DATA AUGMENTATION WITH GAN

Tony Liu
College of Engineering
University of Miami
yxl1837@miami.edu

ABSTRACT

When working with machine learning, insufficient amount of available data can limit the development of a model. Traditional data augmentation such as applying simple transformation to generate "new" data is one possible solution to this problem.

This paper is trying to explore the potential application of Generative Adversary Network (GAN) on data augmentation. That is, to train a classifier on GAN-generated data samples.

A variant of state-of-art class-conditional GAN, DiffAugment-based BigGAN is used as the generator in this paper. This paper then developed DenseNet and DIANet on CIFAR-10 data where part of the original data is replaced or augmented and then compare their performance.

The paper demonstrates that naively training a classifier on GAN-generated samples may not achieve the performance as that on the original dataset.

Index Terms— GAN, BigGAN, Data Augmentation, DenseNet, DIANet

1. INTRODUCTION

Dataset plays an important role in the field of machine learning. When training a model, the dataset is expected to be not only sufficiently large but also diverse enough so that it is a good representation of the reality. A representative dataset makes it easier for the model to generalize on future unseen data samples.

In practice, there are cases where dataset can be insufficient and collecting more data is infeasible. Usually, there are four solutions to that problem. 1) Using a simpler model that available data can afford. However, since the final performance of a model can be relevant to the model's complexity, a simpler model may not achieve desired performance. 2) Using ensemble methods. Multiple smaller models can be trained on available data and then combined for better performance, but this would add extra complexity to the work. 3) Transfer learning. This is potentially a better solution but if the problem is domain-specific, it could be difficult to find a pre-trained model for transfer learning. 4) Data augmentation. This is probably so far, the most used solution if there is lack of data. In fact, which one of the above solutions to use depends on the problem and available data at hand. This paper focuses on image data augmentation only.

2. BACKGROUND

2.1 Data Augmentation

Traditionally, the idea of data augmentation on image data is to apply transformation such as rotating, scaling, zooming, changing brightness, on existing data to generate "new" data. For example, in Fig 2.1, the original image is used to generate three other images, so instead of only one image, we now have four images to use.

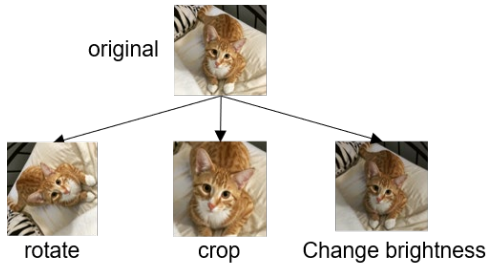


Fig 2.1 Traditional Data Augmentation

Data augmentation as in Fig 1.1 is reasonable since if a cat is rotated, cropped or the brightness is changed, it is still a cat. The model is thus trained to recognize rotated cat, cropped cat or cat of different brightness so that it would correctly classify those images where the cat is not well-positioned.

Data augmentation is an easy and inexpensive way to add more data to available dataset although the images generated in this way may not be as good as additional brand-new independent examples.

2.2 GAN and Class-conditional GAN

Recent years have witnessed remarkable improvement in Deep Generative Models, especially Generative Adversarial Networks (GAN), which is able to generate nearly photorealistic images.

A typical GAN has two components as in Fig 2.2: A generator and a discriminator, competing against each other, which is the reason why it is called generative adversarial network.

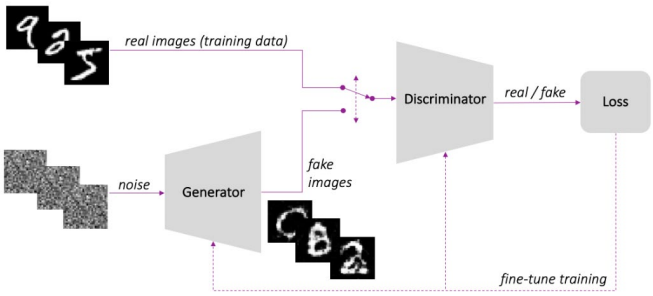


Fig 2.2 Typical GAN architecture

One the one hand, the generator is trying to generate fake data, sending it to the discriminator and wants the discriminator fail to identify it. On the other hand, the discriminator is receiving both the data from real training set and the fake data from the generator. The goal of discriminator, which is by nature a classifier is to accurately predict the data it receives is from real training set or from the generator.

When training a GAN, the goal is usually to obtain a generator that is eventually able to generate fake data that looks almost the same as real training data. The input of the generator is a standard Gaussian and the output of it is a sample from some distribution D on R^d, which is supposed be closer to some target distribution D_{real} during the training process. The training will use samples from D_{real} and the fake data from the generator together to train a discriminator that is trying to maximize its ability to distinguish the samples from D_{real} and D. As long as the discriminator is successful at making a distinction between these two kinds of samples, the output of it can be used as a feedback to the generator and thus improving its distribution D, making it one step closer to D_{real}.

The training is continued until the generator is able to generate perfect fake data which makes the discriminator has nothing better to do than random guessing when deciding the sample it receive comes from D or D_{real}. The discriminator can be discarded when the training is complete if generator is the only concern.

If the generator of the GAN developed as stated above is used for generating synthetic images, there is no way to manually specify the label of the generated data. The generator itself will decide which class the generated image should belong to. This weakness of GAN limits its usage in data augmentation.

As opposite to this kind of non-class-conditional GAN, a class-conditional GAN can generate images of manually specified labels. For example, if a class label such as "cat" is specified, the generator would then generate an image of cat. To achieve this, class information is needed when developing a class-conditional GAN. Fig 2.3 illustrates the difference between non-class conditional GAN and class-conditional GAN during training and generating fake images.

For class-conditional GAN, the input of the generator is a noise vector concatenated with a one-hot vector. The noise vector part allows the generator to generate diverse set of examples, but it should be the one within a certain class specified by the one-hot label part. This is what lets the user control the label of the generated images.

As for the input of the discriminator, the one-hot class information could also be fed in as additional channels apart from the RGB channels of the original image. These additional channels have the same size as that of the RGB channels. The values within these channels correspond to the one-hot label. For example, if the one-hot label is [0,0,1] then all the values of first and second channels are 0, whereas the values of the third channel are all 1. Therefore, to fool the discriminator, the generated images need to look like the examples from the class it claims.

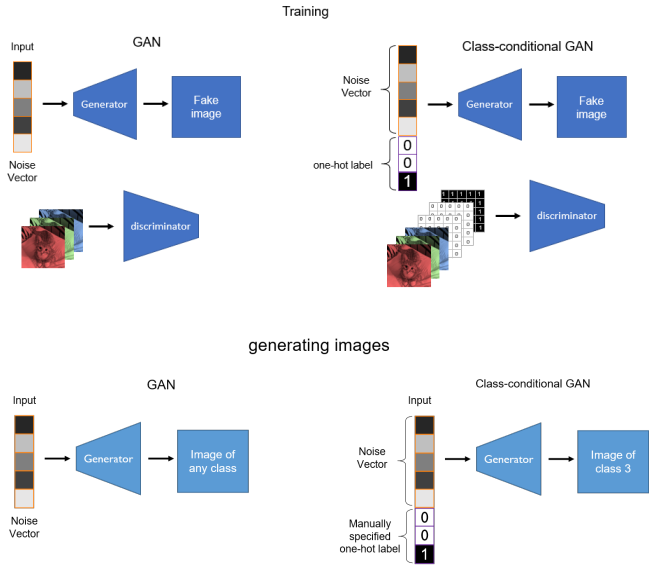


Fig 2.3 Difference between GAN and class-conditional GAN

2.3 Inception Score and Fréchet Inception Distance

Inception Score (IS) and Fréchet Inception Distance (FID) are two widely used methods for measuring the quality of generated image samples.

The inception score evaluates the GAN-generated images by applying a pre-trained Inception network to GAN-generated images and then comparing the conditional label distribution with the marginal label distribution.

$$IS = exp(E_{x \sim p_g} D_{KL}(p(y|x) \,||\, p(y)))$$

where p(y|x) is the conditional probability distribution outputted by the pre-trained Inception network given x, which is the GAN-generated samples. If the sample is of

good quality, p(y|x) should have low entropy. p(y) is the marginal probability computed as:

$$p(y) = \int_z p(y|x = G(z))dz$$

where G(·) is the generator of GAN, z is the noise vector. If the generated images are diverse, then p(y) should be uniform. A better GAN is supposed to achieve higher inception score.

In FID, the Inception network is used to extract features from the generated data. A multivariate Gaussian distribution is then used to model the real data distribution for the features which is computed as below:

$$FID(x, g) = \|\mu_x - \mu_g\|_2^2 + Tr(\sum_x + \sum_g - 2(\sum_x \sum_g)^{\frac{1}{2}})$$

where Tr is the trace of the matrix. A better GAN is supposed to achieve lower FID.

2.4 GANs for Data Augmentation

One of the downstream applications of GAN is data augmentation, but probably due to the low-quality of the generated samples, GAN has achieved limited success in this task. Recently, BigGAN, the state-of-art class-conditional GAN, can generate photorealistic images of ImageNet up to 512×512 resolution. More importantly, it also achieves the IS and FID similar to the dataset where it was trained. In other words, it seems that BigGAN has indeed learned the real data distribution of the dataset. If it was true, then one would expect to use BigGAN-generated images as a replacement of the original dataset since it is easier to store a GAN model on local disk and the GAN can provide unlimited amount of data of any class.

This paper designs experiments and tests the hypothesis that BigGANs can be used for data augmentation.

3. EXPERIMENT

This paper hypothesizes that if the state-of-art GAN really capture the data distribution from which it was developed, then a classifier trained on the generated data should be at least as good as the one trained on real data, no matter the structure of the classifier. To this end, DiffAugment-based BigGAN pre-trained on CIFAR-10 is used as the image data generator. DenseNet and DIANet, which are two classifiers with very different structure, are used as the baseline classifier.

3.1 BigGAN and DiffAugment-based BigGAN

As stated in section 2.4, BigGAN has been by far the state-of-art class-conditional GAN. As the name suggests, the BigGAN is focused on scaling up the GAN models which includes: 1) two or four times the number of parameters; 2) eight times the batch size; 3) two architectural changes; 4) modified regularization scheme. It turned out that GANs benefit dramatically from scaling.

DiffAugment-based BigGAN, on the other hand, is a variant of BigGAN. It modifies how BigGAN update its parameters and is trained on augmented data, which leads to even better performance than the original BigGAN.

In Fig 3.1, unlike traditional data augmentation where transformation is applied to only real samples form training set, the transformation was applied to both real samples (T(i)) and generated samples (T(ii)). When updating generator, gradients need to be back-propagated through T(iii), which requires T to be differentiable w.r.t. the input.

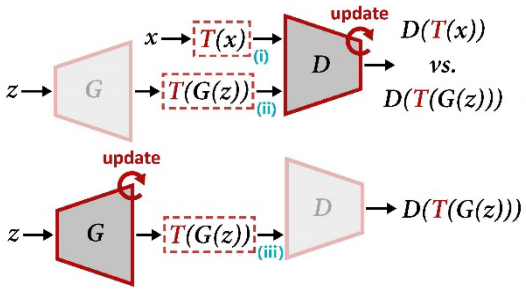


Fig 3.1 Overview of DiffAugment for updating discriminator (top) and generator (bottom).

The author proposed that the transformation for data augmentation should be differentiable so that generator can obtain useful gradient from discriminator. In terms of cost function, the same transformation is also added to generator's cost function and one should keep in mind that this transformation should be differentiable.

$$L_D = E_{x\sim p_{data}(x)}[f_D(-D(T(x)))] + E_{z\sim p_z(z)}[f_D(D(T(G(z))))]$$
$$L_G = E_{z\sim p_z(z)}[f_D(D(T(G(z))))]$$

The author then conducted extensive experiments to demonstrate the benefit of this method and concluded that it brings consistent improvement to a wide range of GAN with different architecture.

This paper uses a DiffAugment-based BigGAN pre-trained on 100% CIFAR-10 data to generate synthetic image data. The DiffAugment-based BigGAN achieved 8.70 on FID, which is better than 9.59 achieved by the original BigGAN.

Some sample images generated by the DiffAugment-based BigGAN is shown in Fig 3.2.

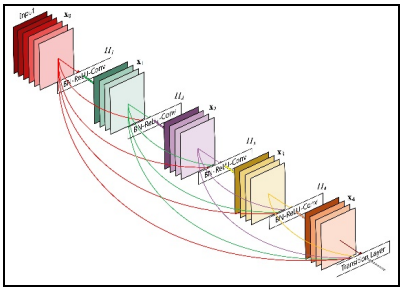
Fig 3.2 Sample images (automobile, bird, and horse) generated by DiffAugment-based BigGAN

3.2 DenseNet

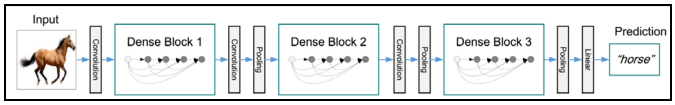
Traditionally, a hidden layer in the convolutional neural network is connected only to its preceding and succeeding layer. The output of one layer is the only input to the next layer. In DenseNet, however, each layer is directly connected to every other layer. In other words, the output of each layer is passed on as inputs to all succeeding layers and the input of each layer includes the output of all preceding layers.

The entire DenseNet is divided into several dense blocks (as in Fig 3.3 (a)) within which, layers are densely connected. There are transition layers (as in Fig. 3.3 (b)) between the dense blocks, which do convolution and pooling that decreases the dimension of the feature maps.

Since the layers are densely connected, the extracted features are fully reused across different layers so that the number of parameters needed can be substantially reduced. Furthermore, the depth of the whole network can also be reduced, which then alleviates the vanishing-gradient problem. The performance of DenseNet is shown in Table 3.1.



(a) A 5-layer dense block



(b) A deep DenseNet with three dense blocks

Fig 3.3 Overview of DenseNet

Table 3.1 Results of DenseNet on CIFAR-10

Model	# of Parameters	Error Rate (%)
DenseNet (L=40, k=12)	1.0M	7.00
DenseNet (L=100, k=12)	7.0M	5.77
DenseNet (L=100, k=24)	27.2M	5.83
DenseNet-BC (L=100, k=12)	0.8M	5.92
DenseNet-BC (L=250, k=24)	15.3M	5.19

* L means depth; k means growth rate.

The DenseNet with L=40 and k=12 is used in later experiments.

3.3 DIANet

DIANet incorporates LSTM in the Dense-and-Implicit-Attention (DIA) unit that allows sharing an attention module throughout different network layers, which encourages the integration of layer wise information and recurrently fuses the information from preceding layers to enhance the attention modeling at each layer.

The DIA unit can be used as a plugin of each layer such as Fig 3.4 left, and this is equivalent to add a global central DIA unit that connects every layer in the network. Each pair of the layers is thus implicitly connected through the implicit global DIA unit.

The performance of DIANet on CIFAR-10 is shown in Table 3.2

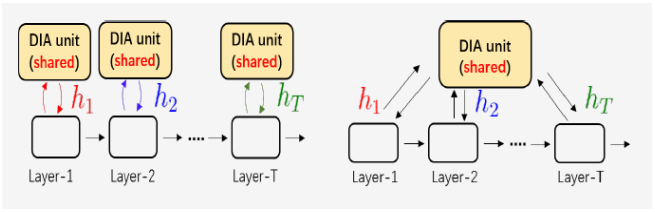


Fig 3.4 Left: explicit structure of DIANet. Right: implicit connection of DIA unit.

Table 3.2 Results of DIANet on CIFAR-10

	Original		DIANet (r=4)	
	#P	ER	#P	ER
ResNet164	1.70M	6.46	1.92M	5.42
PreResNet164	1.70M	4.99	1.94M	4.77
WRN52-4	12.05M	4.04	12.28M	3.83
ResNext101,8x32	32.09M	4.27	32.96M	3.76

"r" means reduction ratio of DIANet
#P means number of parameters
ER means error rate (%)

The DIANet with r=4 based on ResNet164 is used in later experiments.

3.4 Experiment Setup

In this paper, a DiffAugment-based BigGAN pre-trained on 100% CIFAR-10 data is used as data generator. The two baseline classifiers are DenseNet with L=40, k=12 and DIANet with r=4 based on ResNet164.

Two kinds of experiments are conducted. The first experiment is data replacement. The two baseline classifiers are trained on the CIFAR-10 dataset where 20%, 40%, 60%, 80% or 100% of the samples are replaced with the GAN-generated ones. The number of the samples of each class remain the same as that of the original dataset. So, after replacement, the dataset is still balanced as it was before. The original testing set is not changed and is used for evaluating the accuracy of the classifier. The second experiment is data augmentation. The original CIFAR-10 dataset is added with 20%, 40%, 60%, 80% or 100% more samples generated by GAN. The two baseline classifiers are then trained on the augmented dataset separately. The

augmented dataset is still balanced, and the testing set is still not changed and used for evaluation.

The hypothesis is that if the GAN really captures the real data distribution of CIFAR-10, the performance of the classifiers in the two experiments should be at least as good as that achieved on original dataset.

The configuration of the training is set to be the same as that in the paper where the classifier was proposed, which is listed as follow:

DenseNet

- Depth:40
- Growth rate:12
- Optimizer: stochastic gradient descent (SGD)
- Using batch size 64 for 300 epochs
- Initial learning rate is set to 0.1 and is divided by 10 at 50% and 75% at the total number of epochs
- Weight decay of 10^{-4} and a Nesterov momentum of 0.9 without dampening

DIANet

- Based on ResNet56
- Reduction ratio:4
- Optimizer: stochastic gradient descent (SGD)
- Using batch size 64 for 200 epochs
- Initial learning rate is set to 0.1 and is divided by 10 at 40%, 60% and 80% at the total number of epochs
- Weight decay of 10^{-4} and a Nesterov momentum of 0.9 without dampening

3.5 Data Replacement Experiment Results

Based on Fig 3.5, for both classifiers, the training error curves remain almost the same no matter the replacement ratio. However, the testing error curve moves up as the replacement ratio increases. This suggests that the classifiers are getting overfitting as more original samples are replaced. Therefore, the performance of the classifiers on the original testing set is expected to be slightly worse.

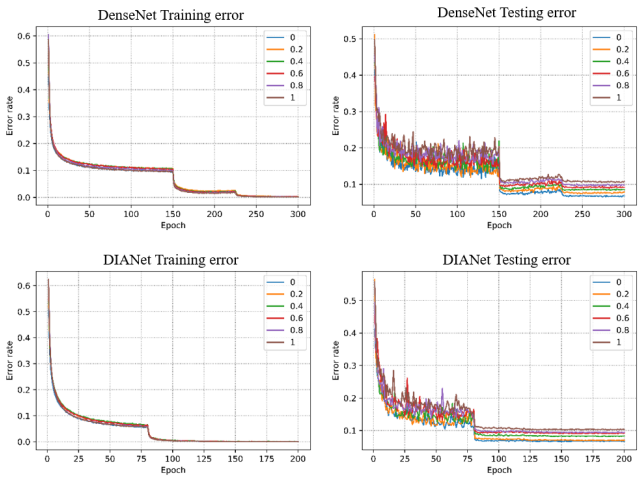


Fig 3.5 Learning curve of replacement experiment with different replacement ratio

In Fig 3.6, increasing replacement ratio results in systematically and consistently worse performance. When the replacement ratio grows from 0 to 1, top-1 error of DenseNet increases from 6.82% to 10.75% (increased by 57.62%), and the top-1 error of DIANet increases from 6.64% to 10.33% (increased by 55.57%). To further explore what happened to the classifier, the performance of the classifier on each class is shown in Fig 3.7.

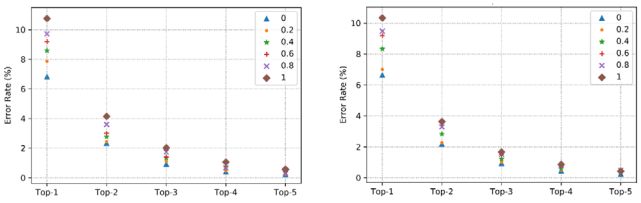


Fig 3.6 Performance of DenseNet and DIANet trained on CIFAR-10 with different replacement ratio (left: DenseNet; right: DIANet)

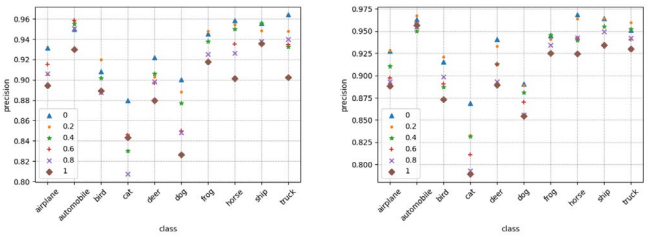


Fig 3.7 Performance of DenseNet and DIANet on each class in replacement experiment (left: DenseNet; right: DIANet)

In Fig 3.7, for both classifiers, most classes suffer a drop in performance compared to the original dataset, that is, when replacement ratio is 0. A few classes (such as automobile and frog) get improved but only marginally.

3.6 Data Augmentation Experiment Result

Given the performance of the classifiers in replacement experiments, one may not expect improved performance by augmenting CIFAR-10 dataset with GAN-generated samples.

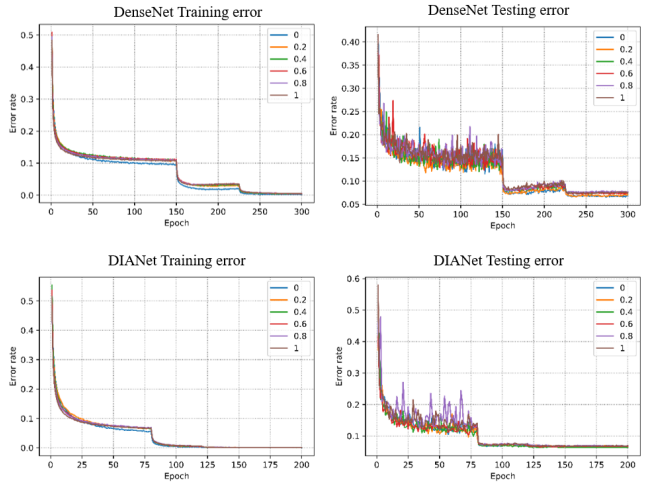


Fig 3.8 Learning curve of replacement experiment with different replacement ratio

In Fig 3.8, for DenseNet, the training error curve of every replacement ratio is almost the same, but the testing error curve moves up as the replacement ratio increases. Given the training history as this, the classification performance of DenseNet may not be better by augmenting available data with synthetic data.

As for DIANet, however, testing error curve dose not move up as DenseNet did, suggesting that the model may slightly benefit from augmented training data.

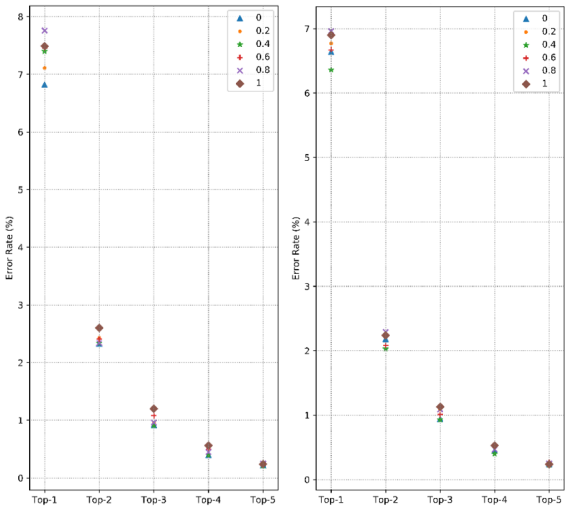


Fig 3.9 Performance of DenseNet and DIANet trained on CIFAR-10 with different augmentation ratio (left: DenseNet; right: DIANet)

In Fig 3.9, for DenseNet, the performance is always worse compared with the original training set, which is the same case as in replacement experiment. Perhaps surprisingly, for DIANet, when the augmentation ratio is 0.4, the Top-1 error is 6.36%, which is slightly better than 6.64% achieved on original dataset. The error rate is decreased by 4%.

The performance of the two classifiers on each class is shown in Fig 3.10

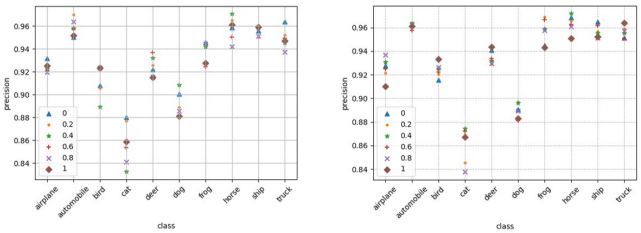


Fig 3.10 Performance of DenseNet and DIANet on each class in augmentation experiment (left: DenseNet; right: DIANet)

In fig 3.10, training either of the models on augmented training data does not guarantee consistent improved or worse performance on each class. For DensetNet, compared with the performance of original dataset, airplane, cat and truck is deteriorated but automobile is improved. For other classes, performance of training on original dataset is in the middle. There is no systematic improvement or deterioration over all the class as in the replacement experiment. For DIANet, when the augmentation ratio is 0.4, the performance is even better than the original dataset for most classes, which makes the overall performance slightly better than that without augmentation.

4. ANALYSIS

Based on replacement experiment, both two classifiers suffer from replacing original data with synthetic data. The classifiers are getting more overfitting as the ratio of synthetic data increases. There is systematic drop in performance on every class, which leads to the worse overall performance. The source of overfitting is that the GAN-generated data is not diverse and representative enough so that the classifiers simply memorize those data and cannot generalize to real data.

The synthetic data is generated by sampling a noise vector form Gaussian distribution, appending the one-hot label to it, and passing it to the generator. The noise vector is randomly generated, in other words, there is no control over the generation of the noise vector, which is probably the reason why the generated data is not diverse. Therefore, GANs need to be guided on how to generate diverse data, which may be achieved by adding additional control schema over the generation of noise vector.

Based on augmentation experiment, adding extra GAN-generated data to original dataset would not necessarily results in classifier being overfitting as in replacement

experiment. For example, there is no traits of overfitting in the training history of DIANet. Unlike replacement experiment, there is no systematic improvement or deterioration to the performance on every class.

Since we know that GAN-generated data could be lack of diversity based on analysis for replacement experiment, we may infer that three is limited useful features that help with classifying contained in the synthetic data. If those features are already included in the original dataset, then it makes sense that training the classifier on the augmented data brings no benefit. If those features are contradictory to those learned from original data, then the classifier may perform better on some classes while worse on the others, which is the case in the augmentation experiment.

5. CONCLUSTION

In this paper, the application of GAN on data augmentation is explored. The state-of-art class-conditional GAN is used to generate synthetic image data. DenseNet and DIANet are the two baseline classifiers used for comparison.

This paper demonstrates that the classifier naively trained on GAN-generated samples may not be as good as that trained on real images. The key reason is that GAN-generated images are lack of diversity so that the classifier cannot generalize well to real data.

Although a negative result, this paper suggests that to utilize GAN for downstream application such as data augmentation for classification, a "diversity control mechanism" should be carefully designed and incorporated when generating synthetic images to prevent classifier from overfitting.

6. REFERENCES

[1] Brock, A., Donahue, J. and Simonyan, K., 2018. Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096.

[2] Zhao, S., Liu, Z., Lin, J., Zhu, J.Y. and Han, S., 2020. Differentiable augmentation for data-efficient gan training. Advances in Neural Information Processing Systems, 33.

[3] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., 2017. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp.4700-4708).

[4] Huang, Z., Liang, S., Liang, M. and Yang, H., 2020. DIANet: Dense-and-Implicit Attention Network. In AAAI (pp. 4206-4214).

[5] Tanaka, F.H.K.D.S. and Aranha, C., 2019. Data augmentation using GANs. arXiv preprint arXiv:1904.09135.

[6] Dat, P.T., Dutt, A., Pellerin, D. and Quénot, G., 2019, September. Classifier Training from a Generative Model. In 2019 International Conference on Content-Based Multimedia Indexing (CBMI) (pp. 1-6). IEEE.

[7] Ravuri, S. and Vinyals, O., 2019. Classification accuracy score for conditional generative models. In Advances in Neural Information Processing Systems (pp. 12268-12279).

[8] Ravuri, S. and Vinyals, O., 2019. Seeing is not necessarily believing: Limitations of biggans for data augmentation