)**.** Let X and Y be random variables on $\mathcal{X}$ and $\mathcal{Y}$, respectively, and R be a real-valued Borel random vector. We call a random function A_R an *augmentation* if

$$\arg\max_{y \in \mathcal{Y}} P(Y = y|X) = \arg\max_{y \in \mathcal{Y}} P(Y = y|A_R \circ X). \tag{2}$$

For some random variable X, the term $P(\cdot|X) := P(\cdot|\sigma(X))$ denotes the random variable obtained by conditioning on the σ-algebra $\sigma(X)$ generated by X.

By Definition 2, an augmentation A_R is a random function taking some input $\mathbf{x}$ and returning some random variable $A_R(\mathbf{x})$ realizing values in $\mathcal{X}$. Under this framework, various augmented vectors $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_s \in \mathcal{X}$ can be obtained from a single input $\mathbf{x} \in \mathcal{X}$ by observing $A_R(\mathbf{x})$, i.e. by sampling from the corresponding distribution. Some examples on how to construct simple data augmentation functions given the definition above are provided in the next section.

Examples: Let $\mathcal{X} \subset \mathbb{R}^n$ be a set of images of cats and dogs, and $\mathcal{Y} = \{0, 1\}$ a corresponding set of labels.

1. Adding a small fraction of noise to an image of some cat does not make it classified as a dog. This noise adding can be formalized as augmentation
$$A_N(\mathbf{x}) = \mathbf{x} + N \tag{3}$$
for some random variable $N : \Omega \to [0, \epsilon]^n$ with small $\epsilon > 0$.
2. Cropping an image $\mathbf{x} \in \mathcal{X}$ can be modeled by zeroing a random number of the "outer" pixels, where we assume the boarder of the image is encoded in the first and last elements of the n-dimensional vector e.g.
$$A_U(\mathbf{x}) = (0, \ldots, 0, x_U, \ldots, x_{n-U}, 0, \ldots, 0), \tag{4}$$
where U is a uniformly random variable taking values on $\{1, \ldots, n\}$.
3. Swapping an image $\mathbf{x} \in \mathcal{X}$ can also be modeled by an augmentation, e.g.
$$A_R(\mathbf{x}) = (x_n, x_{n-1}, \ldots, x_2, x_1), \tag{5}$$
where R is a random variable.

Let us now point out a core property of augmentations which can be applied to construct new augmentations as e.g. used in our experiments in Section 3.

The following Lemma follows directly from Definition 2 and the associativity of the composition.

Lemma 1. If A_{R_1} and B_{R_2} are augmentations then $A_{R_1} \circ B_{R_2}$ is also an augmentation.

Proof. The following holds:
$$\begin{aligned}\arg\max_{y \in \mathcal{Y}}\{P(Y = y|X)\} &= \arg\max_{y \in \mathcal{Y}}\{P(Y = y|B_{R_2} \circ X)\} \\ &= \arg\max_{y \in \mathcal{Y}}\{P(Y = y|A_{R_1} \circ (B_{R_2} \circ X))\} \\ &= \arg\max_{y \in \mathcal{Y}}\{P(Y = y|(A_{R_1} \circ B_{R_2}) \circ X)\},\end{aligned}$$
where the first two lines follow from Definition 2 and the last equality follows from associativity of the composition. □

In the next section, we provide some examples on how to construct more complex data augmentation functions by using Lemma 1.

Examples (ctd.): Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a set of images, and $\mathcal{Y} = \{1, \ldots, l\} \subset \mathbb{N}$ a corresponding set of labels.

4. Classical image data augmentation is often defined by domain experts who introduce simple transformations such as horizontal flip, random rotations, and noise adding. Following Lemma 1 this can be formalized using A_N, A_U and A_R as in Examples 1-3 above as follows:
$$A_N \circ A_U \circ A_R. \tag{6}$$
5. A conditional GAN (Mirza & Osindero, 2014) can be formalized by
$$A_{R_1} \circ \ldots \circ A_{R_l} = \begin{cases} A_{R_1} \text{ if } \arg\max_{y \in \mathcal{Y}} P(Y = y|X) = 1 \\ \ldots \\ A_{R_l} \text{ if } \arg\max_{y \in \mathcal{Y}} P(Y = y|X) = l \end{cases} \tag{7}$$
where $A_{R_1}, \ldots, A_{R_l}$ are class-specific augmentations.
6. Mixup data augmentation Zhang et al. (2017) transforms the data by sampling from the vicinity distribution of the examples to enlarge the support of the data distribution. To formalize this augmentation, Definition 2 can be extended to pairs of inputs $\mathcal{X} \times \mathcal{X}$ in the usual way by using product spaces. In this way one can formalize an augmentation as a convex combination of two different inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ by
$$(1 - U) \cdot \mathbf{x}_1 + U \cdot \mathbf{x}_2, \tag{8}$$
where U is a uniform random variable on $[0, 1]$.

B EMPIRICAL ANALYSIS OF DATA AUGMENTATION (EXTENDED)

Our empirical analysis of the augmentation functions defined above is structured into three parts: (1) performance analysis, where we look at the effect of data augmentation on classification performance and adversarial robustness; (2) boundary analysis, where we analyse how the decision boundary of a model is affected by the augmentation; and (3) influence analysis, where we look at how much a model relies on augmented training samples when predicting on the real test examples and their adversarial counterparts.

B.1 PERFORMANCE ANALYSIS

Our experimental test domain is image classification, using CIFAR10. Two criteria for performance evaluation are considered: 1) classification performance, in terms of classification accuracy on a test set; and 2) adversarial robustness. To assess the latter, we carry out *Projected Gradient Descent (PGD)* (Kurakin et al., 2016; Madry et al., 2017) attacks on each trained model. Having full access to the model, we create adversarial examples for each test example, and compute the accuracy for the adversarial test set. Details of the attack models are provided in Appendix D.

We compare two augmentation scenarios: in the first, each data augmentation function is investigated independently from the others (see Section B.1.1). In the second (Section B.1.2), combinations of data augmentations are studied. More information about the training procedures is given in Section D in the Appendix.

B.1.1 INDEPENDENT DATA AUGMENTATION FUNCTIONS

In order to analyse the effect of each data augmentation function independently, we apply classical, Mixup, and GAN augmentation functions to the training data, independently and separately. More specifically, we apply the augmentations as defined in Examples 4-6 of Appendix A, and additional details about the setup in these experiments are provided in Appendix D. In each experiment, we vary the probability of augmenting with $p \in \{0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. We then report the accuracy on the test set, and the adversarial accuracy for the 4 PGD attacks across all data augmentation functions, and all p values. Each experiment is repeated 3 times, and the mean and standard deviation of the performance measures are reported in Figure 1.

B.1.2 COMPOSITION OF DATA AUGMENTATION FUNCTIONS

In a second set of experiments, we investigate two compositions of data augmentation functions: (1) Mixup with classical augmentation, and (2) GAN augmentation with classical augmentation. Classical data augmentation was chosen for this combination because it achieved the best results in the above experiments (with augmentation probability of $p = 0.5$, which we now use in the composition experiment). The probabilites for the Mixup and GAN augmentation are varied in the range of $\{0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. For the function composition experiments, we report the test accuracy and adversarial robustness, varying the value of augmentation probability in each experiment. Each experiment is repeated 3 times, and the mean and standard deviation of the performance measures are reported in Figure 4 in the Appendix.

B.2 BOUNDARY ANALYSIS

In this section we propose the following measure of smoothness for functions.

Definition 3 (Smoothness)**.** Let X be a random variable on $\mathcal{X}$. We define the *ϵ-smoothness* of some function $f : \mathcal{X} \to \mathcal{Y}$ by:

$$\text{Smoothness}(f) := P\left(f(X) = f(\mathbf{x}) \mid \mathbf{x} \in \partial B_\epsilon(X)\right), \tag{9}$$

where $\partial B_\epsilon(X)$ is the surface of a ball $B_\epsilon(X)$ around X with radius $\epsilon > 0$. In other words, for a given input $\mathbf{x}$ and its predicted label $f(\mathbf{x})$, smoothness computes the ratio of the samples on the ϵ-sphere of $\mathbf{x}$ for which the model predicts the label $f(\mathbf{x})$.

We now define the *smoothness curve* as a curve that represents *the average of the smoothness values per percentile*, for a given set of samples. The higher the percentile values, the more robust the model is.

Our smoothness measure can be related to adversarial robustness: If the boundary around a sample is smooth, then it is less likely that the prediction of the model can be flipped by an example in that neighborhood. We empirically show that if a model has smoother decision boundaries around unseen test examples, this model is more adversarially robust. Based on these observations, if the majority of samples have smooth boundaries around them, then the model is more robust. Hence, adversarial robustness increases with smoothness.

In practice, we sample a number of points from the surface of the ϵ-sphere around every $\mathbf{x}_i$, and calculate the ratio of sampled points that have the same class prediction as the label $f(\mathbf{x}_i)$, where f is the classifier model we analyse. This is our estimate of the *smoothness* around $\mathbf{x}_i$. We then compare the distribution of smoothness measures for different models in Figure 2. The details of the experimental setup are provided in Appendix D.

B.3 Influence analysis

We use *influence functions* to analyse the importance of normal, and GAN augmented training data in relation to adversarial vulnerability of the resulting classifiers. For a given test example, influence functions compute an importance value for each training point that shows how much it contributed to the prediction of that test example, by estimating the change in the loss on that test example that would result if the training point were removed from the training set (Koh & Liang, 2017). Details about influence functions can be found in Appendix E.

To analyse a classifier, we first compute the influence values of the training data (both real and GAN-augmented) for a given test example. We then compute the average over the *positive* influence values[1] for the real, and for the GAN augmented training samples separately. We repeat this over all test examples, and collect the average influence values for real and GAN data. Now we count how often in the test set, the GAN augmented examples had a higher mean influence value, compared to the mean of influence values in the real train data. This value then represents how often GAN samples were more influential compared to the real samples, over the test set. We repeat this experiment for the adversarial version of the test set. We can now compare the influence of GAN samples over the normal test set, and the adversarial test set. More information about the analysis with influence functions can be found in Section D. The results provided in Figure 3 compare classifiers that were trained with different amounts of GAN augmentation.

C Experimental Results (extended)

C.1 Performance analysis

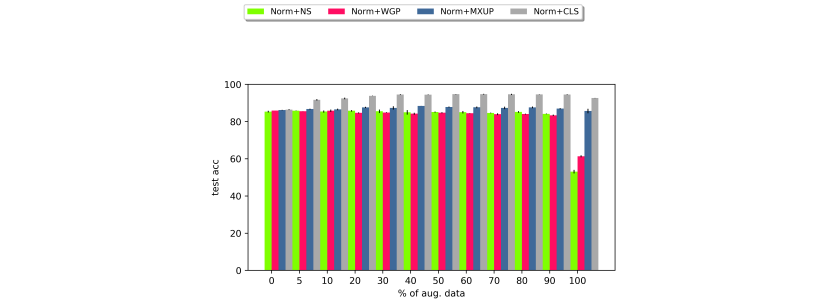
In this section, we present and discuss the results of the performance analysis for the investigated data augmentations.

C.1.1 Independent data augmentation functions

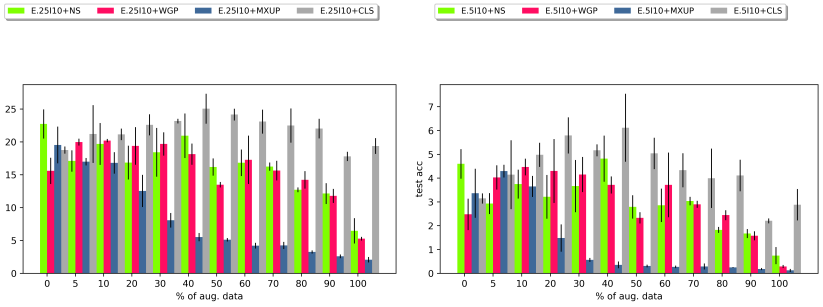
As can be seen in Figure 1, both GAN and Mixup data augmentation improve test accuracy compared to non-augmented classifiers (0% augmented), while being outperformed by classical data augmentation. Comparing GAN and Mixup with non-augmented classifiers, we observe that the adversarial vulnerability increases with the amount of augmentation in the majority of the cases. This suggests that inductive bias introduced by well-understood image transformations is more helpful, in terms of classification performance and adversarial robustness, than other augmentation methods.

It can be additionally observed that while the classification performance in both Mixup and GAN data augmentation does not change significantly when varying the amount of augmentation p (except for the extreme case of $p = 1$ in GANs), adversarial vulnerability increases considerably with p in all the attacks on the classifiers that used Mixup and GAN data augmentation. This difference is more apparent in the weaker attacks (e.g., PGD 10 iterations), which suggests that using Mixup and GAN augmentation alone increases adversarial vulnerability. With stronger attacks (PGD with 100 iteration), the difference between GAN and classical augmentation reduces, while Mixup remains the most vulnerable method.

[1] We are interested to know which training examples contribute positively to making a prediction.

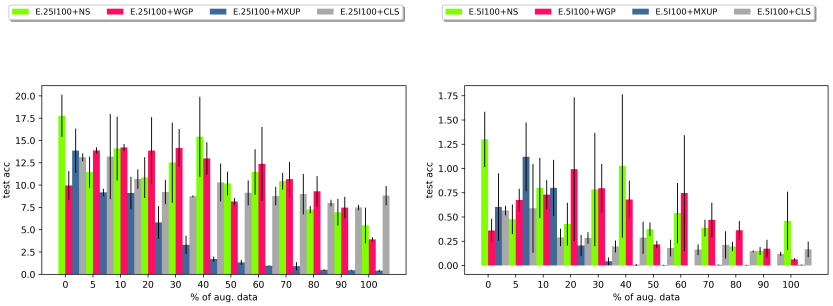


(i) Normal accuracy.



(ii) PGD accuracy with $\epsilon = 0.25$ and 10 iterations.

(iii) PGD accuracy with $\epsilon = 0.5$ and 10 iterations.



(iv) PGD accuracy with $\epsilon = 0.25$ and 100 iterations.

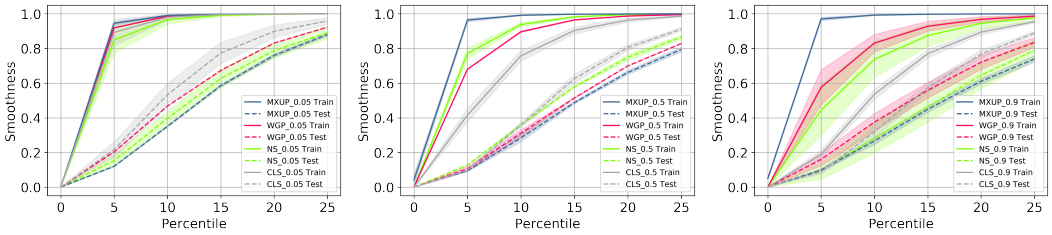
(v) PGD accuracy with $\epsilon = 0.5$ and 100 iterations.

Figure 1: *Performance Analysis:* Comparison between different augmentation methods on the test set of CIFAR10. E=ϵ in PGD. I=PGD iterations. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty. MXUP: Mixup augmentation. CLS: Classical augmentation.

A general pattern with Mixup augmentation is increasing the probability of augmentation p consistently leads to lower robustness. In GAN augmented classifiers, on the other hand, the relation between robustness and amount of augmentation differs based on the strength of the attack. In weaker attacks it appears that classifiers with $p \geq 0.5$ are becoming considerably more vulnerable than classical augmentation. In stronger attacks however, the difference between GAN augmented and classical augmented classifiers decreases.

C.1.2 Composition of data augmentation functions

We observe a different pattern in the function composition experiments. As can be seen in Figure 4 in the Appendix, the classification performance of Mixup and GAN data augmentation models reaches the performance of the classical data augmentation. Mixup seems to achieve better classification performances in all cases, when combined with classical data augmentation. While the performance



(i) 0.05 augmentation. (ii) 0.5 augmentation (iii) 0.9 augmentation

Figure 2: *Boundary Analysis:* Comparison between the effect of different augmentation methods on the smoothness of the decision boundaries of the classifier in the neighbourhood of train and test samples.

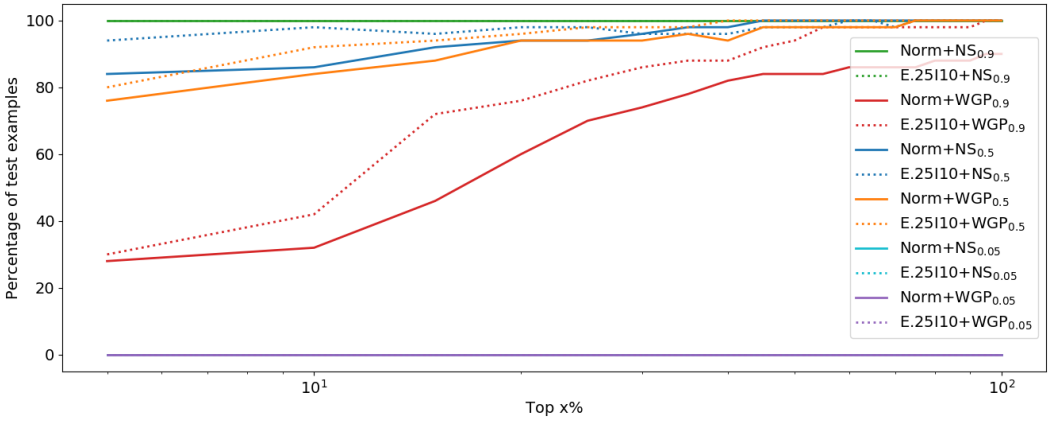


Figure 3: *Influence Analysis:* Amount of test examples that were influenced more by GAN augmentation, on average. Note: Some of the lines are overlapping: Norm+NS$_{0.9}$, E.25I10+NS$_{0.9}$ at the top and Norm+NS$_{0.05}$, E.25I10+NS$_{0.05}$, Norm+WGP$_{0.05}$, E.25I10+WGP$_{0.05}$ at the bottom.

of GAN augmented classifiers have been significantly improved by the composition with classic data augmentation, in the extreme case of $p = 1$ the performance degradation is apparent. This suggests that although $P(Y|A_{R_1} \circ \ldots \circ A_{R_l} \circ X)$ (see Definition 2 and Example 5) has been maximised in the GAN training to match the generated samples to their conditioning labels, i.e. $P(Y|A_{R_1} \circ \ldots \circ A_{R_l} \circ X) \leq P(Y|X)$.

Similar to the independent experiments, classical augmentation remains the most robust classifier in the majority of the cases. Looking at the adversarial accuracies, we can see two different patterns for the Mixup augmented classifiers and GAN augmented classifiers. The Mixup augmented classifiers show more robustness for weaker attacks (PGD with 10 iterations), compared to GAN augmented classifiers. However, in stronger attacks (PGD with 100 iterations), GAN and Mixup augmented classifiers achieve similar results. Over all, in the majority of the cases the classical augmentation achieves better robustness compared to GAN and Mixup augmentation.

C.2 Boundary Analysis

Figure 2 compares the smoothness of different classifiers with different augmentation methods and augmentation probabilities on the train and the test sets, as explained in Sections 3.2 and D.6.

We observe that overall the more probable a data augmentation is, the less smooth the classifier will become around the original training samples. However, different augmentation methods result in different amount of smoothness around *test* examples (see Figure 7). For example, Mixup augmentation that encourages a linear relationship between the training samples and the labels, results in the most smooth boundaries around training samples among data augmentations, while having the least smooth boundaries around the test examples. We can see this by looking at the 10th percentile

of the *training data* which has a smoothness very close to 1 ($\epsilon = 2$); and the 10th percentile of the *testing data* has a smoothness very close to 0 (Figure 7).

This observation is aligned with the adversarial vulnerability of Mixup, which is the most vulnerable among all methods. GAN augmentations have smoother boundaries around test examples compared to Mixup, but less smooth compared to classical; which is also in agreement with their adversarial vulnerability as reported in Figure 1.

The detailed values of different percentiles are reported as follows: For $\epsilon = 2$, the train and test sets are reported in Table 4, Table 6 respectively. For $\epsilon = 4$, the train and test sets are reported in Table 5, Table 7 respectively.

C.3 INFLUENCE ANALYSIS

In this section, we compare the performance of GAN-augmented models using their influence values on normal and GAN-augmented training data[2]. The results of influence analysis are provided in Figure 3. As can be seen, all influence values for adversarial examples (dotted line) are higher than the influence values of real test examples (solid lines). This means that the decisions of a model on adversarial examples have been influenced by GAN-augmented examples more, than the decisions on normal test examples. This shows that when a model that used GAN-augmentation wants to make a (wrong) decision on an adversarial example, it relies on GAN-augmented training examples. However, when the same model wants to predict on a real test example, then the GAN-augmented examples are not as important for this decision compared to the previous case. We can additionally observe that this influence of GAN examples increases by the amount of augmentation used in the training of the models. As the models with higher amount of GAN augmentation are more vulnerable, we can relate adversarial vulnerability to the influence of the non-robust augmented data.

D EXPERIMENTAL SETUP

All experiments have been implemented in python using `pytorch lib` (Paszke et al., 2019). For logging the experiments and controlling hyperparameters and randomness, `sacred lib` (Greff et al., 2017) is used. For analysing the results, `incense lib` (Busche, 2019) was used to fetch the results from the `sacred lib` database.

D.1 IMAGE CLASSIFICATION

Image classification experiments are carried out on CIFAR10 (Krizhevsky et al., 2009) using a ResNet50 (He et al., 2016). ResNet was trained using SGD and weight decay penalty of coefficient $5e-4$. Each classifier was trained for 200 epochs and learning rate schedule was used with initial value of 0.1, which was reduced twice by a factor of 10 every 80 epochs. This resulted in the best standard accuracy of 94.92% on the test set using classical data augmentation with probability 0.5. Our ResNet50 model achieves 96.01% accuracy on the test set using a composition of classical data augmentation with probability 0.5, and Mixup data augmentation with probability 1. (see Table 3).

D.2 ATTACK MODELS

We carry out 4 different untargeted PGD attacks with $l2$ norm. The parameters of these 4 attacks are provided in Table 1. The attacks are applied using the robustness library (Engstrom et al., 2019).

D.3 CLASSICAL DATA AUGMENTATION

For classic data augmentation, cropping with a random amount, horizontal-flipping, changing the brightness, contrast and saturation of an image by a random amount, rotation of an image by a random amount were applied with the probability p as detailed in Appendix B.

[2]In this work, we only provide influence analysis for GAN-augmentations due to computational limits.

	Iterations		norm
$\epsilon = 0.25$	10	100	l2
$\epsilon = 0.5$	10	100	l2

Table 1: PGD attack parameters used in the experiments.

	WGP	NS
FID	20.11	18.30

Table 2: FID for WGP and NS GANs on CIFAR10.

	CLS0.5	CLS0.5+MXUP1.0
Acc(%)	94.92	96.01

Table 3: ResNet50 test accuracy on CIFAR10.

D.4 MIXUP DATA AUGMENTATION

For Mixup augmentation, the parameter of the $Beta$ distribution α was set to 1, as recommended in (Zhang et al., 2017).

D.5 GAN DATA AUGMENTATION

For GAN augmentation, two GAN models namely Non-saturating (NS) GAN (Goodfellow et al., 2014) and a Wasserstein GAN with Gradient penalty (WGP) (Gulrajani et al., 2017) have been trained to their convergence. Both GANs are designed to be label-conditional, capable of generating samples that match the condition label. We evaluated these models with Fréchet Inception Distance (FID) (Heusel et al., 2017), which showed both achieved near state-of-the-art FIDs on CIFAR10 (see Table 2). Random generated samples from both GANs can be found in Figure 9 in the Appendix. Each GAN model was sampled to generate 50k samples, and was conditioned on the training labels of CIFAR10.

D.6 BOUNDARY EXAMPLES

We analysis 12 different classifiers with two values $r = 2, 4$ of r controlling the radius of the r-sphere as explained in Section 3.2. The classifiers training data are augmented using Wasserstein GAN, Non-saturating GAN, classical augmentations or Mixup augmentation. The probability of applying the augmentation is one of 0.05,0.5,0.9. As a result, we get 12 classifiers using 1 out of 4 possible augmentation methods and 1 out of 3 possible augmentation probability. We study the smoothness of these classifiers using spheres with radius $r = 2$ or $r = 4$ as explained in Section 3.2. In practice, we sample 1000 points from the r-sphere around data samples from the train or test set. We use random 10000 data sample from each of the sets to study the smoothness with respect to a classifier. We plot every fifth percentile for each classifier, r-sphere and data set. We plot the mean and the standard deviation of percentiles for the repeated experiments in Figures 2, 7 and 8 and report them in Tables 4, 6, 5 and 7.

D.7 INFLUENCE FUNCTIONS

For the influence analysis 6 different classifiers, only augmented with GAN generated data, have been analysed. The data has been generated using both GANs (NS and WGP), using low ($p = 0.05$), medium ($p = 0.5$) and high ($p = 0.9$) amount of data augmentation.

In each run we analysed the 50 same test examples and their adversarial counterparts. The adversarial examples have been generated for each trained classifier separately, using 10 iterations of PGD, with $\epsilon = 0.25$.

For computing the influence functions a PyTorch implementation[3] has been used. Since influence functions require the computation of implicit Hessian-vector products, which is, although being

[3] https://github.com/expectopatronum/pytorch_influence_functions/commit/ecce2d27e3d46b3125bb3dd963beebd7a5407959

more efficient than explicitly computing the inverse of the Hessian, still very time consuming (Koh & Liang, 2017).

In order to save computation time, preliminary experiments were conducted to determine reasonable values for the parameter `recursion_depth` and the percentage of training samples used, which we called `trainset_percentage`. Both parameters have been varied in separate experiments (recursion_depth $\in \{1, 5, 10, 50\}$ and trainset_percentage $\in \{10, 25, 50, 100\}$) and the results did not look significantly different. Due to computational limitations and the size of our classifiers it was not possible to use r_averaging > 1. According to Koh & Liang (2017) this leads to noisier results but the most influential points can still be identified. One run with the minimum parameters (`recursion_depth=1`, `r_averaging=1`, `trainset_percentage=10`) takes 1 hour on an RTX 2080 Ti for analysing 10 test examples.

E MOTIVATING INFLUENCE FUNCTIONS

In this work we investigated the impact of different augmentation methods on classification performance and adversarial vulnerability. To get a better understanding of why and how data augmentation affected both, we wanted to have a closer look at the learned classifier in terms of the training data. In order to obtain insights into the training data we are interested in the change of the models parameters if we did not have a specific training point. Due to the high number of training samples and model parameters leave-one-out retraining in this scenario is infeasible. With the use of influence functions – originally a technique from robust statistics (Cook (1977), Cook & Weisberg (1980), Cook & Weisberg (1982)) – we can efficiently approximate this parameter change Koh & Liang (2017).

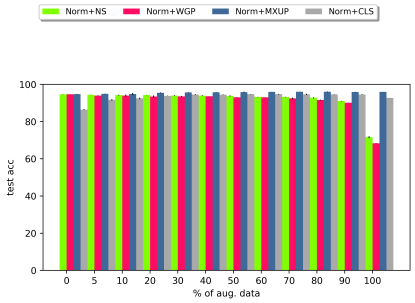
Influence functions estimate how a model's prediction would change if a training point z was removed by upweighting it slightly by some ϵ.

Koh & Liang (2017) define

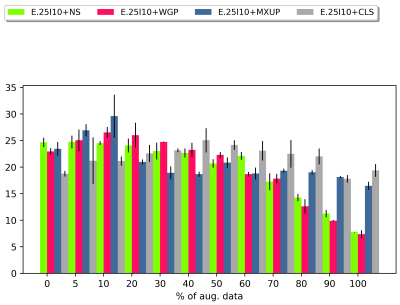
$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{10}$$

as the influence of upweighting training point z on the loss of test point z_{test}. The authors showed that influence functions can be used for a wide variety of model analysis, e.g. for debugging domain mismatch and fixing mislabeled examples (Koh & Liang, 2017). Therefore we believe that they are a suitable tool for analysing GAN augmentation in the context of adversarial vulnerability.

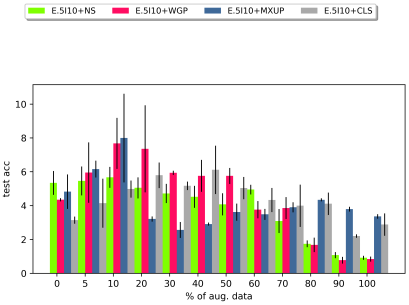
F ADDITIONAL RESULTS



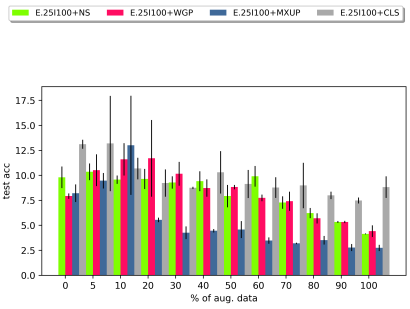
(i) Normal accuracy.



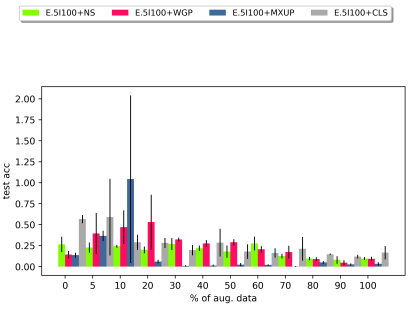
(ii) PGD accuracy with $\epsilon = 0.25$ and 10 iterations.



(iii) PGD accuracy with $\epsilon = 0.5$ and 10 iterations.

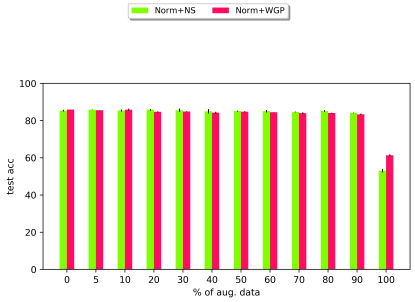


(iv) PGD accuracy with $\epsilon = 0.25$ and 100 iterations.

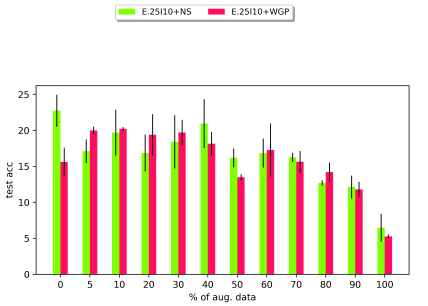


(v) PGD accuracy with $\epsilon = 0.5$ and 100 iterations.

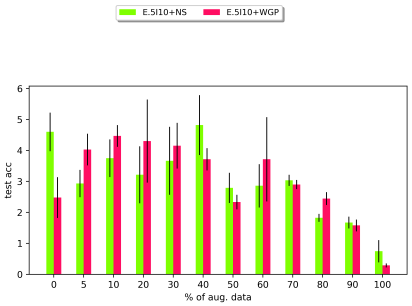
Figure 4: Comparison between different augmentation methods on the test set of CIFAR10, when classic data augmentation was additionally applied with probability of 0.5 to all methods. E=ϵ in PGD. I=PGD iterations. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty. MXUP: Mixup augmentation. CLS: Classical augmentation.



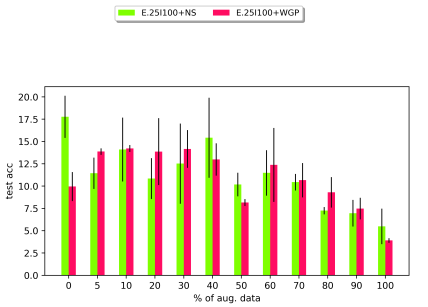
(i) Normal accuracy.



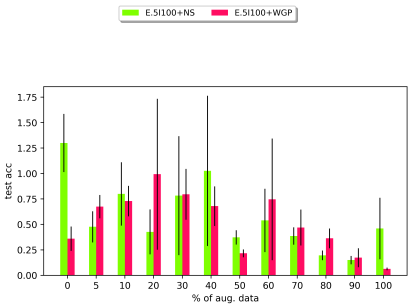
(ii) PGD accuracy with $\epsilon = 0.25$ and 10 iterations.



(iii) PGD accuracy with $\epsilon = 0.5$ and 10 iterations.

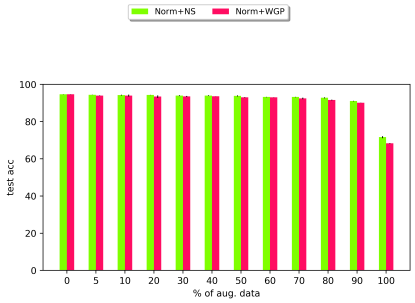


(iv) PGD accuracy with $\epsilon = 0.25$ and 100 iterations.

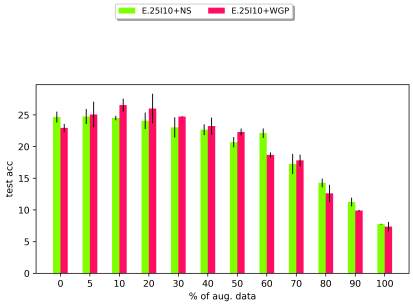


(v) PGD accuracy with $\epsilon = 0.5$ and 100 iterations.

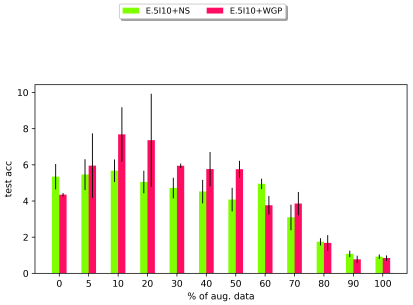
Figure 5: Comparison between GANs (NS vs WGP) for GAN augmentation on. E=ϵ in PGD. I=PGD iterations. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty.



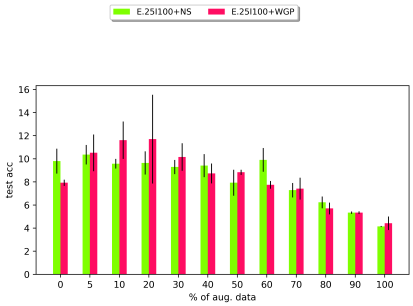
(i) Normal accuracy.



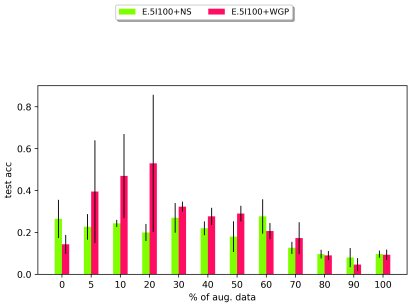
(ii) PGD accuracy with $\epsilon = 0.25$ and 10 iterations.



(iii) PGD accuracy with $\epsilon = 0.5$ and 10 iterations.



(iv) PGD accuracy with $\epsilon = 0.25$ and 100 iterations.



(v) PGD accuracy with $\epsilon = 0.5$ and 100 iterations.

Figure 6: Comparison between different GANs (NS vs WGP) for GAN augmentation on CIFAR10 dataset, when classic data augmentation was additionally applied with probability of 0.5 to all methods. E=ϵ in PGD. I=PGD iterations. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty. CLS: Classical augmentation.

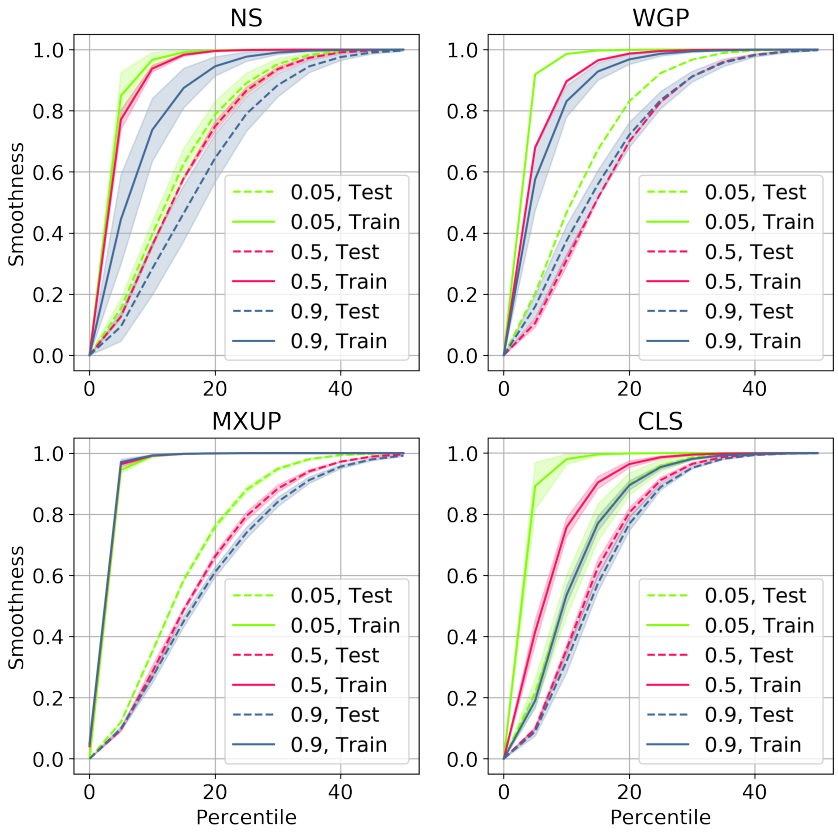
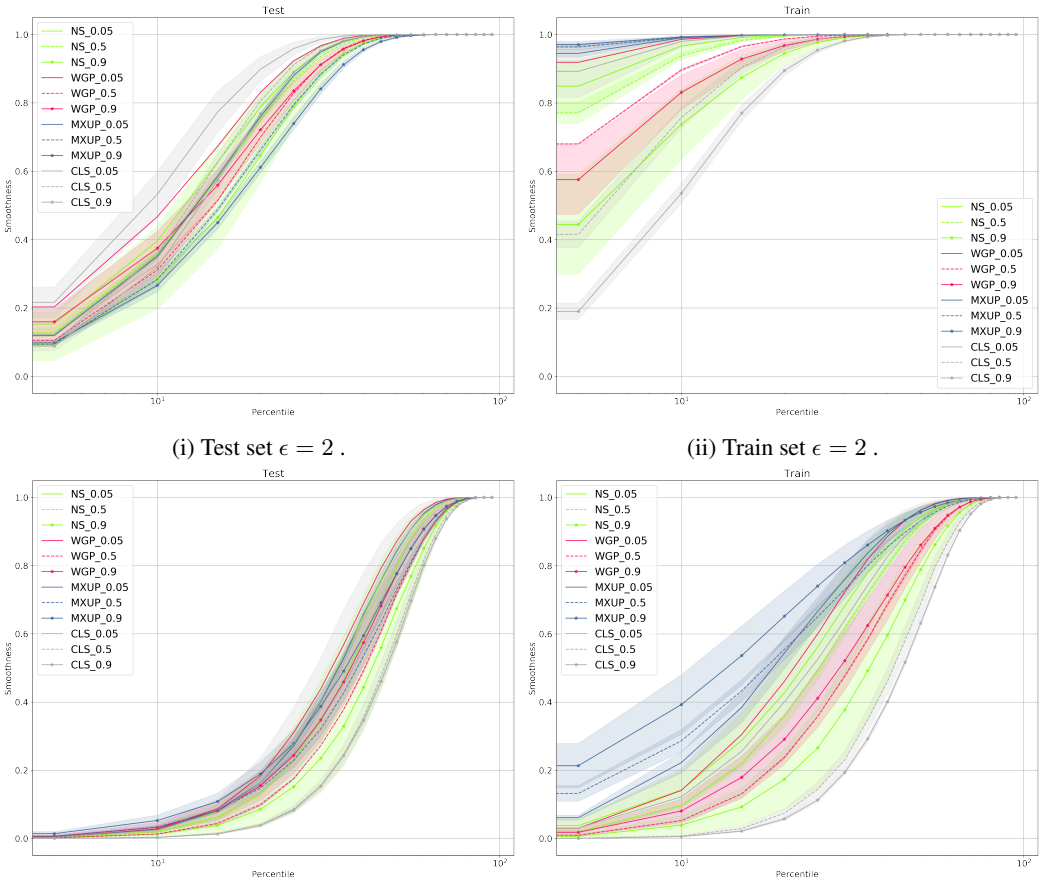


Figure 7: Comparing the smoothness of classifiers trained using different data augmentation methods, on the train and the test sets, data augmentation is applied with probabilities 0.05,0.5,0.9. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty. MXUP: Mixup augmentation. CLS: Classical augmentation as explained in Section D.6. Using ϵ-sphere with $\epsilon = 2$ as explained in Section 3.2.

aug.	prob.	mean	10th percentile	25th percentile	median	75th percentile
NS	0.05	0.97	0.97 ± 0.02	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
NS	0.5	0.97	0.94 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
NS	0.9	0.92	0.74 ± 0.10	0.98 ± 0.02	1.00 ± 0.00	1.00 ± 0.00
WGP	0.05	0.98	0.99 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
WGP	0.5	0.95	0.90 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
WGP	0.9	0.94	0.83 ± 0.05	0.99 ± 0.01	1.00 ± 0.00	1.00 ± 0.00
MXUP	0.05	0.98	0.99 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
MXUP	0.5	0.99	0.99 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
MXUP	0.9	0.99	0.99 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
CLS	0.05	0.98	0.98 ± 0.02	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
CLS	0.5	0.93	0.76 ± 0.03	0.99 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
CLS	0.9	0.89	0.54 ± 0.03	0.95 ± 0.01	1.00 ± 0.00	1.00 ± 0.00

Table 4: Smoothness over the train set for different classifiers trained with different augmentation methods. Using ϵ-sphere with $\epsilon = 2$ as explained in Section 3.2. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty. MXUP: Mixup augmentation. CLS: Classical augmentation as explained in Section D.6.



(i) Test set $\epsilon = 2$.

(ii) Train set $\epsilon = 2$.

(iii) Test set $\epsilon = 4$.

(iv) Train set $\epsilon = 4$.

Figure 8: Comparing the smoothness of different classifiers trained using different data augmentation methods, on the both train and test set, data augmentation is applied with probabilities 0.05,0.5,0.9. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty. MXUP: Mixup augmentation. CLS: Classical augmentation as explained in Section D.6. Using ϵ-sphere with $\epsilon = 2, 4$ as explained in Section 3.2.

aug.	prob.	mean	10th percentile	25th percentile	median	75th percentile
NS	0.05	0.75	0.14 ± 0.05	0.56 ± 0.10	0.94 ± 0.03	1.00 ± 0.00
NS	0.5	0.73	0.10 ± 0.00	0.49 ± 0.00	0.92 ± 0.01	1.00 ± 0.00
NS	0.9	0.64	0.04 ± 0.03	0.27 ± 0.12	0.79 ± 0.09	0.99 ± 0.00
WGP	0.05	0.77	0.14 ± 0.00	0.60 ± 0.00	0.96 ± 0.00	1.00 ± 0.00
WGP	0.5	0.67	0.05 ± 0.00	0.36 ± 0.00	0.84 ± 0.01	1.00 ± 0.00
WGP	0.9	0.69	0.08 ± 0.03	0.41 ± 0.08	0.86 ± 0.05	0.99 ± 0.00
MXUP	0.05	0.79	0.22 ± 0.03	0.67 ± 0.04	0.96 ± 0.01	1.00 ± 0.00
MXUP	0.5	0.78	0.29 ± 0.03	0.65 ± 0.03	0.93 ± 0.01	1.00 ± 0.00
MXUP	0.9	0.82	0.39 ± 0.09	0.74 ± 0.06	0.96 ± 0.02	1.00 ± 0.00
CLS	0.05	0.74	0.12 ± 0.08	0.52 ± 0.17	0.93 ± 0.05	1.00 ± 0.00
CLS	0.5	0.58	0.01 ± 0.00	0.14 ± 0.02	0.69 ± 0.04	0.99 ± 0.00
CLS	0.9	0.56	0.01 ± 0.00	0.11 ± 0.01	0.63 ± 0.01	0.98 ± 0.00

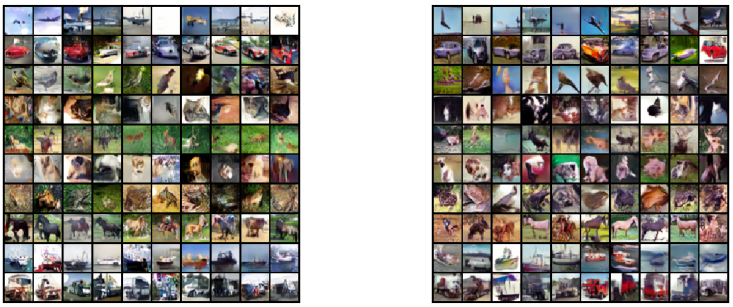
Table 5: Smoothness over the train set for different classifiers trained with different augmentation methods. Using ϵ-sphere with $\epsilon = 4$ as explained in Section 3.2. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty. MXUP: Mixup augmentation. CLS: Classical augmentation as explained in Section D.6.

augmentation	probabiltiy	mean	10th percentile	25th percentile	median	75th percentile
NS	0.05	0.86	0.40 ± 0.04	0.89 ± 0.03	1.00 ± 0.00	1.00 ± 0.00
NS	0.5	0.85	0.36 ± 0.01	0.87 ± 0.01	1.00 ± 0.00	1.00 ± 0.00
NS	0.9	0.83	0.28 ± 0.09	0.79 ± 0.06	1.00 ± 0.00	1.00 ± 0.00
WGP	0.05	0.88	0.47 ± 0.00	0.92 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
WGP	0.5	0.84	0.31 ± 0.02	0.83 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
WGP	0.9	0.85	0.37 ± 0.05	0.83 ± 0.03	1.00 ± 0.00	1.00 ± 0.00
MXUP	0.05	0.86	0.35 ± 0.01	0.88 ± 0.01	1.00 ± 0.00	1.00 ± 0.00
MXUP	0.5	0.83	0.28 ± 0.02	0.79 ± 0.01	1.00 ± 0.00	1.00 ± 0.00
MXUP	0.9	0.82	0.27 ± 0.02	0.74 ± 0.02	0.99 ± 0.00	1.00 ± 0.00
CLS	0.05	0.89	0.53 ± 0.07	0.96 ± 0.02	1.00 ± 0.00	1.00 ± 0.00
CLS	0.5	0.86	0.36 ± 0.02	0.91 ± 0.01	1.00 ± 0.00	1.00 ± 0.00
CLS	0.9	0.85	0.32 ± 0.04	0.89 ± 0.01	1.00 ± 0.00	1.00 ± 0.00

Table 6: Smoothness over the test set for different classifiers trained with different augmentation methods. Using ϵ-sphere with $\epsilon = 2$ as explained in Section 3.2. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty. MXUP: Mixup augmentation. CLS: Classical augmentation as explained in Section D.6.

augmentation	probability	mean	10th percentile	25th percentile	median	75th percentile
NS	0.05	0.65	0.03 ± 0.01	0.28 ± 0.05	0.84 ± 0.04	1.00 ± 0.00
NS	0.5	0.63	0.02 ± 0.00	0.23 ± 0.01	0.80 ± 0.01	1.00 ± 0.00
NS	0.9	0.58	0.01 ± 0.01	0.15 ± 0.07	0.67 ± 0.11	0.99 ± 0.01
WGP	0.05	0.67	0.03 ± 0.00	0.31 ± 0.00	0.87 ± 0.00	1.00 ± 0.00
WGP	0.5	0.59	0.01 ± 0.00	0.17 ± 0.00	0.72 ± 0.01	0.99 ± 0.00
WGP	0.9	0.63	0.03 ± 0.01	0.24 ± 0.05	0.78 ± 0.05	0.99 ± 0.01
MXUP	0.05	0.65	0.03 ± 0.00	0.27 ± 0.02	0.84 ± 0.02	1.00 ± 0.00
MXUP	0.5	0.61	0.03 ± 0.00	0.23 ± 0.02	0.73 ± 0.02	0.99 ± 0.00
MXUP	0.9	0.64	0.05 ± 0.02	0.28 ± 0.04	0.78 ± 0.04	0.99 ± 0.00
CLS	0.05	0.65	0.02 ± 0.02	0.28 ± 0.11	0.85 ± 0.08	1.00 ± 0.00
CLS	0.5	0.54	0.00 ± 0.00	0.09 ± 0.01	0.60 ± 0.04	0.98 ± 0.01
CLS	0.9	0.53	0.00 ± 0.00	0.08 ± 0.01	0.58 ± 0.01	0.97 ± 0.00

Table 7: Smoothness over the test set for different classifiers trained with different augmentation methods. Using ϵ-sphere with $\epsilon = 4$ as explained in Section 3.2. NS: GAN augmentation with Non-saturating GAN. WGP: GAN augmentation with Wasserstein GAN with Gradient penalty. MXUP: Mixup augmentation. CLS: Classical augmentation as explained in Section D.6.



(i) NS GAN. (ii) WGP GAN.

Figure 9: Random generated samples from the NS and WGP GANs, conditioned on different labels. The conditioning labels in each row are fixed.