Bayesian Data Augmentation

Pierre Le Jeune - s182169@student.dtu.dk

1 Introduction

For years, deep learning methods have proved to be really effective. Major breakthroughs have been made thanks to those kind of models in the past few years, in computer vision, medicine, robotic, etc. Yet, to reach such performances, deep learning requires a lot of data to train on. Data augmentation techniques focus on generating new samples from datasets in order to create more training examples and improve performances of already existing models. In this work, our goal was first to reproduce the work from [1], that uses generative adversarial networks to create new training examples during the training. Then we compare the performances with state-of-the-art augmentation techniques introduced in [2]. Finally, we try to combine those techniques in different ways.

2 Background

2.1 Data augmentation

Data augmentation is an usual technique used in Deep Learning to improve performances of various models. The main idea of Data Augmentation is to generate data points similar to training examples. Most of the time, this is done by finding some reasonable transformations, i.e. transformations which modify data without changing the label. For images, transformations can be random rotations, translations, scaling but it can also be more complex such as the ones describe in [2]. These transformations are represented by a restricted class (Continuous and Piecewise-Affine - CPA) of diffeomorphisms over the training manifold. With this restriction, the transformations are reasonable and tractable, which are fundamental properties to build real-world applications. Figure 1 shows examples of the effects of such transformations on MNIST images. An other benefit of these transformations is that their parametrizations can be learned whereas most of data augmentation techniques require manual parametrization that are time-consuming and seldom optimal.

2.2 Generative adversarial networks

Generative adversarial networks are a class of generative models introduced by Goodfellow in [3]. Since their introduction in 2014, GAN have proved their effectiveness to generate high quality samples (see Figure 2). The main idea of these models is to train two separate networks, a generator and a discriminator at the same time. The generator tries to produce

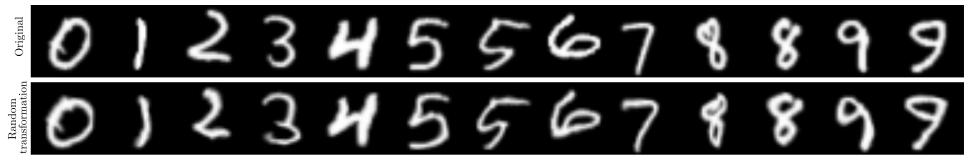

Figure 1: Example of transformations randomly sampled after training. Top row contains images from MNIST dataset and bottom row contains the same images transformed. Image from [2].

Figure 2: Example of high quality images created by Style-GAN [5].

data similar to training examples, while the discriminator tries to distinguish between real and fake samples. To do so, the generator and discriminator are trained to minimize their own loss function:

$$\mathcal{L}^D(\theta^D, \theta^G) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z))) \tag{1}$$

$$\mathcal{L}^G(\theta^D, \theta^G) = -\mathcal{L}^D(\theta^D, \theta^G) \tag{2}$$

Where D and G are functions representing respectively the discriminator and the generator with their own parameters θ^D and θ^G. The variable z is a latent variable sampled from a normal distribution: $z \sim \mathcal{N}(0, I)$.

This sets up a minimax zero-sum game between the generator and the discriminator where the Nash-equilibrium is achieved when $G(z) = p_{data}$, i.e. when the generator has perfectly learned the distribution of the training data. However, in practice this loss does not perform well and another loss is used for the generator in order to improve training.

$$\mathcal{L}^G(\theta^D, \theta^G) = -\frac{1}{2}\mathbb{E}_z \log D(G(z)) \tag{3}$$

A more detailed analysis of the difference between GAN losses can be found in [4].

2.3 Bayesian data augmentation using GANs

As mentioned before, GAN and data augmentation share a common goal: create samples from the training distribution. This is why GAN can be very helpful to enrich datasets. Previous works have already done it in several ways [1, 6, 7] and have shown significant performance improvements compared to classical data augmentation techniques. In our work, we will focus on the method from [1] where a GAN and a classifier are trained at the same time. The GAN is still composed of a generator and a discriminator trained to minimize losses described respectively by equation (3) and (1). The classifier part is then trained with the following loss:

$$\mathcal{L}^C(\theta^D, \theta^G, \theta^C) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log C(x) - \frac{1}{2}\mathbb{E}_z \log C(z) \tag{4}$$

3 Experiments

The main experiments conducted in this work have been focused on comparing the two data augmentation techniques described before: the AlignMNIST dataset from [2] and Bayesian data augmentation using GANs from [1]. Then we tried to combine those techniques in two different ways. First by making GAN learn the transformations and then create fake images from the real distribution by applying the learned transformations. Then, in a simpler manner, by training the GAN from section 2.3 on AlginMNIST dataset.

3.1 Comparison between Bayesian and learned data augmentation

Most of this work was spent on the comparison between two existing data augmentation techniques described above. In order to make fair comparisons, we used the same classifier for both methods using the same parameters. The classifier was a 3-layers MLP (with 2048 units per layer). Only the GAN architecture and parameters have been optimized. This is just because the model used to create AlignMNIST dataset has also been optimized to reach the best performances. All the results from those comparisons are listed in the table 3.1. In this table we compare, a vanilla MLP classifier trained on MNIST, the same classifier trained on both AlignMNIST and AlignMNIST500 defined in [2] and the data augmentation GAN (DAGAN).

Augmentation technique	None	AlignMNIST	AlignMNIST500	DAGAN
Test accuracy	97.14%	97.97%	97.09%	**98.27%**
Number of epochs	100	2	1	50
Number of updates	39100	71000	76000	23500
Training time	0.3h	0.5h	0.5h	1.5h

Table 1: Test accuracy score of different augmentation techniques applied on the same 3-layers MLP. Models have been trained until reaching convergence, this is why the number of epoch and updates changes.

From these results, it can be seen that Bayesian data augmentation seems to be performing a bit better than the AlignMNIST dataset. However the results found with AlignMNIST are way bellow the one described in [2], where the same MLP achieved a 99.42% test accuracy. This might be because we did not optimize the parameters of the classifier much for these experiments, but this choice was made to ensure a fair comparison between different techniques.

3.2 Pedagogical GAN

Even though the results of DAGAN are quite good, the image generated (see Figure 3) are not of good-quality. One way to understand that phenomenon is that the GAN does not try to generate nice images but instead tries to generate images that will be challenging for the classifier. It can be seen as a teacher providing challenging and pedagogical exercises to his students to make them progress.

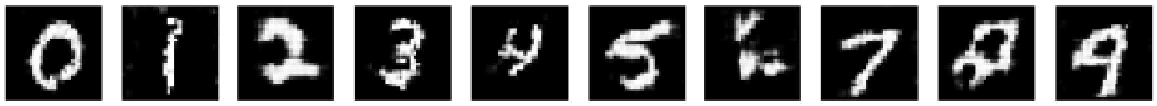

Figure 3: Example of images generated by the Data Augmentation GAN.

One way to make sure that the generator will produce such challenging examples for the classifier is to add a term to the generator loss that is low when the classifier is wrong and high otherwise. One way to do so it to re-write (3) as follows:

$$\mathcal{L}^G(\theta^D, \theta^G, \theta^C) = -\frac{1}{2}\mathbb{E}_{x\sim p(z)} \log D(G(z)) - \frac{1}{2}\mathbb{E}_z \log(1 - C(G(z))) \qquad (5)$$

Using this loss for the generator does not affect much results but decreases a lot the quality of generated images (see Figure 4 and Table 3.2). The fact that the network is still learning well from those examples is surprising and should be further investigated. The size of the discriminator and generator seems to have a great impact on the quality of the generated samples, the bigger the networks were, the better the samples looked. However, increasing the capacity of the GAN was not synonym of better performances.

Augmentation technique	None	DAGAN	PGAN
Test accuracy	97.14%	98.27%	**98.39%**
Number of epochs	100	50	50
Number of updates	39100	23500	23500
Training time	0.3h	1.5h	1.5h

Table 2: Test accuracy comparison between Data Augmentation GAN (DAGAN) and Pedagogical GAN (PGAN).

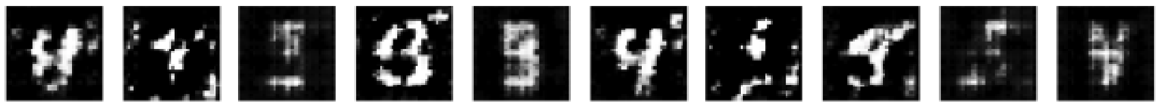

Figure 4: Images generated by the Pedagogical GAN.

3.3 Learning Bayesian transformations

After the comparison detailled above, we tried to implement a new method combining Bayesian data augmentation and learnable transformations. The idea here is to have a GAN that learns to generate the parameters of the transformations. Then for a batch of images, the generator creates fake images by applying the generated transformations to the current batch of images. The discriminator still tries to distinguish between real and fake images while the classifier tries to predict the classes of all images. Thereby, the losses from (1), (3) and (4) can still be used as before. We will refer to this architecture as Data Augmentation Transformation GAN (DATGAN).
This method was hard to train and results have not been consistent over different runs. Though, some runs have shown promising results with higher performances than the vanilla MLP. But it remains quite far from other data augmentation techniques in term of performances while being harder to train and taking way more time to reach convergence. The learned transformations seems to be reasonable, but looks suspiciously close to identity (see Figure 5). It might certainly be mode collapse of the generator, obviously learning identity is the best way to fool the discriminator. This explains why the accuracy results do not show much improvement compared to vanilla MNIST (see Table 3.3).

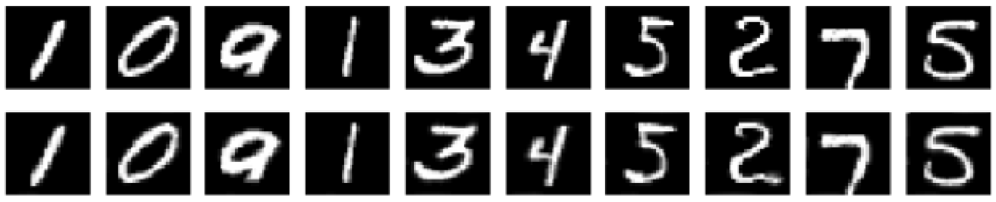

Figure 5: Transformed images with learned transformation with DATGAN

Figure 6: Example of transformation generated by DATGAN. First rows contains images from MNIST and the second one contains the corresponding transformed images using generated transformations.

Augmentation technique	None	AlignMNIST	DAGAN	DATGAN
Test accuracy	97.14%	97.97%	98.27%	97.46%
Number of epochs	100	2	50	50
Number of updates	39100	71000	23500	23500
Training time	0.3h	0.5h	1.5h	2.3h

Table 3: Result of experiment conducted on DATGAN architecture.

3.4 Augmented dataset and Bayesian data augmentation

This last experiment was pretty simple compared to the first two. The idea was to replace the MNIST dataset by the AlignMNIST and train the DAGAN architecture on it. This allows to get benefits from both data augmentation techniques and the results obtained are higher than in any of the previous experiments. Combining these techniques is computationally expensive, it took way longer to reach convergence. Nevertheless, that seems worthwhile as the accuracy gain was almost the sum of the gains of each separated techniques. That means that these techniques are not mutually exclusive and can work together. The results of this experiment can be found in table 3.4. As before the table contains as well some results of the previous experiments for the sake of comparison.

Augmentation technique	None	AlignMNIST	DAGAN	DAGAN + AlignMNIST
Test accuracy	97.14%	97.97%	98.27%	**99.23%**
Number of epochs	100	2	50	2
Number of updates	39100	71000	23500	71000
Training time	0.3h	0.5h	1.5h	2.5h

Table 4: Results of combining DAGAN and AlignMNIST dataset.

4 Discussion

The data augmentation techniques compared above have all their pros and cons. Data augmentation GAN seems to reach higher performances but the training is 3 times longer than training on AlignMNIST. Of course, AlignMNIST dataset must have taken some time to be generated, but once it is done training is fast while DAGAN always needs a lot of time to train. However, if time is not the limiting factor but memory is, DAGAN might be a good alternative as it does not require to store huge dataset in memory as AlignMNIST does. Furthermore, the technique used in [1] can be applied to any kind of data while AlignMNIST can only be done on images. Therefore, the Pedagogical GAN extension quickly tackled in this work (section 3.2) is also applicable to and should be investigated further.
Also one important flaw of data augmentation techniques using GAN is that it might be quite difficult to make the GAN converging without falling in mode collapse (i.e. the generator finds a local optimum to its loss by always generating the same example to fool the discriminator). This is a common problem that have been addressed in many ways in [8, 9, 10]. Even if none of those techniques have been recognized as a full solution to address mode collapse, it might be interesting to improve the DAGAN architecture based on these works.

5 Conclusion

To conclude, this work has made a fair comparison between different data augmentation techniques and have highlighted their pros and cons. Bayesian data augmentation can perform as well as state-of-the-art augmented datasets on MNIST and still has room for improvements. Finally, we have proved that these techniques are not mutually exclusive and can work together in order to achieve higher performances rather easily.

References

[1] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. In *Advances in Neural Information Processing Systems*, pages 2797–2806, 2017.

[2] Søren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John Fisher, and Lars Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Artificial Intelligence and Statistics*, pages 342–350, 2016.

[3] 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*, pages 2672–2680, 2014.

[4] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[6] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[7] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[9] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[10] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.