Are Generative Classifiers More Robust to Adversarial Attacks?

Julien Delavande, Soël Megdoud

1 Introduction

We have chosen to work on the paper "Are Generative Classifiers More Robust to Adversarial Attacks?" from Yingzhen Li, John Bradshaw, and Yash Sharma [2]. This paper aims to answer if generative classifier are more robust than discriminative ones on adversarial attacks. The original paper also explores three detection methods for adversarial inputs in generative classifiers. Furthermore, the robustness of these detection methods is tested against several adversarial attacks types like gradient-based (including FGSM, PGD, and Carlini & Wagner (CW)).

2 Generative Classifiers

2.1 Introduction

Generative classifiers model the joint distribution $p(x, y) = p(x|y)p(y)$ of data and their labels. During inference, the classifier predicts the label using Bayes' rule:

$$p(y^*|x^*) = \frac{p(x^*|y^*)p(y^*)}{p(x^*)} = \text{softmax}_{c=1}^{C} \left[\log p(x^*, y_c)\right]$$

2.2 Deep Bayes: conditional deep LVM as a generative classifier

We introduce a latent variable z to better model the joint distribution [2]. This leads to a conditional distribution (not factotized):

$$p(x|y) = \frac{\int p(x, z, y)dz}{\int\int p(x, z, y)dzdx} \text{ and } p(x, y) = \int p(x, z, y)dz.$$

The numerator is marginalized over z to approximate the likelihood, and the denominator is constant for all classes y.

2.3 Factorization of $p(x, z, y)$

The joint distribution $p(x, z, y)$ can be factorized based on various assumptions on dependancies between x, z, et y:

$$\begin{aligned}
p(x, z, y) &= p(z)p(y|z)p(x|z, y) && \text{(GFZ)}\\
p(x, z, y) &= p_D(y)p(z|y)p(x|z, y) && \text{(GFY)}\\
p(x, z, y) &= p(z)p(y|z)p(x|z) && \text{(GBZ)}\\
p(x, z, y) &= p_D(y)p(z|y)p(x|z) && \text{(GBY)}\\
p(x, z, y) &= p_D(x)p(z|x)p(y|z, x) && \text{(DFX)}\\
p(x, z, y) &= p(z)p(x|z)p(y|z, x) && \text{(DFZ)}\\
p(x, z, y) &= p_D(x)p(z|x)p(y|z) && \text{(DBX)}
\end{aligned}$$

For example, in GFZ, z affects both x and y and x depends both on z and y whereas in BGZ z is the only link between x and y, because they are conditionally independents.

The initial character "G" to denote generative classifiers with $p(x|.)$ and "D" to denote discriminative classifiers.

Each factorisation reflects different assumptions about the data generation process. The robustness and accuracy of the models depend on the factorisation chosen.

2.4 Training with ELBO

During training, the log-likelihood of the data, $log(p(x, y))$, is often intractable to maximize. Therefore, we introduce a recognition model $q_\phi(z|x)$: an approximation to the intractable true posterior $p_\theta(z|x)$ which acts as a probabilistic encoder. This way, we can optimize the log-likelihood maximizing its variational lower bound (ELBO) [1]:

$$\mathcal{L}_{\text{VI}} = \mathbb{E}_{q(z|x,y)} \left[\log p(x, z, y) - \log q(z|x, y) \right].$$

2.5 Inference with Importance Sampling

During inference, we also need to approximate the intractable integral:

$$p(x, y) = \int p(x, z, y) dz.$$

Therefore, imporance sampling is being used to to approximate it. We reformulate the integral using the auxiliary distribution $q(z|x^*, y^*)$, which gives :

$$\int p(x^*, z, y^*) dz = \int \frac{p(x^*, z, y^*)}{q(z|x^*, y^*)} q(z|x^*, y^*) dz.$$

By sampling $z_k \sim q(z|x^*, y^*)$, we get:

$$\int p(x^*, z, y^*) dz \approx \frac{1}{K} \sum_{k=1}^{K} \frac{p(x^*, z_k, y^*)}{q(z_k|x^*, y^*)}.$$

Finally, the probability of class $p(y^*|x^*)$ is approximated as:

$$p(y^*|x^*) \approx \text{softmax}_{c=1}^{C} \left(\log \frac{1}{K} \sum_{k=1}^{K} \frac{p(x^*, z_k^c, y^c)}{q(z_k^c|x^*, y^c)} \right).$$

Therefore, $q(z|x^*, y^*)$ is used to efficiently sample z.

3 Adversarial Attacks

Robustness of generative models can be tested against different types of adversarial attack. Adversarial attacks aim to subtly modify inputs to fool the classifier. For simplicity, we will concentrate on so-called gradient attacks.

White-box gradient adversarial attacks are techniques where an attacker exploits full knowledge of a model's architecture, parameters, and gradients to generate adversarial examples. These attacks use the gradients of the model's loss function with respect to the input to craft small perturbations that mislead the model into making incorrect predictions.

3.1 Fast Gradient Sign Method (FGSM)

FGSM is a simple, single-step attack that perturbs the input in the direction of the gradient of the loss function to maximize misclassification. The adversarial example x_{adv} is computed as:

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, y)),$$

where ϵ controls the magnitude of the perturbation. FGSM is computationally efficient but less effective against robust models.

3.2 Projected Gradient Descent (PGD)

PGD extends FGSM by applying multiple iterative perturbations while ensuring the adversarial example remains within an ϵ-ball around the original input. At each iteration, the input is updated using:

$$x_{\text{adv}}^{t+1} = \text{Proj}_{\|x - x_{\text{adv}}\|_\infty \leq \epsilon} \left(x_{\text{adv}}^t + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(x_{\text{adv}}^t, y)) \right),$$

where α is the step size. PGD is more powerful than FGSM, as it thoroughly explores adversarial directions through iterative refinements.

3.3 Carlini & Wagner Attack (CW)

The CW attack formulates adversarial generation as an optimization problem to minimize the perturbation size while causing misclassification. It solves:

$$\min_\delta \|\delta\| + c \cdot \mathcal{L}_{\text{adv}}(x + \delta, y),$$

where δ is the perturbation and c balances the trade-off between minimizing $\|\delta\|$ and maximizing the adversarial loss $\mathcal{L}_{\text{adv}}$. CW attacks produce subtle, highly effective adversarial examples that are challenging to detect.

4 Detection Methods of Adversarial Attacks

In classification, if an adversarial image is mistakenly classified, this would suggest that either the image is ambiguous or that, in the context of a perfect generative model, the logit $\log p(x_{\text{adv}}, y_c)$ is be unusually low. Therefore, we can analyze the logits $\log p(x^*, y_c)$, where $c = 1, ..., C$, computed for a test input x^*. The objective is to reject both unlabeled inputs x with low marginal probability $p(x)$ (as a proxy for $p_D(x)$) and labeled data pairs (x, y) with low joint probability $p(x, y)$. Three detection methods proposed are described.

4.0.1 Marginal Detection

Marginal detection identifies inputs that are far from the data manifold by evaluating their marginal probability $p(x)$. The approach involves rejecting an input x if its marginal log-probability $-\log p(x)$ exceeds a predefined threshold δ:

$$-\log p(x) > \delta.$$

To determine the threshold δ, we compute the following statistics on the training data D:

$$\begin{aligned} \bar{d}_D &= \mathbb{E}_{x \sim D}[-\log p(x)] \quad \text{(mean of log probabilities)} \\ \sigma_D &= \text{Var}_{x \sim D}[\log p(x)] \quad \text{(variance of log probabilities)}, \end{aligned}$$

and set the threshold as: $\delta = \bar{d}_D + \alpha \cdot \sigma_D$ where α controls the tolerance for rejecting inputs.

4.0.2 Logit Detection

Logit detection evaluates the joint probability $p(x, y)$, considering both the input x and its predicted label $y = F(x)$. Inputs are rejected if the joint log-probability $-\log p(x, F(x))$ is lower than a class-specific threshold δ_y:

$$-\log p(x, F(x)) > \delta_y.$$

To compute δ_y, we calculate class-specific statistics on the training data for each class y:

$$\bar{d}_c = \mathbb{E}_{(x,y_c)\sim D}[-\log p(x, y_c)] \quad \text{(mean log-probability for class } c\text{)},$$
$$\sigma_c = \mathrm{Var}_{(x,y_c)\sim D}[-\log p(x, y_c)] \quad \text{(variance for class } c\text{)}.$$

The threshold for each class y_c is then defined as: $\delta_c = \bar{d}_c + \alpha \cdot \sigma_c$. This approach ensures that inputs with abnormally low joint probabilities are detected and rejected.

4.0.3 Divergence Detection

Divergence detection compares the probability vector $p(x)$ predicted by the classifier for an input x to a reference distribution p_c, which represents the average probability vector for class c over the training data.

Let $p_c = \mathbb{E}_{(x,y_c)\sim D}[p(x)]$ denote the mean classification probability vector for class c. For a given input x^* with predicted class c^*, the divergence $D[p_c^*||p(x^*)]$ is computed using a selected distance measure D, such as:

- **KL-divergence:**
$$D_{\mathrm{KL}}[p_c^*||p(x^*)] = \sum_i p_c^*[i] \cdot \log \frac{p_c^*[i]}{p(x^*)[i]},$$
- **Total Variation:**
$$D_{\mathrm{TV}}[p_c^*||p(x^*)] = \frac{1}{2} \sum_i \left| p_c^*[i] - p(x^*)[i] \right|.$$

The input x^* is rejected if:

$$D[p_c^*||p(x^*)] > \bar{d}_{c^*} + \alpha \cdot \sigma_{c^*},$$

where $\bar{d}_{c^*}$ and σ_{c^*} are the mean and standard deviation of the divergence measure $D[p_c||p(x)]$ computed on the training data for class c^*.

This method is particularly effective for detecting overconfident or underconfident predictions and identifying inputs that deviate significantly from the training distribution.

5 Implementation

5.1 Setup

The following tests have been done on the base of the repository of the original paper at the ICML 2019 conference. However, the repository is 6 years old and all the librairies are depreciated: Tensorflow 1.10.1 and cleverhans custom version.

Tensorflow 1.10 only works with python 3.6 maximum and the difficulty is that on free GPU environments (Google colab and Kaggle), 3.6 is no longer a version of python offered. It was therefore impossible to use the GPU environments for this project.

The solution we found was to pull a Docker image containing python 3.6, install specific tensorflow and the associated libraries inside the container and mount the code of the repository in volume. This

made it possible to run the code in the original conditions of the paper, but unfortunately on a local CPU only.

Using this limited CPU framework, we have reduced the paper's experiments to testing the various factorisations for FGSM attacks only. Other attacks take too long (e.g. 5h for a PGD attack). We also set aside the detection methods for lack of CPU time (would take days to compute).

However, we decided to test these attacks on a new dataset, Fashion MNIST. This has 10 classes like MNIST, but the photos are from fashion items. As the paper suggests that MNIST is too simple a dataset to compare generative and discriminative classifiers and that the difference is on real photo datasets, we can make the comparison.

5.2 Fashion MNIST dataset and adversarial attacks

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

Here is a visualisation of Fashion MNIST images attacked by FGSM attacks for different epsilons.

Figure 1: Fashion MNIST examples corrupted with FGSM attacks

For epsilon = 0.1, you can visually see the effect of the attack on the pixels, but the class is still easily discernible. For epsilon = 0.3, the images are really degraded.

5.3 FGSM attacks on classifiers

As in the orignal paper, we have trained the 6 classifiers architectures as described in section 2.3. Here is the success rate of attacks according to their magnitude (epsilon).

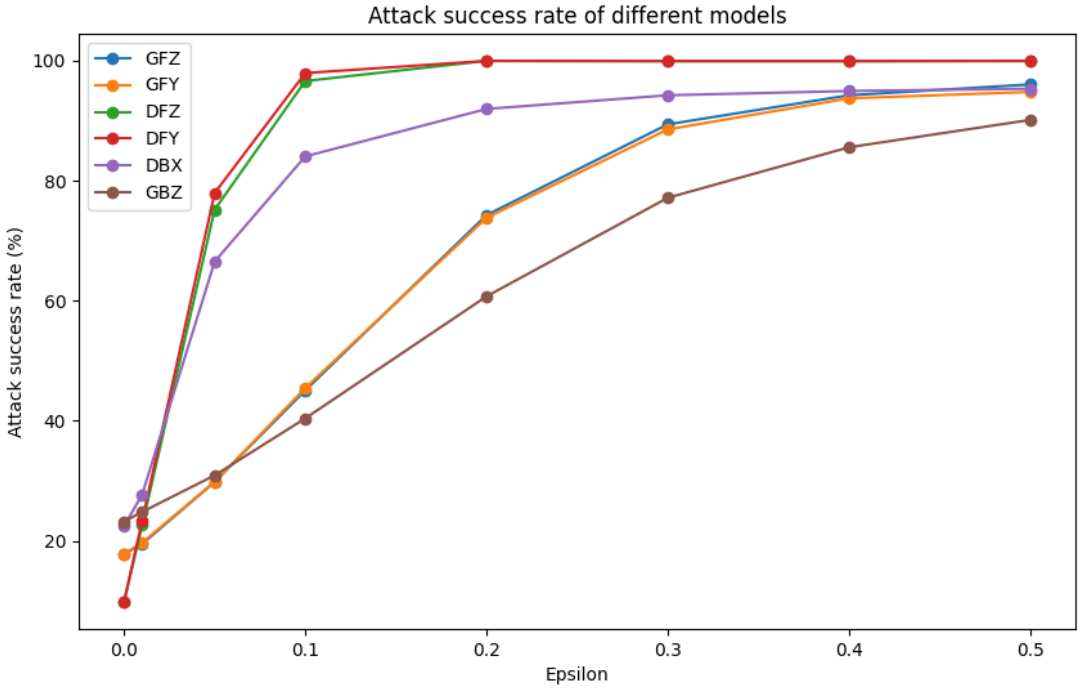

Figure 2: FGSM succes rate for different models.

You can see that the success rate of attacks increases considerably with epsilon, which is normal since the degradation of the image makes it less comprehensive. Generative models are also much more robust to attacks, with a much lower success rate. In addition, generative models are more robust when epsilon increases, since the rate of successful attacks increases much faster for discriminative models.

Another conclusion is that the fact that the dataset is photo-realistic increases the difference between generative and discriminative models, as the difference on simple MNIST is less significant in the article.

Here is a visualization of some examples classifies by a GFY model for different epsilon.

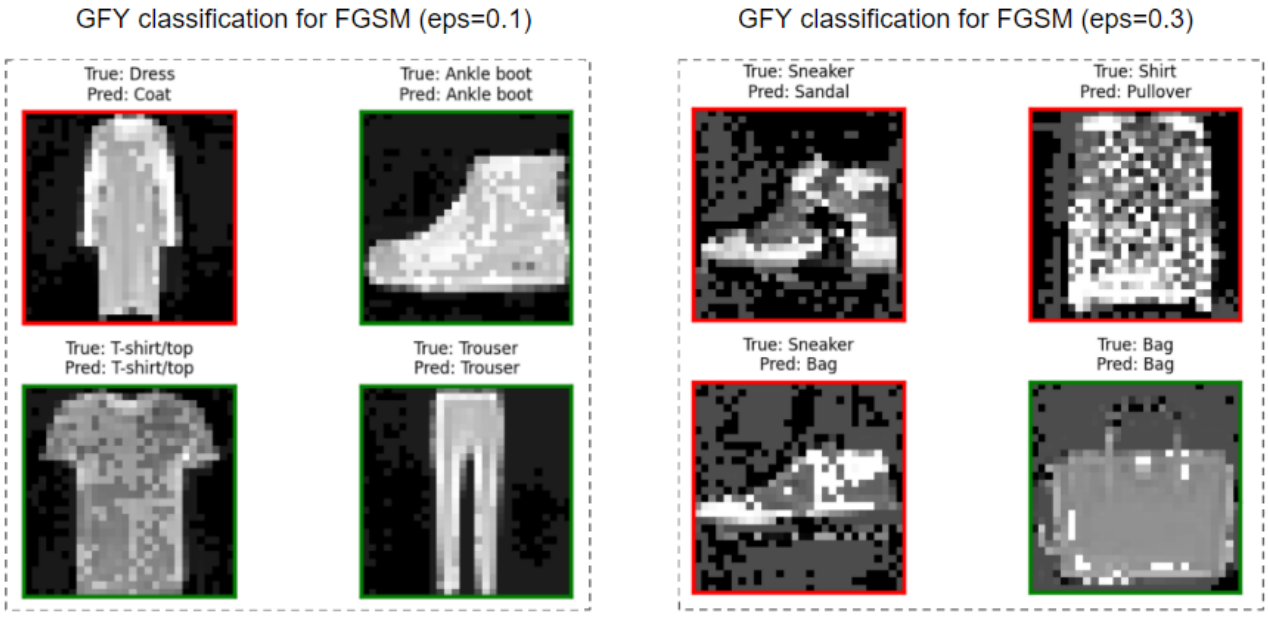

Figure 3: Classification examples of GRY model on Fashion MNIST Dataset

6 Conclusion

In this project, we explored the robustness of generative classifiers against adversarial attacks, focusing on the FGSM attack using the Fashion MNIST dataset. Our findings highlight that generative models exhibit significantly greater resilience to adversarial perturbations compared to their discriminative counterparts. This robustness becomes more pronounced as the perturbation magnitude increases, showcasing the potential of generative classifiers in real-world, photo-realistic scenarios. While computational constraints limited the scope of our experiments, the results reaffirm the conclusions from the original paper and emphasize the importance of generative approaches in adversarially robust machine learning. Future work can extend these insights to more complex attacks and datasets.

References

[1] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: `1312.6114 [stat.ML]`. URL: `https://arxiv.org/abs/1312.6114`.

[2] Yingzhen Li, John Bradshaw, and Yash Sharma. *Are Generative Classifiers More Robust to Adversarial Attacks?* 2019. arXiv: `1802.06552 [cs.LG]`. URL: `https://arxiv.org/abs/1802.06552`.