Data Augmentation using Deep Generative Adversarial Networks and Adversarial Auto-encoders

Jarvin Mutatiina (S3555631) Parth Tiwary(S3576434)
Sohyung Kim(S3475743) Thijs Eker(S2576597)

February 15, 2019

Abstract

In this paper we compare classification accuracy between convolutional neural networks(AlexNets) trained on a small dataset and a dataset that is augmented with synthesized samples. We augment 64 by 64 resolution images for the Swedish leaf dataset (for each class) using an Auxiliary Classifier Generative Adversarial Network(ACGAN) and Wasserstein Autoencoders(WAEs). We train two AlexNets in parallel, for original and augmented dataset. Mode collapse conditions are also kept in check for each class and in case of mode collapse, the collapsed class is replaced with the original dataset. We propose a comparison of validation accuracy averaged over eight runs for four training set sizes, both for AlexNet trained on original dataset and augmented dataset. Furthermore we perform a t-test on the validation accuracy: we conclude that increase in accuracy is much more significant in case of smaller training dataset when compared to the larger training dataset. In short, augmentation is more effective on smaller datasets, but even on big enough training sets we still see some performance increase.

1 Introduction

Data is the key to all state-of-the-art deep learning implementations. More variation in the data during the training of the model usually results in models which generalize better. However, we do not always have enough data to incorporate these variations.We could try to address the scarcity of data using traditional data augmentation techniques e.g. rotation, shear, translation etc. These techniques can be effective and are also widely used. However, there is a conceptual limitation on the number of images that can be generated using these techniques, which consequently imposes an upper bound on the variations we can include in a dataset or number of augmentations generated.

An alternative solution to this problem is generative modelling. In generative modelling, we attempt to learn a probabilistic distribution in order to generate images which are similar to the existing dataset. We implicitly start with an assumption that our dataset X is generated from a probability distribution $P_{gt}(X)$. Our objective would be to learn a probability distribution P which we can sample from, such that the underlying constraint is to make P maximally similar to $P_{gt}(X)$. There are obvious concerns about assumptions imposed and their influence on samples generated.

In this paper, we compare results from two such state-of-the-art frameworks in generative modelling: Auxiliary Classifier Generative Adversarial Network(ACGAN) [5] and Wasserstein Auto-encoders[10]. We use Swedish Leaf Dataset for our analysis. For the specifics of the implementation, we generate(augment) images per class for the 15 classes contained in the leaf dataset. Two AlexNets[3] are trained in parallel, one on the original dataset and one on the augmented dataset, which is followed by an exhaustive comparison and *t-test* for gauging the significance of the results obtained. In the following sections, we discuss methodology used to various aspects of the research i.e. ACGAN,WAE and

AlexNet classifier. We also explain the experimental setup and finally discuss the results while exploring the drawbacks and alternatives for our approach.

2 Methodology

This section explores the methodology and various architectures used in this research.

2.1 The Swedish leaf dataset

The Swedish leaf dataset[9] consists of 1,125 plant leaves images of 15 different Swedish tree species(classes), with 75 images per class. The images vary in size and resolution. The images are down-scaled to a 64 by 64 size due to computational constraints. The low resolution of images however, diminishes some of the distinguishing features that would otherwise be recognizable on the larger sized images. This is evident in the image comparison shown in Figures 1 and 2, with class 7 of the leaves changing to a drastically more round shape after rescaling.

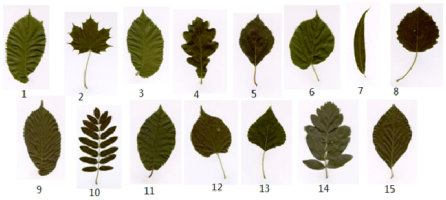

Figure 1: The original Swedish dataset

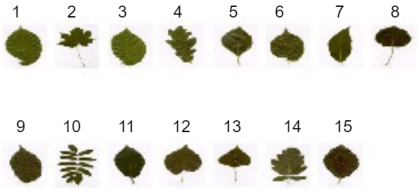

Figure 2: The scale downed Swedish dataset

2.1.1 Preliminary standard data augmentation

The following preliminary standard/traditional augmentation methods [6] were employed; rotation- randomly rotate images, width shift- randomly shift images horizontally, height shift - randomly shift images vertically, channel shift - slightly change color scale, zoom- randomly zoom into the images plus horizontal flip which mirrors the images. These techniques ensure a large enough preliminary dataset to be used for the various architectures. The augmentation is also performed on the fly i.e. before training of the various model which enforces randomness across the different training cycles.

2.2 Machine learning pipeline / architecture

The main components of the machine learning pipeline used in this research are; Auxiliary Classifier generative adversarial network(ACGAN), Wasserstein Auto-encoders(WAE) and a convolutional neural network - AlexNet. These are all individually standing architectures that are interconnected into a pipeline, as shown in the figure 3, which generates results that are the point of comparison for this research.
ACGAN and WAE are separately trained to generate new sets of data that are twice as large as the original training set. Consecutively, two convolutional neural networks(AlexNet) are trained with original training samples and mixed samples (original and generated samples). Finally, the trained models are tested on the validation set and their validation accuracies are compared to examine a significant difference.

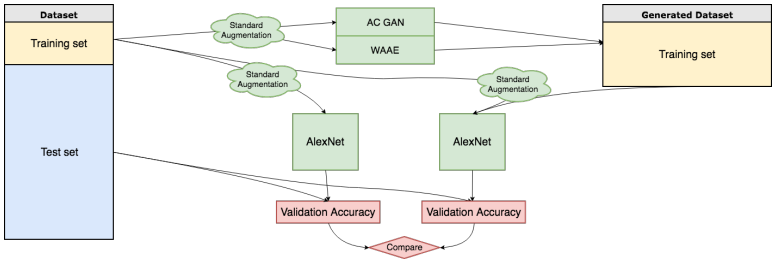

Figure 3: Machine learning pipeline/architecture

2.3 Auxiliary Classifier generative adversarial network(ACGAN)

For our research, we used different ways of synthesizing new samples to augment the original dataset. One option we fully explored was training an ACGAN and using the trained generator to create new samples.

2.3.1 Architecture

Figure 3 gives a quick overview of how an ACGAN works. There are two important parts that make up the ACGAN, one is the generator and the other the discriminator. The generator is a network that gets as input the latent vector Z consisting of 100 random values between 0 and 1 multiplied with a class label. This is passed through 5 deconvolutional layers to create an image of the size 64*64*3. The discriminators input comes from two sources, it can be a real sample from the training set X or a picture that was synthesized by the generator. The task of the discriminator is to classify the picture as real or fake and as one of the 15 classes.

2.3.2 Hyperparameters

For the ACGAN we used the following parameters: Both the generator and discriminator are trained with an Adam optimizer with a learning rate of 0.0002. The generator uses 5 deconvolutional layers that all use batch normalization and a ReLU activation function(except the last layer which is tanh). Most of the settings were taken from a paper describing guidelines for training deep convolutional GANs[7], some of them required some extra tuning. The discriminator uses 4 convolutional layers with dropout(25%), leaky ReLU($\alpha = 0.2$) and batch normalization. The last layer is a fully connected dense layer with a sigmoid activation function. The network was trained for 50000 epochs, using a batch size of 128.

2.3.3 Collapsed classes

When training the ACGAN, it is common for 1 or 2 classes to collapse, meaning it will generate the same picture for every latent input. These collapsed classes make the augmented training set unbalanced and reduce performance. Therefore we chose to automatically detect collapsed classes and not use them to augment the training set. To detect collapsed classes we generate 50 images for every class using the trained generator. We compare the images with each other using a simple pixel based euclidean distance. When the average distance of pixels is lower than 0.05, we conclude a class is collapsed(pixels range from -1 to 1) and the generator will not be used to synthesize samples for this class(real training samples are duplicated instead). Figure 5 shows images synthesized with a generator which collapsed for class 1, Therefore the output is always the same for this class. Figure 6 shows the average inner class differences over time while training the same GAN.

2.4 Wasserstein Auto-encoders(WAE)

Auto-encoders are much more stable to train than generative adversarial networks, however, in terms of quality reconstruction on large compression ratio, GANs tend to outperform autoencoders[11]. This leaves a tradeoff between training stability and quality of reconstruction. For this project, we use a

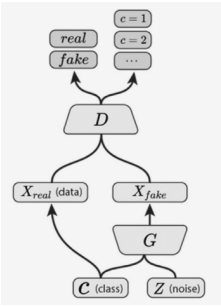

Figure 4: Auxiliary Classifier generative adversarial network(ACGAN) architecture[1]

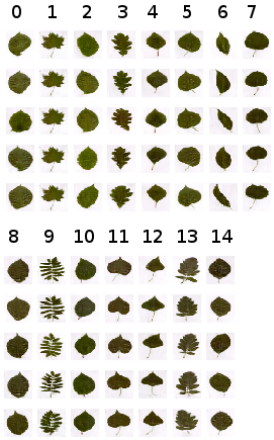

Figure 5: Synthesized leaf images, class 1 is collapsed

Figure 6: Plot of the average euclidean pixel distance over time for the ACGAN that produced the images in Figure 5

Wasserstein Auto-encoder(WAEs)[10] as an alternative for ACGAN.

We train WAE per class for all the 15 classes of the dataset for 200 epochs. For the specifics of the architecture, encoder: four 2D convolutions each followed by a leaky ReLUs and batch normalization which is followed by another 2D convolution, decoder: four 2D deconvolutions each followed by ReLUs and batch normalization, which is followed by another 2D deconvolution and a sigmoid function, discriminator: two linear layer followed by ReLU and a sigmoid function respectively. Encoder and decoder architecture is scaled to accept and reconstruct images of resolution 64x64x3. Samples generated per class from implementation of WAE can be seen in figure 7

It should be noted that because of the time constraint, integration of this implementation of *WAE* into our overall pipeline could not be accomplished and hence some of the crucial results for an exhaustive comparison are still work in progress.

Figure 7: Images generated per class using *wasserstein autoencoders*

2.5 Convolutional Neural Network (CNN) - AlexNet

The CNN architecture(as shown in figure 8) is solely based on the original AlexNet[4] with 5 convolutional layers, 3 dense or fully connected layers, Rectified Linear Units(ReLU) activation, Batch normalization, Drop out, max pooling and Adam optimizer. These features of the architecture contribute immersely to the successful learning of the Neural network as described below;

ReLU Nonlinear activation - the non-linearity accelerates the training of the network as compared to other activation functions like sigmoid; whose derivative becomes very small in the saturating region and hence no updates to the weights.
Max Pooling - The first two and last convolutional layers are followed with overlapping max pool layers. As explored by Krizhevsky et al[4],overlapping makes it difficult to overfit,as compared to non-overlapping scheme.

Batch normalization - which happens after every layer, reduces the saturating nonlinearities with internal covariate shift by fixing the mean and variance of the layer input[2].This reduces the dependence of the gradient on the scale of the parameter and also in turn accelerates the training of the AlexNet.
Drop out - Using the drop out layer after every dense layer, randomly switches out the activation with probability 0.4. The drop our results in different architectures and leads to prediction being averaged over these ensembles of models.
Adam Optimizer - combines the two re-known learning optimizers; Root Mean Square Propagation(RMSprop) and Adaptive Gradient(AdaGrad) to improve its effectiveness. Adam optimizer computes the adaptive learning rates for each parameter and also maintains an exponentially decaying average of past gradients[8]. A combination of the default parameters of the Adam Optimizer and a learning rate of 0.00001 were used for our experiments.
In addition to the hyper-parameter tuning, the input images to the CNN were varied by traditionally augmenting the data as described in section 2.1.1.

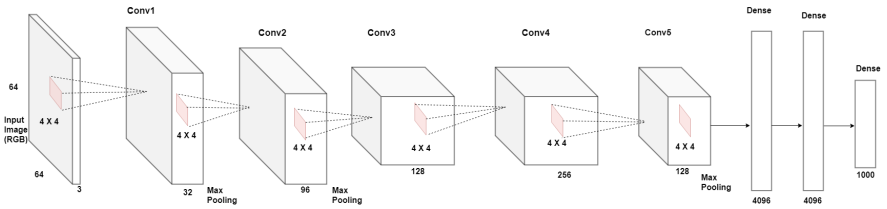

Figure 8: AlexNet architecture

3 Experiments

Experiment are ran for four different sized training sets, since the classification task is not a hard one we also opted for some very small datasets. We did this because, for augmentation to have a positive effect, the training set should be a bit too small to get optimal results. The different sizes of training datasets are shown in table 1: the percentages show what part of the dataset is used for the training set, the remaining images are used for validation.

Split	Original training dataset	Augmented training dataset
0.05	3 images per class	3 images per class + 6 generated pictures
0.1	7 images per class	7 images per class + 14 generated pictures
0.2	15 images per class	15 images per class + 30 generated pictures
0.8	60 images per class	60 images per class + 120 generated pictures

Table 1: Splits for training dataset

During training, the images are traditionally augmented on the fly as explained in section 2.1.1. The various architectures of this research are implemented using python and the Tensorflow(with Keras) and Pytorch libraries. The hyperparameters for the different models can be found in there respective sections in the methodology. For every split, we run the pipeline 8 times and the results are an average of these runs. For every new run, the dataset is again randomly split into a training and validation set.

4 Results

For the result of our experiments, we measure validation accuracy of original and augmented dataset over eight runs in four different percentage splits of dataset. The accuracy metrics obtained for all 8 runs are averaged out for every split. Then we use a two tailed paired t-test to see whether one of the datasets performs significantly better.

In order to see the difference between original and augmented dataset for every split, we plot the averaged 8 runs validation accuracy with both datasets. Figure 9 shows the metrics on the validation sets of

original and augmented datasets, which are dotted and straight lines respectively. Looking at the zoomed figure 10 of the same result for more detail, the differences between original and generated datasets can be seen clearly. For all four splits, the augmented dataset always has considerably better performance than original one. When observing the p-values, we can see that this is not always significant; only for the smallest dataset, augmentation significantly increases validation accuracy.

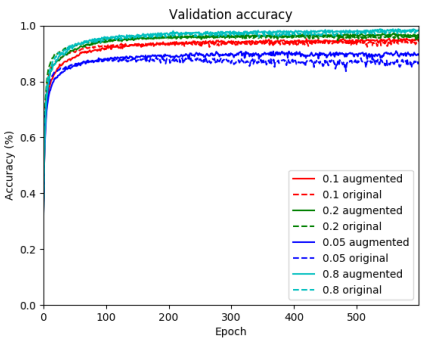

Figure 9: Averaged validation accuracy in each splits of datasets

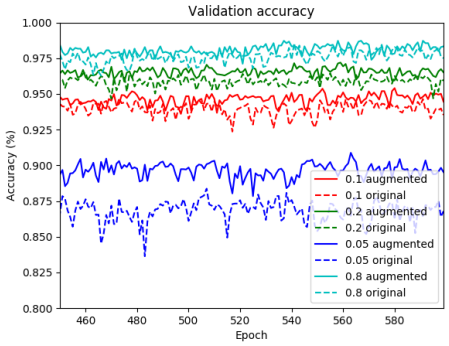

Figure 10: Zoomed version of averaged validation accuracy in each splits of datasets

Table 2 presents the detailed numbers of averaged classification validation accuracies, differences between original dataset and augmented dataset and p-value. All validation accuracies mentioned in this table are taken at the last epoch. Therefore, the averaged classification accuracies does not present the best performance for each split. For instance, the 5% split(original) has its best classification accuracy around 150 epochs and after that, the validation accuracy gets worse because of overfitting.

Split	Augmented dataset	Original dataset	Difference	p-value
0.05	89.532.14	86.792.71	2.731.45	0.00160
0.1	94.460.93	93.532.09	0.931.86	0.22629
0.2	96.511.07	95.930.81	0.581.12	0.21030
0.8	98.330.76	97.501.55	0.831.29	0.12991

Table 2: Average results for augmentation with ACGAN

Generally, the validation metrics of both original and augmented datasets are better when the training datasets are bigger. However, in all experiments, the augmented datasets have higher validation accuracy than original datasets. Furthermore, the differences in validation accuracies are considerably better for very small training sets such as 5% of dataset.

5 Discussion and conclusion

This work introduced a comparison between classification accuracy of the augmented and original dataset. For augmentations, we primarily deploy image synthesis and use two state-of-the-art *generative* models, namely, ACGAN and WAE.To assess the effectiveness of the augmentation we compared validation accuracies from two AlexNets, one trained on original dataset and the other trained on augmented dataset. For all splits(sizes of training sets) the average validation accuracy was higher for classifiers trained on augmented dataset. After comparing the validation accuracy with a pairwise t-test it was shown that this increase in accuracy was only significant for the smallest training set(the 5% split).

Several directions exist for improving and building upon this work. We will try and put up a discussion on each of them categorically:

Training - model collapse: Model collapse is unavoidable occurrence in training GANs but identification of the occurrence is a more crucial point of research. Currently, we are identifying model collapse

using *euclidean similarity* and the collapsed class is replaced with same class from original dataset. This can be a point of improvement with more efficient ways of dealing with model collapse configurations. Another interesting point of extension can be to include a measure for assessing the diversity and variability in the generated images using a diversity measure.[5]

Quality - training stability tradeoff: We decided to use wasserstein autoencoders as a more stable alternative for ACGAN, which proved to be more stable as the concept of model collapse was done away with and training was much stable. Reconstruction quality from *WAE* is much better when compared to closely related variants, for example, variational autoencoderswhich tend to generate blurry images. However, training WAEs per class is not very convenient. We train one WAE per class and generated images from the trained model for each class. An improvement could be autoencoder that can conditionally generate images per class.

Scaling the architecture: Scaling the architecture for images of higher resolution,e.g. *(128x128x3) or (256x256x3)*, which might significantly contribute towards improvement of final accuracy of the classifier. However, training a relatively larger architecture means more instability when it comes to training a GAN.

Different dataset: Another point of extension can be to use this implementation on a relatively difficult task/dataset, for example *agriplant* dataset.

References

[1] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *arXiv preprint arXiv:1803.01229*, 2018.

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Nips proceedings. *Learning Deep Features for Scene Recognition using Places Database*, Jan 1970.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] Odena, Augustus, Olah, Christopher, and Jonathon. Conditional image synthesis with auxiliary classifier gans. *[astro-ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field*, Jul 2017.

[6] Pawara Pornntiwa, Okafor Emmanuel, Schomaker Lambert, and Wiering Marco. Data augmentation for plant classification.

[7] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[8] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[9] Oskar Söderkvist. Computer vision classification of leaves from swedish trees, 2001.

[10] Tolstikhin, Ilya, Bousquet, Olivier, Gelly, Sylvain, Schoelkopf, and Bernhard. Wasserstein autoencoders. *[astro-ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field*, Mar 2018.

[11] Masaru Takeuchi Zhengxue Cheng, Heming Sun and Jiro Katto. Performance comparison of convolutional autoencoders, generative adversarial networks and super-resolution for image compression. *arXiv:1807.00270v1 [eess.IV] 1 Jul 2018*, Jul 2018.