Inverted Generator Classifier

accurate and robust gradient-descent based classifier

JeongIk Cho[1]

Dept. of Computer Science and Engineering[1]

College of Engineering[1]

Konkuk University, Seoul, Korea[1]

jeongik.jo.01@gmail.com[1]

Abstract

In the field of deep learning, traditional classifier takes input data and output predicted labels. The conditional GAN receives the latent vector and the condition vector, and generates data with the desired condition. In this paper, I propose an inverted generator classifier that predicts the label of input data by finding condition vectors and latent vectors that can generate input data by using a generator of conditional GAN. Inverted Generator Classifier uses the trained generator of conditional GAN as it is. To find the data closest to the input data, Inverted Generator Classifier takes the latent vector of the generator for each condition as a variable and model parameters as constants, and performs gradient descent repeatedly to find the data closest to the input data. Then, among the data generated for each condition, the condition vector of the data closest to the input data becomes the predicted label. Inverted Generator Classifier is slow when predicting because it predicts based on gradient descent, but accuracy is high and very robust against adversarial attacks [1] such as noise.

Abbreviations

Inverted Generator Classifier: IGC

1. Inverted Generator Classifier

1.1 Training

IGC uses a trained generator of conditional GAN as a model. Any of several methods [2, 3, 4] for training conditional GAN can be used. Also, no additional training is required after training the conditional GAN.

1.2 Prediction

IGC performs prediction through two stages: latent space search and decision. Latent space search takes the latent vector as variables for each condition, and model parameters as constants, and uses two losses: data difference loss and latent restriction loss, to find the latent

vector that can generate the data closest to the input data. The latent space search is to find a latent vector that minimizes these two losses through multiple gradient descent.

The data difference loss is the loss to find the latent vector that can generate the data closest to the input data for each condition. The latent restriction loss is a loss for the latent vector not to search beyond the latent space too much.

The loss for IGC to perform latent space search is as follows.

$$L = L_{DD} + \lambda_{LR} L_{LR}$$

$$L_{DD} = \sum_{for\ each\ (cnd,ltn)\ in\ S_{in\_vec}} dif(G(cnd, ltn), d)$$

$$L_{LR} = \sum_{for\ each\ (cnd,ltn)\ in\ S_{in\_vec}} average\left(abs\left(z\_score(ltn)\right)\right)$$

L is the loss for IGC to perform latent space search through gradient descent. L_{DD} is data difference loss, and L_{LR} is latent restriction loss. λ_{LR} is the weight of latent restriction loss. S_{in_vec} is a set of tuples cnd (condition vector) and ltn (latent vector). S_{in_vec} has cnd corresponding to each class as many as the number of classes. For example, if there are 10 classes, S_{in_vec} has 10 tuples and the condition vectors of each tuple are all different.

G is a trained generator of conditional GAN. $G(cnd, ltn)$ is one data generated by G by receiving cnd and ltn. d is one input data. dif is a function that measures the difference between two data. z_score is a function that calculates the z score of each element of the input vector based on the distribution of latent vectors used when training G. For example, if G was trained by 3 dimension latent vector, each element of which follows a Gaussian distribution (mean=0, standard deviation=1), $z\_score([1,2,-3])$ is $[1,2,-3]$. abs is a function that converts each element of the input vector to an absolute value. $average$ is a function to find the average of each element of the input vector.

To reduce L, latent vectors for each conditions are taken as a variable and model parameters as constants, and gradient descent is performed. If gradient descent is repeatedly performed a certain number of times, the latent space search stage ends.

In the decision stage, a difference between data generated for each condition and input data is measured by a dif function, and IGC decides the condition with the smallest difference as a predicted label.

Label	Condition Vector				Latent vector		
num 0	1 (untrainable)	0 (untrainable)	0 (untrainable)	...	0.3 (trainable)	-1.0 (trainable)	...
num 1	0 (untrainable)	1 (untrainable)	0 (untrainable)	...	-0.2 (trainable)	0.1 (trainable)	...
num 2	0 (untrainable)	0 (untrainable)	1 (untrainable)	...	0.7 (trainable)	-0.3 (trainable)	...
...	...	...	...	...	...	...	...

Fig.1 Example of input vector of IGC

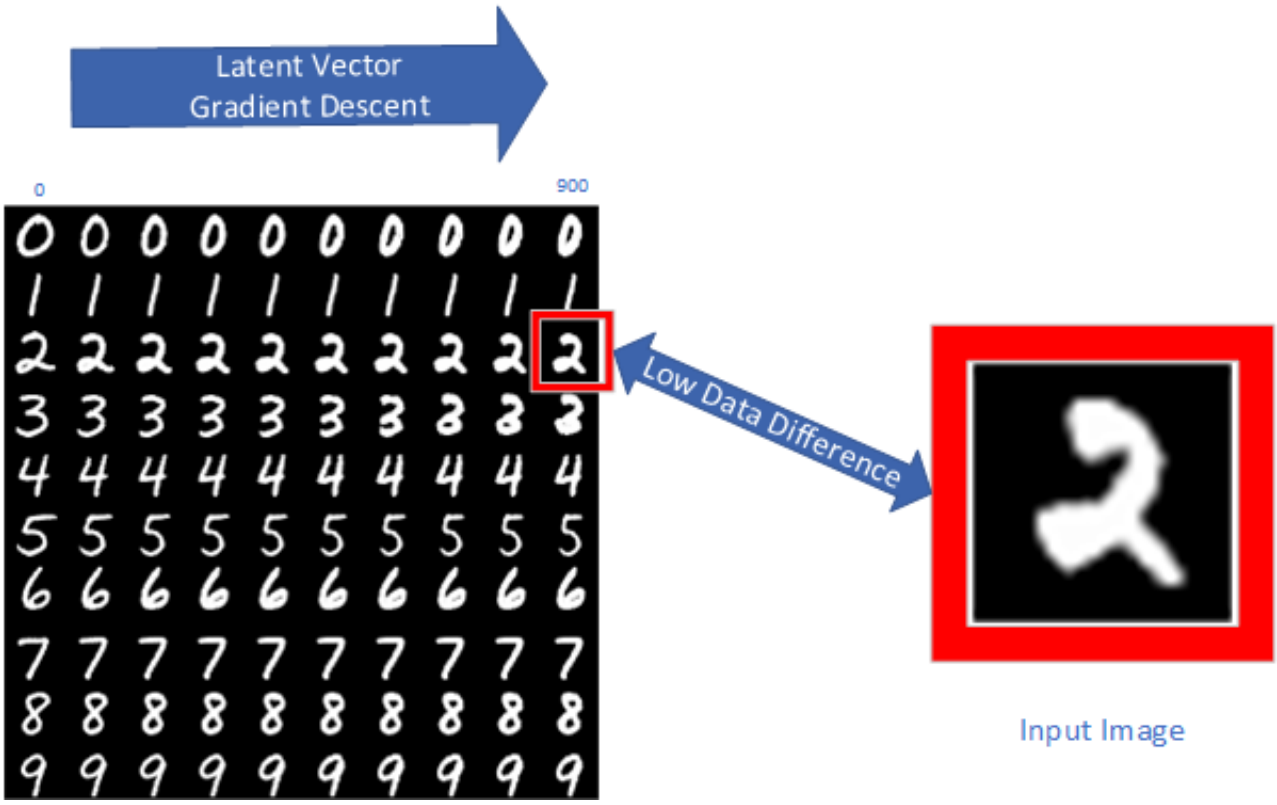

Fig.2 Prediction process of IGC

Fig. 1 is an example of an input vectors of IGC. The condition vector, which is an untrainable variable, does not change when performing gradient descent. However, the latent vector, which is a trainable variable, changes with every gradient descent. Fig. 2 shows the process of IGC prediction. The left-most column is data generated for each condition before gradient descent. Initially, the latent vector is initialized with the average of the latent vector distribution used during generator training. That is, at first, all latent vectors for each condition are the same. Later, the latent vector is changed to produce an image close to the input image through gradient descent. The right-most column shows data generated for each condition after performing gradient descent 900 times. After performing a gradient

descent to some extent, the input condition vector to generate data with the closest distance to the input image be the predicted label of the IGC.

1.3 Multi-label classification

In the case of multi-class classification on one data, IGC can classify through one prediction. However, in order for IGC to perform multi-label classification on one data, prediction is required as many as the number of labels. That is, the condition vector for one label is set as an untrainable variable, and the condition vector of remaining labels and latent vector are set as trainable variables.

2. Experiment

Tensorflow 2.1 without compile option and rtx2080ti was used for the experiment. In this experiment, I used the training dataset of the MNIST handwriting number dataset [5] (60000 images for training, 10000 images for test, 28x28x1 resolution).

I used conditional activation GAN with LSGAN adversarial loss to train conditional GAN. The generator receives a 10-dimensional condition vector and a 256-dimensional latent vector. All elements of the latent vector used in training follow the Gaussian distribution with mean = 0 and standard deviation = 1. The average FID [6] for each condition of the generator after training was measured to be 2.0. Since the MNIST dataset has one channel and their resolution is too low for the inception network, the width, height, and channel are tripled for the FID evaluation (84 × 84 × 3).

For prediction of IGC, gradient descent was performed 100 times for each image, and Adam optimizer [7] (learning rate = 0.001, beta1=0.9, beta2 = 0.999) was used. The latent restriction loss (λ_{LR}) is 0.03 and the dif function is mean absolute error. All latent vectors are initialized to 0. 1000 data randomly selected from the MNIST test dataset were used for the prediction evaluation.

First, I tested the change in accuracy according to the change in the intensity of random Gaussian noise to test images.

The original image was normalized from -0.5 to 0.5, and Gaussian noise with an average of 0 and a standard deviation of 1 was multiplied by sigma σ and added to the original image, and clipped -0.5~0.5 to keep the image stay within range.

$$noised\ image = original\ image + \sigma * noise$$

Test Data size	1000	1000	1000
Sigma	0	0.2	0.4
Accuracy (%)	95.1	94.7	91.2
Time(sec)	2526	2535	2586

Fig.3 Correct case of IGC predict. Without noise. Number 6 in right side is the input image.

Fig.3 Correct case of IGC predict. sigma=0.4. Noised number 2 in right side is the input image.

Fig.4 Incorrect case of IGC prediction. Number 9 in right side is the input image but IGC predicted number 8.

Fig.4 Incorrect case of IGC prediction. Noised number 8 in right side is the input image but IGC predicted number 0.

3. Conclusion

Inverted Generator Classifier is very slow when predicting because it predicts based on gradient descent, but accuracy is high and very robust against adversarial attacks such as noise.

4. References

[1] Xiaoyong Yuan, Pan He, Qile Zhu, Xiaolin Li

Adversarial Examples: Attacks and Defenses for Deep Learning

https://arxiv.org/abs/1712.07107

[2] Mehdi Mirza, Simon Osindero

"Conditional Generative Adversarial Nets", arXiv preprint arXiv:1411.1784, 2014.

https://arxiv.org/abs/1411.1784 (accessed 16 February 2020)

[3] Augustus Odena, Christopher Olah, Christopher Olah, Jonathon B Shlens, Jonathon Shlens

Conditional image synthesis with auxiliary classifier GANs

ICML'17: Proceedings of the 34th International Conference on Machine Learning – Volume 70, 2017, pp. 2642-2651

[4] Jeonglk Cho, Kyoungro Yoon

Conditional Activation GAN: Improved Auxiliary Classifier GAN

http://vixra.org/abs/1912.0204

[dataset] [5] Yann LeCun, Corinna Cortes, Christopher J.C. Burges

THE MNIST DATABASE of handwritten digits

http://yann.lecun.com/exdb/mnist/

[6] Heusel, Martin and Ramsauer, Hubert and Unterthiner, Thomas and Nessler, Bernhard and Hochreiter, Sepp

GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium Advances in Neural Information Processing Systems 30 (NIPS), 2017, pp. 6626-6637

http://papers.nips.cc/paper/7240-gans-trainedby-a-two-t

[7] Diederik P. Kingma, Jimmy Ba

Adam: A Method for Stochastic Optimization

https://arxiv.org/abs/1412.6980