Exploring Generative Classifiers for Adversarial Robustness on MNIST

Calvin Chan, Danwen Ouyang, Duoduo Zhu, Jinglian Wu,
Marcelina Viana, Rui Qiao, and Zeyu Zhang
School of Computing
National University of Singapore
{e0530616,e0503434,e0673970,e0674582,e0573235,e0673187,e0690420}@u.nus.edu

Abstract

Adversarial robustness of deep neural networks is one critical research topic for modern machine learning. In practice, most of the neural network classifiers are discriminative, which models the distribution for the labels conditioning on the inputs $p(y|x)$. Recently, generative models have demonstrated their success in generating samples including images and texts. In fact, generative models can be used for classification by learning the joint distribution $p(y, x)$. We hypothesize that the proper modelling of the likelihood of the input may implicitly provide resilience against minute perturbations. In this work, we explored the adversarial robustness of a wide range of generative models, from Naive Bayes to variational autoencoders, and their respective discriminative counterparts. Empirically, we show that generative models are more robust at the cost of computation.

1 Introduction

Deep neural networks (DNNs) are powerful machine learning models and integrated to broad aspects of human life, enabling technologies like facial recognition, object detection, and self-driving. However, research has demonstrated that DNNs are extremely vulnerable to adversarial attacks [1]. For image classification, minute perturbations on input pixel values which are imperceptible to humans may result in drastic decrease in predictive performance of the image classifier. Adversarial robustness, the resilience against adversely manipulated inputs of the model, has critical impact on the performance and safety guarantee in real-life applications. Theoretically, it is also a stepping stone towards the generalization study of machine learning models.

Most of the deep learning models are discriminative and directly approximate the conditional $p(y|x)$. To tackle the problem of adversarial robustness, researchers have developed various adversarial training techniques that use data augmentation to enrich the coverage of input space at training stage to obtain defence against the attacks. These manners successfully improve the robustness of the model countering attacks considered during training. Unfortunately, it is hard to consider all possible attacks and new attacking methods are developed constantly. The natural question to ask is whether we can improve the robustness from the intrinsic properties of the models.

Generative classifiers model the joint distribution $p(x, y)$ and predict according to Bayes rule:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = softmax_{c=1}^{C}[\log p(x, y_c)] \tag{1}$$

where C is the set of classes. The likelihood term $p(x|y)$ models the generative distribution of the input data for each class, which enables sampling, semi-supervised learning, and prediction with missing inputs. Moreover, the robustness of generative model can be directly reflected from the choice and parameterization of the likelihood. If proper design choices are made for the likelihood,

Preprint. Under review.

when the input x is slightly perturbed to x', a robust generative classifier would ideally ensure a more consistent probability:

$$p(x, y) \approx p(x', y) \tag{2}$$

Despite such beneficial properties, generative classifiers are hard to train and have received much less investigation compared to discriminative models. There have been recent works that adopted latent variable models with neural networks and performed extensive experiments on the simplest image classification dataset on MNIST and CIFAR-10 [2, 3] . Their results demonstrated gain in robustness from generative models, but the knowledge of graphical models were not thoroughly considered.

In this work, we investigate the adversarial robustness of generative classifiers from the perspective of probabilistic graphical models (PGMs). The rationale is that the joint probability term $p(x, y)$ can be factorized in a variety of ways depending on the design of PGMs. Furthermore, situations with latent variables can also be effectively represented and computed according to the PGM. To best study the properties of the models, we choose well-prepared and easy-to-use standard image classification benchmark dataset MNIST. Several generative models closely related to PGMs will be investigated, including:

1. Naive Bayes (NB) Classifier
2. Gaussian Mixture Model (GMM) Classifier
3. Markov Random Field (MRF) Classifier
4. Variational Autoencoder (VAE) based classifier

After obtaining the models, their robustness will be assessed by the change in classification accuracy against various adversarial attacks, including white-box gradient-based attacks and black-box decision based attacks without access to the gradient. All attacks consider both l_2 and l_∞ norms. We hope to understand the best possible robust generative classifier for image classification and compare them to existing discriminative models. We also explore potentiall effective combination of probabilistic graphical models and representations learned from neural networks. Our code is publicly available at: `https://github.com/zyzhang1130/CS5340-group-project`.

2 Generative Models

In this section, we provide an overview of the generative models used to build the classifier on MNIST. We also derive their corresponding discriminative counterparts and highlight their differences. More materials regarding the detailed description, implementations, and hyperparameter settings can be found in the appendices. If not specified, the class marginals $p(y)$ are the same and omitted.

2.1 Gaussian Naive Bayes

Naive Bayes classifier as a simple yet efficient model has the assumption that the features are mutually independent. We apply Gaussian Naive Bayes model by assuming each pixel follows Gaussian distribution with standard mean μ_{iy} and deviation σ_{iy} for feature i and class y. The likelihood of an image for class y is:

$$P(x|y) = \sum_{i=1}^{d} P(x_i|y) = \sum_{i=1}^{d} \frac{1}{\sqrt{2\pi\sigma_{iy}^2}} exp(-\frac{(x_i - \mu_{iy})^2}{2\sigma_{iy}^2}) \tag{3}$$

where d is the dimension of features. Then Softmax is applied to predict final class. MLE is used to obtain the mean and variance values.

2.2 Markov Random Field

Markov Random Field is not usually for general image classification but in this project we modify the Ising model to consider an additional node of class y. The likelihood and energy function are defined as:

$$p(x|\theta, y) = \frac{1}{Z(\theta)} exp(-\sum_{c} E(x_c|\theta_c, y))$$

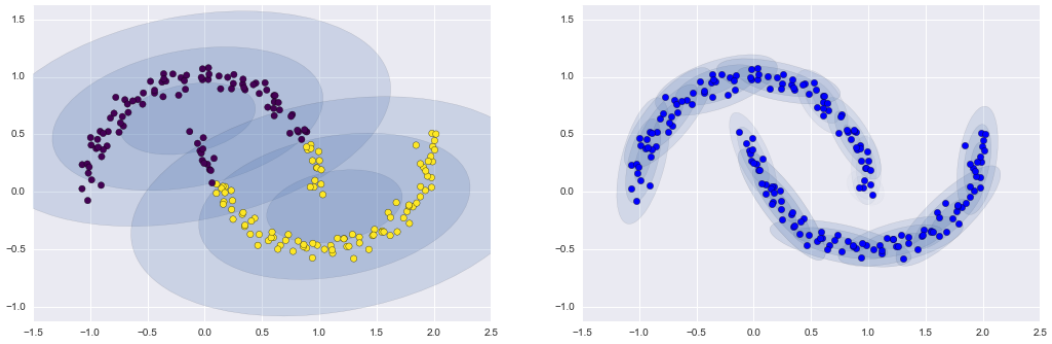
$$\sum_{c} E(x_c|\theta_c, y) = -\sum_{i,j} \theta_{i,j} x_i x_j - \sum_{m} \theta_{m,k} x_m - \sum_{j} \theta_{j,y} x_j$$

$\theta_{i,j}$ denotes potential between pairwise pixels x_i and x_i's four neighbours, $\theta_{m,y}$ denotes the unary potential between each pixel x_i and target class y, and $\theta_{j,y}$ denotes pixel x_i's potential with target class y when it an neighbour pixel. c represents clique and Z is the normalization factor. Then, softmax is applied for label selection and gradient search optimization is applied to obtain the optimal potentials for classification. After training, Markov Random Field reaches accuracy of 81% on MNIST dataset.

2.3 Gaussian Mixture Models

In unsupervised learning, Gaussian Mixture Model (GMM) is often used for clustering which uses the mixing coefficient (latent variable) to model the clusters; and the component of the mixture to model the sample data. The model evaluates the test data and returns the probabilistic cluster assignments. This is similar to the k-mean algorithm except GMM associates each cluster with a smooth Gaussian model instead of associating each cluster with a hard-edged sphere.

Although GMM clustering classifier works well in most situations, it may not work in higher dimensions and when the "shape" is not well defined or organized. Let's consider the "moon shape" dataset in 2-dimension with two clusters [4]. One GMM with two components will not be able to classify this dataset properly as shown in Figure 1a. In this situation, GMM can be used as a generative probabilistic model (Density Estimation) to describe the distribution. As in Figure 1b, 16 GMM components are used. This tends to ignore the clustering classification and many more components are used to "explain" the data. If one wants to combine both clustering classification and generative probabilistic model for density estimation in higher dimensions, then one approach is to have one GMM for each of the clusters. Each GMM will model and describe the data distribution of the sample data of a given cluster. At prediction, each cluster GMM acts as the density estimator to evaluate the test data and returns the probability likelihood by determining how likely the given test data belongs to the cluster (or least unlikely to be an outlier [5]). This is the GMM setup in this paper. One GMM is created for each of the 10 MNIST labels. The probabilities from the 10 GMM are normalized using softmax and the label with the highest probability is chosen. This model is able to reach over 94% accuracy on the MNIST dataset.



(a) Moon shape with one GMM clustering

(b) Moon shape with N GMM

Figure 1: Moon Shape Dataset

Logistic Regression, the discriminative counter part of both GMM and Naïve Bayes, is implemented in PyTorch in this paper. It achieves above 87% test accuracy on the MNIST dataset.

2.4 Neural Networks and MRF

Neural Network is a discriminative series of algorithms composed by a network of neurons. Convolutional Neural Network (CNN) is a subset of Neural Network containing at least one convolution layer. Convolution layer receives multiple inputs from the previous layer with varying weights to create better proximity. In this paper, we explore the robustness of vanilla NN and CNN, as well as CNN with MRF denoiser applied to its input layer.

Clean accuracy for all three models range from 96-97% on binarized MNIST data set. Integrating MRF bias to CNN does not seem to improve accuracy of the model without any attacks. However, it increases computational cost of the model significantly.

2.5 Variational Autoencoder

Variational autoencoder (VAE) uses maximum likelihood estimation to learn a latent representation of the distribution of interest by optimizing the following objective function

$$E[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)] \tag{4}$$

However, in the context of multi-class classification, the distribution is conditioned on its class label. Therefore, the information regarding class also needs to be modeled in conjunction with the distribution itself. In light of this recognition, we use conditional variational autoencoder (CVAE) instead of a typical variational autoencoder to deal with the classification task. The new objective function is defined as

$$E[\log P(X|z,y)] - D_{KL}[Q(z|X,y)\|P(z,y)] \tag{5}$$

where y is the class label. The new model architecture have the following form:

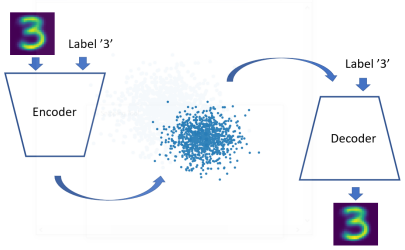


Figure 2: Conditional variational autoencoder model architecture

where the class label is concatenated with the data and fed into the encoder and decoder respectively for training. In this way, the latent distribution is now conditioned on both data and label ($P(z|X,y)$) and output of the decoder is conditioned on both latent variable and label ($P(X|z,y)$). During the testing stage, only the decoder (generative model) is used. When a test instance is given, we first sample from latent space many times to generate many distribution instances $P(X|z,y)$ per class (set to 100 per class for experiments). We use these distributions to approximate the true distributions of each class $P(X|y)$ in the way analogous to Gaussian mixture model:

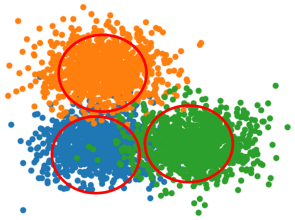


Figure 3: Using instances of $P(X|z,y)$ to approximate the true underlying distribution for each class

The approximated $P(X|y)$ are then used to gauge which class does the test instance most likely belong to. Two different metrics were test: the sum of mean squared errors of test instance with all the means of $P(X|y)$ for each class and the minimum Euclidean distance between the test instance and the $P(X|y)$ closest to test instance from each class. It was shown empirically that the second metric is more accurate by a large margin (around 20%) and therefore the second metric was chosen. This is likely due to the fact that Gaussian decays exponentially with respect to distance from it mean, and

an unweighted sum of all the means overestimates the contribution of the modes further away from the test instances. In addition, it suffices to use maximum likelihood estimation ($P(X|y)$) to predict the class label since the prior of MNIST can be treated as uniform distribution across all classes. The assumption is also validated empirically. The accuracy of the *CVAE Classifier* on unperturbed images is 88.4%.

To construct the discriminative counter part of CVAE Classifier, we remove its decoder. Instead, the sampled latent variable is now fed to fully-connected neural network with ReLU as activation function in the hidden layers and log softmax after the last layer. This time, only the data is fed to the model and the loss is the cross-entropy loss computed against the ground truth label during the training stage. During the test stage, the test instance is fed to the model and the prediction will be available at the output. This discriminative model is called VAE Classifier.

Another variant of this discriminative model is that we train a standard VAE with the objective function according to eq. (4), then use the encoder and latent layer as a pretrained feature extractor and concatenate the same fully-connected neural network to it and only train the NN layers, in the same way as described above. This discriminative model is called VAE Classifier Pre-trained.

2.6 VAE and Conditional Random Field

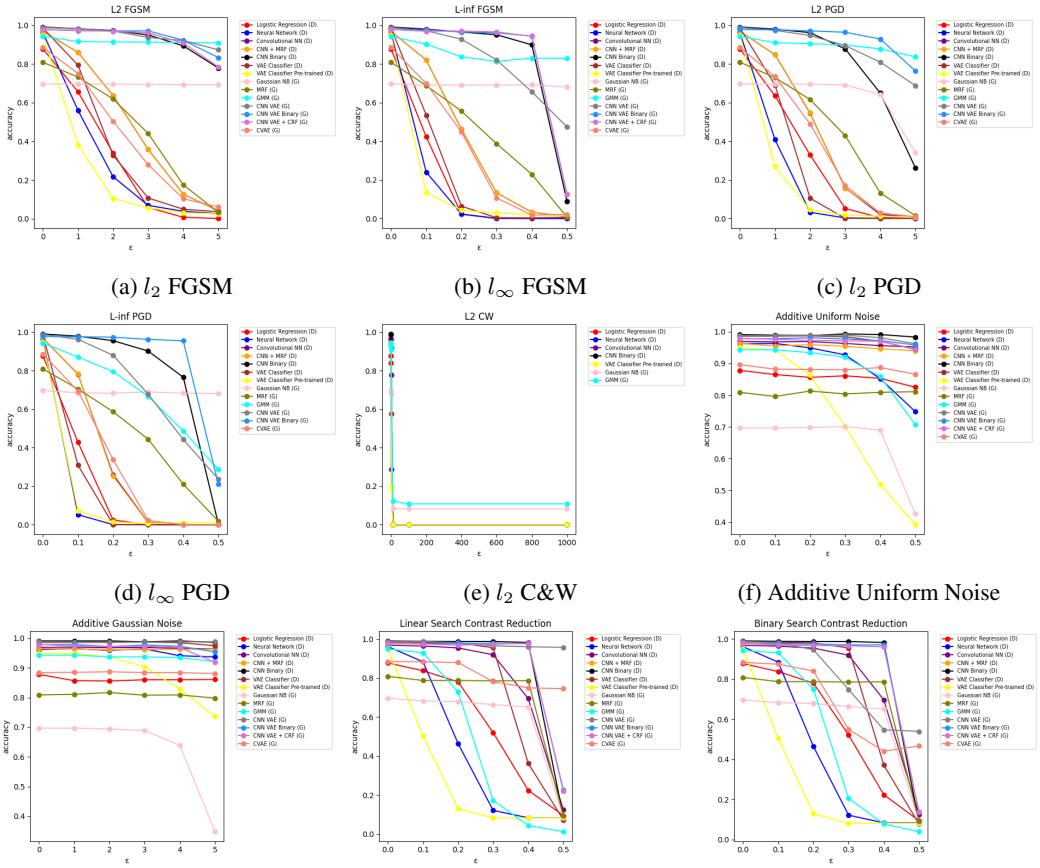
In computer vision, convolution operator often adds efficient inductive bias in the model. Therefore, we also extend the VAE that uses fully-connected neural networks from the previous section to CNN. The encoder is replaced with a shallow CNN to generate latent variable representation. The decoder is now performed by transpose convolution (the reverse operator of CNN) to generate the upsampled image from the latent vector z. We name this model *CNN-VAE*. The discriminative counterpart is the CNN-VAE with the image decoder removed. What is left is simply a CNN that maps the image to features, which is then fed into FNN layers to predict the corresponding class. This counterpart of *CNN-VAE* is called *Convolutional-NN*.

The extension from generative neural networks to graphical models here usually requires discrete variables. In literature, conditional random fields (CRF) are often combined with neural networks for computer vision tasks like image segmentation, which classifies all pixels from one image into segments that represent different objects or backgrounds. Max-Product algorithm can be used to efficiently decode the assignment of segments for few classes of objects. However, the image pixels are 8-bit with 256 values, whose decoding is computationally expensive. Fortunately, MNIST is only greyscale and the pixels can be converted to binary values using a threshold. Therefore, the CRF layer can be applied after the decoder before computing the reconstruction loss. Theoretically, CRF models the probability distribution that considers arbitrary dependency between the pixels according to their location in the image, while Euclidean distance assumes all pixels are independent when computing the loss. Despite the fact that CRF is a discriminative model for image segmentation, here we adopt CRF to decode and generate the image, which maintains the generative property. The models just described above can be transformed to take in binary images with a threshold function. Immediately, we obtain two binarized models named *CNN-VAE-Binary* and *CNN-Binary*. By adding CRF for computing the likelihood of image x, we obtain a third model *CNN-CRF* that utilizes probabilistic graphical models. All five models in this subsection are able to reach above 97% test accuracy on MNIST following the standard training and validation procedure.

3 Adversarial Attacks

The attacks we used can be either l_2 or l_∞ norm. The norm is a measure between the original input and the perturbed input. l_2 norm is also known as the Euclidean norm. In the context of our project it is the square root of the squared differences of all the pixels, whereas l_∞ norm only limits the maximum perturbation for each pixel. Therefore we chose different set of ϵ values for them.

The black-box attacks used in this project include Additive Uniform Noise Attack, Additive Gaussian Noise Attack, Linear Search Contrast Reduction Attack, Binary Search Contrast Reduction Attack and Gaussian Blur Attack. These are decision-based attacks which only rely on the output of the models. The goal is to change the prediction of a model to a specific label in a targeted attack or simply flip the prediction of a model to any incorrect label in a untargeted attack with the smallest perturbation possible.



(a) l_2 FGSM (b) l_∞ FGSM (c) l_2 PGD

(d) l_∞ PGD (e) l_2 C&W (f) Additive Uniform Noise

(g) Additive Gaussian Noise (h) Linear Search Contrast Reduction (i) Binary Search Contrast Reduction

Figure 4: Result of attacks on all models (Some results of generative classifiers on C&W attacks are missing because they take more than weeks to complete)

The white-box attacks used in this project include l_2 Fast Gradient Method, l_∞ Fast Gradient Sign Method (FGSM), l_2 and l_∞ Projected Gradient Descent (PGD) and l_2 Carlini and Wagner (C&W) attack. These are gradient-based attacks which consider the gradient of loss function with regards to the input pixels, and try to nudge the image in the direction that increases the loss.

4 Experiments

We conduct extensive experiments on MNIST Dataset scaled to range [0, 1]. All models are trained using training set and picked using validation set, while most of the models can reach more than 95% test accuracy. We run 10 types of attacks on the test set for each model across 5 different degree of perturbation ϵ. For l_∞ attacks, we run with $\epsilon = [0.1, 0.2, 0.3, 0.4, 0.5]$ since at $\epsilon = 0.5$ the adversaries are already apparent to human eyes. Similarly we run $\epsilon = [1, 2, 3, 4, 5]$ for l_2 attacks. The results are shown in Figure 4 and Table 1 and 2. In this section, we analyze the performance of each generative model with its discriminative counterpart.

Naive Bayes is more robust than its counterparts Logistic Regression for gradient-based attack. The possible reason is that loss function for Gaussian Naive Bayes is negative joint log-likelihood, which is negative log of Gaussian, and it's a convex function while Logistic Regression's loss function is least square error which is not necessary convex. Since gradient attack for example Fast Gradient Sign method usually gets perturbation direction that increase loss first, we could imagine an Naive Bayes' input feature for example $x_i = 0.8$ and $\mu_{iy} = 0.75$, to increase loss, perturbation direction would be the same direction as $\mu_{iy} \to x_i$, and x_i for example became 0.9. This has smaller impact compared with Logistic Regression especially if input image is binarized. Also, the likelihood of

	FGSM	PGD	Additive Uniform Noise	Linear Search Contrast Reduction	Binary Search Contrast Reduction	Gaussian Blur
Discriminative Models						
Logistic Regression	0.000	0.000	0.861	0.520	0.522	0.835
Convolutional NN	0.133	0.011	0.963	0.920	0.92	0.949
CNN+MRF	0.133	0.009	0.954	-	-	-
CNN Binary	0.951	0.902	**0.993**	**0.988**	**0.988**	0.978
VAE Classifier	0.004	0.000	0.984	0.956	0.956	0.971
VAE Classifier Pre-trained	0.028	0.005	0.702	0.082	0.082	0.827
Generative Models						
Gaussian NB	0.691	0.690	0.701	0.664	0.664	0.632
MRF	0.387	0.444	0.804	0.786	0.786	0.744
GMM	0.813	0.667	0.920	0.172	0.206	0.917
CNN VAE	0.821	0.678	0.989	0.966	0.749	**0.985**
CNN VAE Binary	0.961	**0.963**	0.977	0.978	0.971	0.977
CNN VAE + CRF	**0.966**	-	0.971	0.974	0.967	0.966
CVAE	0.106	0.022	0.880	0.782	0.550	0.875

Table 1: Model robustness under l_∞ attacks with $\epsilon = 0.3$

	FGSM	PGD	Additive Uniform Noise	Linear Search Contrast Reduction	Binary Search Contrast Reduction	Gaussian Blur
Discriminative Models						
Logistic Regression	0.000	0.000	0.825	0.095	0.095	0.800
Convolutional NN	0.010	0.000	0.952	0.125	0.125	**0.917**
CNN+MRF	0.010	0.000	0.940	-	-	-
CNN Binary	0.089	0.004	**0.983**	0.087	0.087	0.323
VAE Classifier	0.006	0.000	0.944	0.072	0.078	0.887
VAE Classifier Pre-trained	0.014	0.009	0.392	0.082	0.082	0.452
Generative Models						
Gaussian NB	0.680	**0.682**	0.426	0.088	0.088	0.190
MRF	0.007	0.018	0.811	0.093	0.093	0.215
GMM	**0.829**	0.287	0.707	0.012	0.040	0.847
CNN VAE	0.474	0.237	0.962	**0.958**	**0.540**	0.903
CNN VAE Binary	0.126	0.210	0.958	0.227	0.138	0.619
CNN VAE + CRF	0.125	-	0.945	0.219	0.138	0.630
CVAE	0.021	0.000	0.866	0.746	0.467	0.807

Table 2: Model robustness under l_∞ attacks with $\epsilon = 0.5$

non-target class $\hat{y}$ will not necessarily increase and give false prediction. For decision-based attack, any pixel could have change in any direction and since Naive Bayes' features are independent of each other, it's really sensitive to noise and didn't perform good enough.

Markov Random Field is not as robust as Naive Bayes and GMM for white-box attack, and this could be due to the fact that MRF's loss function is not concave and attack could happen in a direction that make feature an severe outlier. However, it is still more robust than Logistic Regression since after all, small perturbation reduces probability of target class y, but it doesn't increase probability of non-target class $\hat{y}$, and thus generative is more robust. For black-box attack, MRF performs better compared with Naive Bayes since it considers pairwise potential.

Gaussian Mixture Model provides the most surprising result as the robustness shown in Figure 4 consistently stays on top compared to the non-binary classifiers, without much compromise in test accuracy. Especially against the most common FGSM attack, it is arguably the most cost-efficient non-binary classifier due to its extremely low cost in training and testing.

CVAE Classifier outperformed both of its discriminative counterpart (*VAE Classifier* and *VAE Classifier Pre-trained*) for contrast attacks ((h) and (i) of Figure 4) by a significant margin. This result is the same as what we expected. For additive noise attack ((f) and (g)), we can see that while VAE Classifier Pre-trained performance dropped significantly as the perturbation level increase, VAE Classifier has the performance on par with CVAE and both of their performances are not severely affected by incrementally larger perturbation (both accuracies are still around 90% and above for the largest ϵ for (f) and (g) respectively). This is likely due to the fact like the additive noise attacks didn't exploit the models' intrinsic properties and hence have less visible effect on fooling the models. The cause of significant drop in performance of VAE Classifier Pre-trained needs further investigation. Gaussian blur attack (Figure 7) has similar performances to that of additive noise attack due to the same reason.

For white-box gradient-based attacks ((a) to (d)), we can see that the perturbation level has visible effect on both CVAE Classifier and its two discriminative counterparts. However, CVAE always has higher classification accuracies compared to VAE Classifier and VAE Classifier Pre-trained before all

of their accuracies eventually dropped to near zero. Figure 5 shows one instance from training data of l_2 fast gradient method perturbed by ϵ from 0.1 to 0.5.

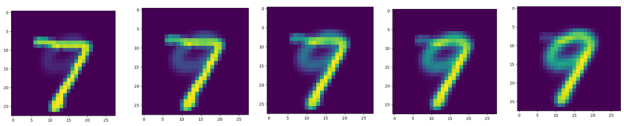


Figure 5: Image instance perturbed by l_2 fast gradient method with ϵ from 0.1 to 0.5

As ϵ increased from 0.1 to 0.5, the digit '7' is gradually morphed into digit '9'. At some point even for human it is not trivial to tell if the image is supposed to be '7' or '9'. Since CVAE Classifier does classification based on pixel-wise value difference, it works in a similar way as how human distinguish one class of image from another without the extra step of doing inference (i.e., thinking), it is natural for CVAE to be fooled. Nevertheless, it still outperformed its discriminative counterparts. Furthermore, in fact when $\epsilon = 0.2$ we can already tell the occurrence of adversarial attack judging by observing with naked eye, so the attack already defeats it purpose.

We also tested vanilla NN classifiers performance. It has a similar behavior to VAE Classifier but always worse than it (more significant for attack (h), (j) and Figure 7). This could be due to the fact that the introduction of latent space acts as a regularization (due to Gaussian constraint) and improved generalization VAE Classifier against mild amount of perturbation compared to vanilla NN. Based on the observations, in reality we can expect CVAE to have reasonably robust performances against both gradient-based attacks and black-box attacks.

CNN-VAE related models (*CNN-VAE*, *CNN-VAE-Binary* and *CNN-CRF*) are clearly more robust compared to their discriminative counterparts (*CNN*, *CNN-MRF* and *CNN-Binary* respectively) across all types of attacks. We can see that allowing binarization of the input image in the model adds significant robustness for small perturbations due to the threshold function. The defensive capability of the binary models drop sharply for l_∞ attacks with $\epsilon = 0.5$ because this is the threshold where the binarized pixel values can be flipped, which is making a larger difference on model perceiving the input image compared to the same degree of perturbation on continuous non-binary models. We can also see that adding a CRF for better generative modelling leads to no better robustness but significantly more computational cost. By observing the generated images, we hypothesize that the generative classifier with independence assumption for computing the likelihood on the pixels is already good enough on MNIST dataset. Making the model more complex may only have gains on other datasets like CIFAR and ImageNet.

Overall, We have observed that the performances across various models on Additive Noise attacks are relatively similar. This is mainly because for discriminative models, the performance under these three attacks only decreases significantly starting from $\epsilon = 0.4$, the majority of which are not captured in our experiment due to the selected standardization of ϵ values. For Contrast Reduction attacks, we can see that CNN VAE and fully connected CVAE perform better under these attacks compared to their discriminative counterparts. However, Logistic Regression and GMM performance are relatively similar. We suspect that this is because it is easier for the gradient-blind blackbox attack to exploit shallow models.

However, the run time cost of generative classifiers are also significantly higher (50 times more for VAE models, see Table 3 in Appendix) compared to discriminative models, which is arguably both good and bad. In practice, more computationally efficient inference is often preferable. On the bright side, this makes gradient-based attacks much harder to perform, especially the strongest C&W attack for CNN-VAE require weeks to run one iteration.

5 Conclusion

In general, generative classifiers are more robust compared to their discriminative counterparts, due to the explicit modelling on the likelihood of input images. The trend of robustness is clearly demonstrated for gradient-based attacks, and somewhat clear on blackbox attacks. In future, it is promising to extend these models to more realistic datasets including CIFAR and ImageNet.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2014.

[2] Y. Li, "Are generative classifiers more robust to adversarial attacks?," in *ICML*, 2019.

[3] L. Schott, J. Rauber, M. Bethge, and W. Brendel, "Towards the first adversarially robust neural network model on mnist," *arXiv: Computer Vision and Pattern Recognition*, 2019.

[4] J. VanderPlas, *python data science handbook*, ch. 5. O'Reilly Media, Inc., 2016.

[5] R. Laxhammar, G. Falkman, and E. Sviestins, "Anomaly detection in sea traffic - a comparison of the gaussian mixture model and the kernel density estimator," pp. 756 – 763, 08 2009.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[7] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[8] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," pp. 3–14, 11 2017.

6 Appendix

A Model Description and Implementation

A.1 Gaussian Mixture Models

MNIST dataset has 10 digits/clusters. The observation x_1 to x_n are assumed to be independent from a random vector X with dimension D, where D is 784 (28 x 28). 10 GMMs are created, one for each of the 10 clusters. All GMMs are assumed to have the same numbers of components which is given during initialization. Each GMM is a standard GMM where the mixing coefficient is categorical and assumed to be drawn from a discrete random variable and the component of the mixture is a multi-variate Gaussian.

Based on GridSearch, 60 components and Spherical covariance types seem to provide the optimal result for this particular setup and dataset. The model is shown in Figure 6.

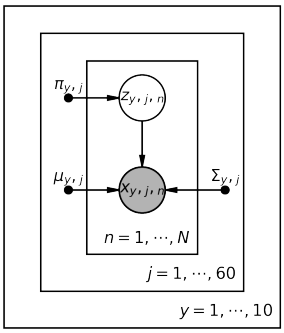


Figure 6: The plate notation of the gaussian mixture model

The probabilities from the 10 GMM are normalized using softmax and the cluster with the highest probability is chosen:

$$\underset{y}{\operatorname{argmax}} \left(softmax_{y=1}^{Y=10} [\Sigma_{j=1}^{J=60} \pi_{y,j} \mathcal{N}(x|\mu_{y,j}, \Sigma_{y,j})] \right)$$

where:
j, the number of Guassian components. J is initialized to 60 in this setup
$\pi_{11}...\pi_{yj}$, the weights of the mixing coefficient
$\mu_{11}...\mu_{yj}$, the mean of each component of the mixture, with dimension 784 (28 x 28)
$\Sigma_{11}...\Sigma_{yj}$, the variance of each component of the mixture, with dimension 784 x 784

Here, Y is 10 representing the 10 labels of the MNIST dataset.

A.2 Logistic Regression

The Logistic Regression model is implemented in PyTorch using the built-in linear function. The linear function takes in the 784 pixel values as the list of independent variables and output a tensor with 10 elements, each represents the probability of one of the MNIST labels. This tensor is fed into PyTorch CrossEntropyLoss criterion that combines negative log likelihood and also the softmax function in order to normalize the results during the loss optimization. From PyTorch documentation, the CrossEntropyLoss equation is given by:

$$loss(x, class) = -log\left(\frac{exp(x[class])}{\sum_j exp(x[j])}\right) = -x[class] + log\left(\sum_j exp(x[j])\right)$$

A.3 VAE and CRF

To model $P(x|y, z)$, we convert the class label y into 10-dimensional one-hot encoding and concatenate it with the latent vector z. The reconstruction loss on the input is also computed using Euclidean distance. At prediction time, for each class y, we sample k (by default 10) vectors from the posterior distribution $P(z|y)$, then the decoder generates k images in order to compute the likelihood of x_{test} by average the k reconstruction losses. The class with the highest posterior on x_{test} is chosen as prediction.

During implementation, one problem for the binary models is that the threshold function is not differentiable and thus disable the gradient-based attacks. Fortunately, there is a common trick in deep learning known as straight-through estimator (STE) such that we assign a explicit gradient (e.g. linear) when performing backpropagation.

The CNN-based VAE has the same architecture used in [2]. However, their model is implemented in Tensorflow and has no easy combination to graphical models. Thus, we reimplemented the repository to the more friendly Pytorch framework and verify the correctness of implementation by successful reproduction of their results in the paper.

B Adversarial Attacks

B.1 Black-box Attacks

Additive Uniform Noise Attack

Additive Uniform Noise Attack samples uniform noise with a fixed size with respect to a specific distance measure. The noise is then injected into an image to fool the model to flip its prediction. We have tested our models against the l_∞ version of this attack.

Additive Gaussian Noise Attack

Additive Gaussian Noise Attack samples Gaussian noise with a fixed size with respect to a specific distance measure. The noise is then injected into an image to fool the model to flip its prediction. We have tested our models against the l_2 version of this attack.

Linear Search Contrast Reduction Attack

Linear Search Contrast Reduction Attack reduces the contrast of the input image using a linear search in order to find the smallest perturbation required to flip the model's prediction. We have tested our model against the l_∞ version of this attack.

Binary Search Contrast Reduction Attack

Binary Search Contrast Reduction Attack reduces the contrast of the input image using a binary search in order to find the smallest perturbation required to flip the model's prediction. We have tested our model against the l_∞ version of this attack.

Gaussian Blur Attack

Gaussian Blur Attack blurs the input image using a Gaussian filter with linearly increasing standard deviation so as to flip the model's prediction. We have tested our model against the l_∞ version of this attack.

B.2 White-box Attacks

Fast Gradient Method

Fast Gradient Method (FGM) is an intuitive way to perturb the pixels in the direction that increases the loss function. The amount of perturbation is clipped by the given values of ϵ, which can be seen as the radius of the l_2 ball it projects onto. Given a loss function $J(x; w)$, FGM creates an attack x by:

$$x = x_0 + \epsilon \cdot \frac{\nabla_x J(x_0; w)}{\|\nabla_x J(x_0; w)\|_2}$$

Fast Gradient Sign Method

Fast Gradient Sign Method (FGSM) [6] is a l_∞ variant of FGM. It calculates the gradient and takes only the sign. Each pixel is then changed by either $+\epsilon$ or $-\epsilon$:

$$x = x_0 + \epsilon \cdot sign(\nabla_x J(x_0; w))$$

Projected Gradient Descent

Projected Gradient Descent (PGD) is a more powerful multi-step adversary[7]. Gradient steps are taken in the direction of greatest loss repeatedly and projected back into l_2 or l_∞ space if necessary. We have tested our models against both l_2 and l_∞ PGD, with 50 and 40 iterations respectively.

Carlini and Wagner Attack

The Carlini and Wagner (C&W) attack considers a rectifier function to better optimize the mode of attack[8]. Given a model with logits Z, the attack uses gradient descent to solve:

$$\underset{x}{minimise} \|x - x_0\|^2 + \lambda \cdot max\{(\underset{i \neq t}{max}\{Z(x)_i\} - Z(x)_t), -\kappa\}$$

The difference $\underset{i \neq t}{max}\{Z(x)_i\} - Z(x)_t$ is used to compare the target class t with the next-most-likely class. We can fix this by taking the maximum of this quantity with $-\kappa$ that controls the confidence of the adversarial examples.

The C&W attack usually has higher time complexity compared to the other attacks mentioned above. It requires a binary search to find the regularization parameter λ, and a gradient descent with small step size and large number of steps.

C Experiments

We show some additional experimental results here due to the space limit.

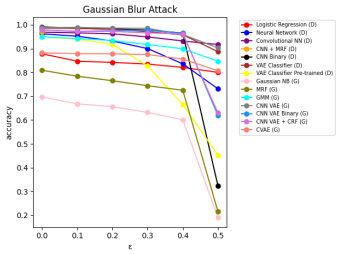


Figure 7: Gaussian Blur Attack

The generative model is computationally much more expensive compared to discriminative models, which can be seen from Table 3.

	l_2 FGM	l_∞ FGSM	l_2 PGD	l_∞ PGD
CNN VAE	115s	119s	4293s	3441s
CNN	3s	3s	85s	70s

Table 3: Attack runtime cost for *CNN-VAE* generative model vs *CNN* discriminative model