Generative Adversarial Image Refinement for Handwriting Recognition

Deepak Dilipkumar
Advisor: Barnabás Póczos

26th November, 2017

1 Structured Abstract

Background. Although handwriting recognition and OCR are often considered to be solved problems, state-of-the-art models trained on specific datasets perform very poorly on real world samples. Additionally, publicly available labelled datasets are often very small, leading to difficulties in training deep learning models that typically require a lot of data. There are a number of synthetic handwriting data generation techniques in the literature, but the difference in distribution between generated data and real data limits the performance of models trained purely on synthetic data.

Aim. To refine synthetic handwriting data using limited real data so as to improve word-level classification accuracy of a model trained on the synthetic data.

Data. Two datasets are used in this project. The first consists of maintenance records from Boeing aircrafts (private), and the second is the IAM Handwriting Database (publicly available).

Methods. The recently introduced simGAN shows good qualitative and quantitative performance on the task of refining synthetic eye gaze and hand pose data. This method has been extended to the word images used in word-level handwriting recognition tasks, along with experimental modifications to the loss function.

Results. Refined images generated using the simGAN and its variants show impressive classification performance when used for training, improving top 1 accuracy from 52% to nearly 63% on the Boeing dataset.

Conclusions. Handwriting recognition continues to be a challenging problem on the wide variety of handwriting styles encountered in the wild, especially considering the rarity of large annotated datasets. Synthetic data generation coupled with refiners built using real data provide an efficient and convenient solution to this problem.

Broader impacts. The implications of data refiners go well beyond handwriting recognition alone. While deep learning has become very popular in recent years, the need for large amounts of data cannot always be met easily, and the results presented here provide a promising avenue of research.

Keywords: Handwriting Recognition, Deep Learning, Generative Adversarial Networks

2 Introduction

It is commonly believed both inside and outside the machine learning community that handwriting recognition is a problem that has been completely solved. The reality is that the existing state-of-the-art models perform well only in one of two extremely restrictive conditions: recognizing printed text, and recognizing handwriting styles similar to those on which they were trained. In addition, publicly available annotated datasets are often too small to train deep learning models effectively. It is thus a challenge to build a robust system that performs well on a wide variety of handwriting styles.

Deep learning applications that are faced with the problem of insufficient annotated data often opt to generate synthetic data. This is an attractive option as a well-built generation system offers the possibility of an endless supply of training data, allowing for deep and complex network architectures to learn from the data. However, there is a fundamental limitation to this kind of approach: the difference in distribution between the generated synthetic data and the real data that we want to perform well on. It is possible to get outstanding performance when testing on the synthetic data, primarily because of the availability of a large amount of training data. However, this does not guarantee generalization in terms of good performance on the real data. Thus it becomes important to "bridge the gap" between the synthetic data and the real data.

The aim is to build a system that is capable of taking large amounts of synthetic data and a relatively small amount of real data, and learn a mapping from the synthetic data distribution to the real data distribution. Given a new synthetic data point, the system will be capable of refining the data point so as to make the training data more similar to the real data. This is precisely what [19] attempts to do, showing excellent results on eye gaze and hand pose estimation tasks. However, it is unclear whether the approach taken, specifically with reference to the "self-regularization" and "local adversarial" loss will extend well to the task of handwriting recognition, where the images are vastly different from those used in the original paper. This project applies the simGAN method to the handwriting recognition setting, and modifies the loss function based on an improved regularization penalty and a Wasserstein component[1] so as to achieve better word-level classification accuracy using the refined images. Thus we develop a robust handwriting recognition system capable of performing well on the wide range of styles encountered in the real world.

3 Related Work

Handwriting recognition - referred to in certain contexts as Optical Character Recognition (OCR) - has always been a hot topic of research, even before the recent surge in popularity of deep learning. [17] give an exhaustive description of the various tasks, challenges and applications of the problem of handwriting recognition. This includes descriptions of the problems of word and character segmentation, and recognition tasks at the word level and at the character level, both of which are questions that need to be addressed irrespective of whether the problem is approached using deep learning. Some of the earliest successes of both CNN's [13] and of LSTM's [6] lie in the domain

of handwritten text recognition. In this project we specifically tackle the problem of offline handwriting recognition, where the data consists of images of the handwritten words as opposed to pen locations over time (which is a related but different problem called online handwriting recognition).

Most published papers using deep learning for text recognition tend to be in the realm of localization and recognition of words in natural images, such as in [21][22][3][10]. This problem is subtly different from handwriting recognition as the words in these images are typically not handwritten and tend to have other real world objects around them. The problem of handwriting recognition as it is defined for this project is addressed instead in [7][16] in an offline setting and in [5][9][11] in an online setting.

Another popular topic of research recently has been handwriting generation. [4] showed initial good results on this problem, using an LSTM that sequentially predicted pen locations after being conditioned on an input text string to write. A variant of the synthesis task is tackled in [8], where entire images of handwritten digits are generated as opposed to simply pen locations, which is more relevant to our setting of offline handwriting recognition. The data generation techniques used in this project are derived primarily from the One-Shot Learning idea introduced in [12]. The paper uses a concept called Bayesian Program Learning which fits conditional distributions in order to generate concepts (characters) as a composition of parts, sub-parts and spatial relations.

The field of generative models in general has grown significantly in recent years, fueled to a large extent by the introduction of Generative Adversarial Networks (GAN) in [2]. A number of variants of the GAN have come out since then to improve generated image quality and training stability, such as the Deep Convolutional GAN [18]. The Wasserstein GAN [1] is another variant, which made a significant stride in the direction of training stability by modifying the loss function in order to minimize an approximation of the Wasserstein distance. However, the GAN variant most relevant to this paper is the simGAN introduced in [19], in which synthetic images are refined with the help of a set of unlabelled real images, using the traditional GAN loss augmented with additional components to improve performance.

4 Data

This project uses two real world datasets. The first is a private dataset of aircraft maintenance logs from Boeing. This data consists of individually segmented words from the logs, amounting to a total size of 6,980 images with a vocabulary of 491 words.

The second is the IAM Handwriting Database [15] which contains forms of handwritten text. It has handwriting samples from 657 writers and 115,320 isolated and labeled words. All forms, extracted text lines, words and sentences are available for download as PNG files with 256 gray levels, with corresponding XML meta-information included about the image files. All texts in the IAM database are built using sentences provided by the LOB Corpus. The images corresponding to the top 500 most frequently occurring words in the dataset were used for this project, so as to

enable a direct comparison with the Boeing dataset. This brings the dataset size down to 63,924 images.

This real data was augmented with a significant amount of synthetic data - 960 sample images per vocabulary word in each dataset. This comes to a total of 471,360 synthetic images for the Boeing vocabulary and 480,000 synthetic images for the IAM vocabulary. For the purposes of this project, the synthetic data generator is treated as a black box.

5 Approach

5.1 Background

5.1.1 Generative Adversarial Networks

Generative Adversarial Networks (GAN's) were introduced in [2] as a method to sample data from a target data distribution p_{data}. A GAN involves two networks: a generator G with parameters θ_g and a discriminator D with parameters θ_d. The generator takes as input a random noise vector z drawn from a distribution p_z and generates a data point $x = G(z; \theta_g) \sim p_g$. The discriminator D then learns to distinguish between the actual data (drawn from p_{data}) and the data coming from the generator distribution p_g, by learning a function representing the probability that the data point is "real", as $D(x; \theta_d)$. The loss (value) function used to achieve this is:

$$V(D, G) = \mathbb{E}_{x \sim p_{data}}[log(D(x))] + \mathbb{E}_{z \sim p_z}[log(1 - D(G(z)))] \tag{1}$$

The first term is representative of the discriminator's ability to identify points coming from the real data distribution as genuine. The second term represents the discriminator's ability to recognize points coming from the generator as fake. The discriminator thus tries to maximize $V(D, G)$ by learning to distinguish between the two distributions. The generator tries to minimize it by learning and generating data points from the real distribution. So G and D are optimized with respect to θ_g and θ_d so as to minimize the loss functions L_G and L_D respectively, where:

$$L_G(\theta_g) = \mathbb{E}_{z \sim p_z}[log(1 - D(G(z; \theta_g)))] \tag{2}$$

$$L_D(\theta_d) = -\mathbb{E}_{x \sim p_{data}}[log(D(x; \theta_d))] - \mathbb{E}_{z \sim p_z}[log(1 - D(G(z); \theta_d))] \tag{3}$$

The paper shows that for a fixed generator G, the optimal discriminator is:

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Also, the optimal generator is one that learns the data distribution exactly ($p_g = p_{data}$) as is intuitively expected. Putting these results together, we see that if the GAN training process is carried out correctly, G learns to sample from the actual data distribution and D outputs 0.5 uniformly for both the real and generated data points, as both come from the same distribution.

5.1.2 simGAN

A common problem encountered when using synthetic data to train a model is the difference in distribution between the synthetic training data and the real testing data. This often manifests itself, for instance, in high test accuracy on held-out synthetic data coupled with low or plateaued accuracy on the real data.

[19] tackles this problem by using a GAN variant called a simGAN. The setting is very similar to the vanilla GAN setting, with the primary difference being that the input z to the generator is an unrefined synthetic image instead of a random noise vector. The generator (also called a "refiner" in this setting) refines the synthetic image to bring it to the real data distribution. In this setting, p_z becomes the distribution generating the synthetic images, which for the purposes of the simGAN can be treated as a black box.

There are 3 important modifications done to the GAN training process to ensure good performance of the simGAN. First, this setting is not a typical GAN setting in that we are not simply interested in generating data from the target distribution. Since the end goal is to actually train a model (say for classification) using the refined images, it is important that any annotation information is not lost in the refinement process. [19] handles this by adding a "self-regularization" penalty to the generator loss function, so as to discourage it from changing the original image too significantly on a pixel-wise basis. So the generator loss function becomes:

$$L_G^V(\theta_g) = \mathbb{E}_{z \sim p_z}[log(1 - D(G(z; \theta_g))) + \lambda ||G(z; \theta_g) - z||_1] \tag{4}$$

Here, L_G^V is the generator loss for the "vanilla" simGAN.

The second modification is done to prevent the generator from adding artifacts in the image in order to fool the discriminator. This is done by using a "local adversarial loss" which runs the discriminator over image patches instead of the whole image. Thus there are multiple cross-entropy values computed per image, which are summed up to get the complete loss function value for that sample.

Finally, training stability of the discriminator is improved by maintaining a history of images refined by earlier versions of the generator. Each batch of refined images fed to the discriminator during training contains images from both the current version of the refiner and from this history of images from earlier refiners. This ensures that the discriminator does not "forget" the earlier stages of training.

The simGAN idea has been modified for the context of handwriting recognition as described in the next two sections.

5.2 Abstract Loss

The self-regularization loss is a good way to retain annotation information, but having a pixel-wise penalty may not always work well. As is often the case in computer vision, images that are vastly different on a pixel-by-pixel basis can be representative of the same object or concept.

In order to alleviate this issue, the loss function has been modified to instead penalize the difference between abstract features extracted from the synthetic and the refined image, as suggested in [19]:

$$L_G^A(\theta_g) = \mathbb{E}_{z\sim p_z}[log(1 - D(G(z;\theta_g))) + \lambda||\mathcal{F}(G(z;\theta_g)) - \mathcal{F}(z)||_1]$$

where $\mathcal{F}$ is an appropriate feature extractor. Specifically in this case, we use the layers of a fixed discriminator as $\mathcal{F}$. If the discriminator can be broken down as $D(x) = S(D'(x))$, where S is the penultimate layer and the final softmax and D' represents the layers leading up to these, the generator loss function becomes:

$$L_G^A(\theta_g) = \mathbb{E}_{z\sim p_z}[log(1 - D(G(z;\theta_g))) + \lambda||D'(G(z;\theta_g)) - D'(z)||_1] \tag{5}$$

We call this variant the "Abstract" simGAN.

5.3 Wasserstein Loss

The Wasserstein GAN [1] takes a different approach to generative models by altering the loss function. The WGAN attempts to estimate the real data distribution p_{data} with the generator distribution p_g by minimizing an approximation to the Wasserstein or Earth-Mover distance between the two distributions. The Wasserstein distance is defined (using the Kantorovich-Rubinstein duality) as:

$$W(p_{data}, p_g) = \sup_{||D||_L \leq 1} \mathbb{E}_{x\sim p_{data}}[D(x)] - \mathbb{E}_{x\sim p_g}[D(x)] \tag{6}$$

The supremum is taken over all possible 1-Lipschitz functions. As it is not possible to actually optimize over the space of all 1-Lipschitz functions, in practice we optimize this in two steps. First, we minimize the following loss function over the parameter space θ_d representing the discriminator (also called a critic in this setting) for a fixed generator G:

$$L_D^W(\theta_d) = -\mathbb{E}_{x\sim p_{data}}[D(x;\theta_d)] + \mathbb{E}_{z\sim p_z}[D(G(z);\theta_d)] \tag{7}$$

Then we constrain the discriminator parameters to lie within $[-c, c]$ for a small positive constant c. While this is only an approximation to the Wasserstein distance, in practice it has been observed that this procedure improves training stability, with an additional benefit being that the loss function is a good proxy for image quality. Finally, we optimize the generator by minimizing this loss function:

$$L_G^W(\theta_g) = \mathbb{E}_{z\sim p_z}[-D(G(z;\theta_g)) + \lambda||G(z;\theta_g) - z||_1] \tag{8}$$

Note that this involves two terms. The first is the approximation to the Wasserstein distance that the discriminator gives us (or at least the part that involves θ_g), and the second is the self-regularization penalty that is used to retain annotation information. Note that in the implementation, the discriminator and generator architecture remain exactly the same as the previous cases (up to and including final layer softmax).

6 Experiments

6.1 Experimental Setup

As described earlier, experiments are run on two datasets, Boeing and IAM. The datasets consists of a set of real word images with different numbers of images per vocabulary word, and a set of synthetic images with an equal number of images for each vocabulary word. This allows us to understand the impact of the size of the real dataset on performance.

The synthetic images were used only for training in every experiment. The real images are split into 80-20 train/test sets, coming out to 5584 real training images and 1396 real test images for the Boeing dataset, and 51140 real train images and 12784 real test images for the IAM dataset. The entire synthetic dataset and the real training dataset are used for training purposes (for both the classification CNN and for the GAN) with the model's performance evaluated on the real test images.

Three simGAN variants are tested: a vanilla simGAN, a simGAN using the abstract loss (Abstract simGAN), and a simGAN using the Wasserstein loss (Wasserstein simGAN). Each simGAN is trained to make the entire set of synthetic images closer in distribution to the real training images. Following this training phase, the synthetic images are run through the simGAN to generate a set of refined images of equal size to the synthetic dataset (471,360 for Boeing and 480,000 for IAM). These are then augmented with the set of real training images, and are used to train a word-level classification model using a CNN whose architecture is described in the next section. After 10 epochs of training, Top 1 and Top 5 accuracies are calculated using the real test data. This training procedure is shown in Figure 1.

We look at a number of baselines to compare with these models. We use two state-of-the-art OCR systems, Tesseract OCR [20] and Google Cloud Vision API [1]. As these systems do not have access to the actual set of words in the vocabulary and can thus output any arbitrary sequence of characters, we use the Levenshtein edit distance [14] to compute their word-level accuracies. Specifically, we count a prediction by these systems as being "correct" if the Levenshtein distance between the prediction and the ground-truth word is less than a certain fraction of the length of the ground-truth word. We calculate the accuracy using different values of this fraction (0.25, 0.50 and 0.75), which are indicated within brackets in Table 1.

[1] https://cloud.google.com/vision

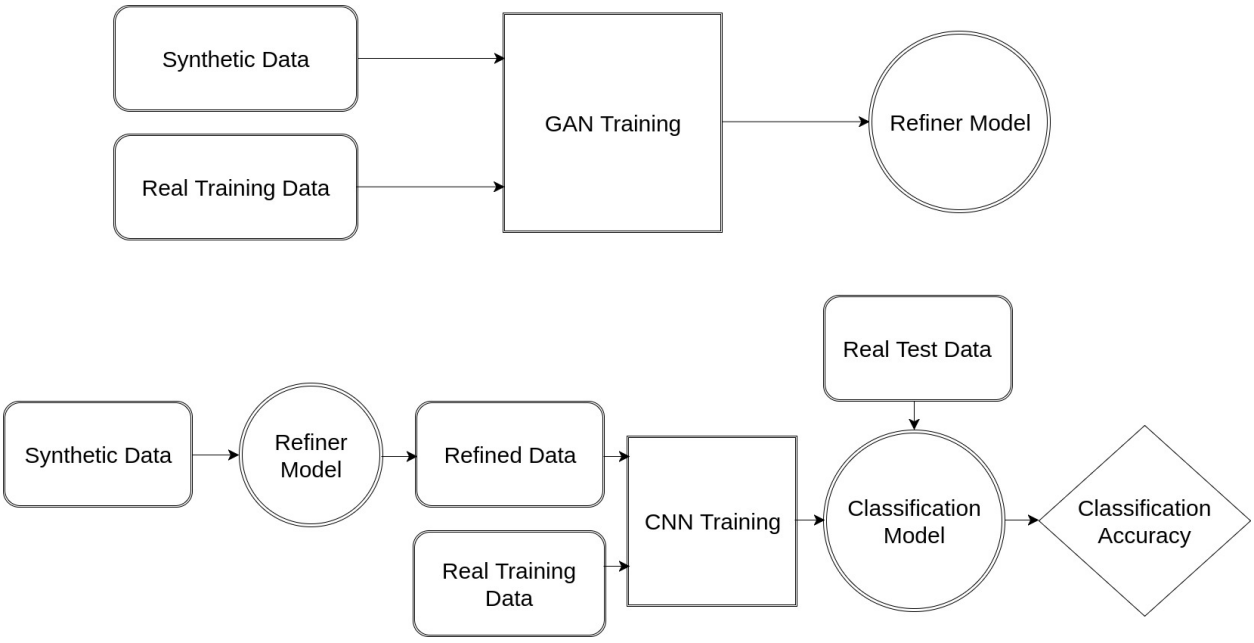


Figure 1: Training procedure followed in experiments. Note that the available real data is split into two parts, train and test, and used in different parts of the pipeline accordingly.

We also report the IAM accuracy results of the offline models used in [6]. Apart from these baselines, two additional custom models are trained to understand the impact of the refinement process on performance. The first is a CNN that is trained on the real training images alone without any synthetic data (indicated as "Real"). The second is a CNN trained on the real training images along with synthetic images that have not been subjected to (indicated as "Real")GAN refinement (indicated as "Real + Synthetic").

6.2 Implementation Details

Input images are first resized to 128 x 256 before being fed into either network. This size was chosen instead of the usual square images chosen in image classification tasks as handwritten word images usually have greater width than height, and resizing them to a square distorts them significantly. Beyond human legibility, classification performance was also seen to be higher with these resized dimensions.

The architecture of the generator and discriminator are those used for hand pose estimation in [19]. The generator consists of (i) convolutional layer with kernel size 7, stride 1 and 64 output feature maps (ii) 10 ResNet blocks, each using a kernel size of 3, stride 1 and 64 output feature maps while retaining the image size using appropriate zero padding (iii) convolutional layer with kernel size 1, stride 1 and a single feature map (iv) tanh layer outputting the final refined image.

The discriminator consists of (i) convolutional layer with kernel size 7, stride 4 and 96 output feature maps (ii) convolutional layer with kernel size 5, stride 2 and 64 output feature maps (iii) maxpool layer with kernel size 3 and stride 2 (iv) convolutional layer with kernel size 3, stride 2 and 32 output feature maps (v) convolutional layer with kernel size 1, stride 1 and 32 output feature maps (vi) convolutional layer with kernel size 1, stride 1 and 2 output feature maps (vii) softmax layer

The architecture used for the word classification CNN is (i) 7 convolutional layers, with output feature maps of size 48, 48, 96, 96, 192, 192 and 192 (ii) maxpool layers after the 2nd, 4th and 7th convolutional layers (iii) 2 fully connected layers with 4096 and 4096 hidden units (iv) softmax layer of size equal to the vocabulary size. A kernel of size 3 with stride 1 was used in all of the convolutional and maxpool layers. The fully connected layers used dropout with probability 0.5.

ADAM optimizer was used for the GAN training, and AdaGrad was used for the CNN.

7 Results

The accuracy results after 10 epochs are shown in Table 1. First, we observe that our models completely outperform Tesseract and Google Cloud Vision for both datasets, even when the threshold for accuracy calculation is set at a very lenient 0.75. Additionally, we improve over the results presented in [6] for IAM.

To understand the impact of the refinement process alone, note that all three simGAN variants (vanilla, abstract and Wasserstein) improve classification performance over the unrefined synthetic image model for the Boeing dataset. Also, the best top 1 and top 5 accuracies are achieved by the Abstract simGAN for both datasets.

Additionally, the graphs in Figure 2 show the convergence characteristics for the 10 epochs of training. Again, we see that all three simGAN variants improve over the training process that uses only unrefined images for the Boeing dataset. The Abstract simGAN and Wasserstein simGAN stay above the curve for the vanilla simGAN throughout the training process. While the Wasserstein simGAN does marginally better in the beginning, the Abstract simGAN converges to nearly the same accuracy at the end of the training phase, even performing marginally better after the final epoch as shown in Table 1.

The results of our methods are less convincing for the IAM dataset, with all the models converging to nearly the same final accuracy. The training curves are also very similar, except for the vanilla simGAN which takes longer to converge than the other models.

Model	Boeing Top 1	Boeing Top 5	IAM Top 1	IAM Top 5
Tesseract (0.25)	1.79	-	1.31	-
Tesseract (0.50)	4.01	-	2.54	-
Tesseract (0.75)	7.95	-	5.37	-
Google Vision (0.25)	10.60	-	10.47	-
Google Vision (0.50)	16.05	-	16.26	-
Google Vision (0.75)	18.91	-	19.37	-
HMM [6]	-	-	64.5	-
LSTM [6]	-	-	74.1	-
Real	28.01	50.86	77.25	92.57
Real + Synthetic	52.08	74.00	77.60	92.66
simGAN	60.03	78.80	74.17	90.94
Abstract simGAN	**62.97**	**83.38**	**79.69**	**92.98**
Wasserstein simGAN	61.89	82.52	75.11	91.38

Table 1: Top 1 and Top 5 classification accuracy of different models on the two datasets.

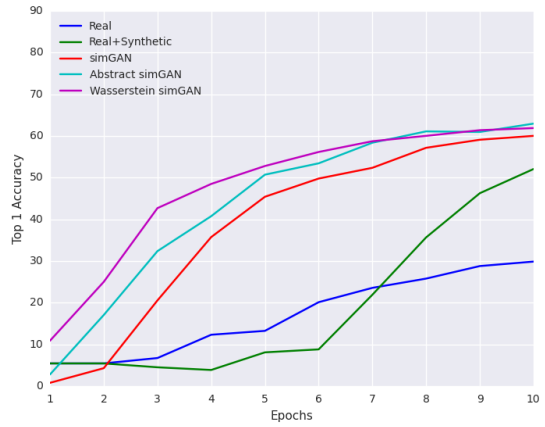
8 Discussion

As seen in Table 1, our models improve significantly over the chosen baselines. However, it is important to take these results in context, noting the advantages that our models possess to begin with. First, we have access to real training data (including the complete vocabulary set) for both datasets. Tesseract and Google Cloud Vision, however, have been trained beforehand and do not have access to training data specifically from these datasets. They are general-purpose OCR engines that can be applied to any vocabulary, which our model is currently incapable of. Nevertheless, the Tesseract and Google Cloud Vision results show that the best publicly available OCR systems perform poorly on real-world handwriting recognition tasks.

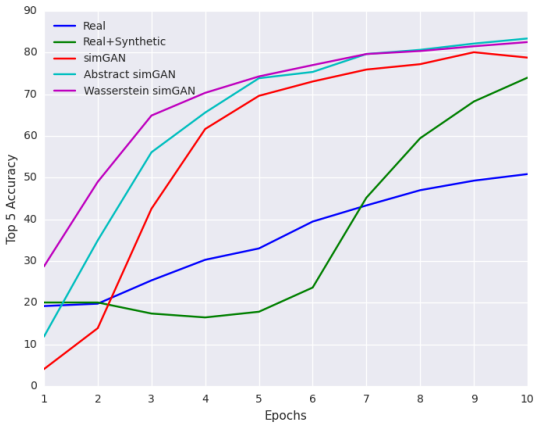
While [6] does have access to real training data, they tackle the IAM dataset as whole. We focus instead on the 500 most frequently occurring words in the dataset. While these top words still account for more than 55% of the dataset, the full vocabulary setting is clearly more challenging.

Looking at the differences in accuracy among our different models, we find some interesting patterns. For the Boeing dataset, the insufficient number of real training points leads to low accuracy when trained without data augmentation. Simply adding unrefined synthetic images using our black box generator leads to a significant boost in accuracy.

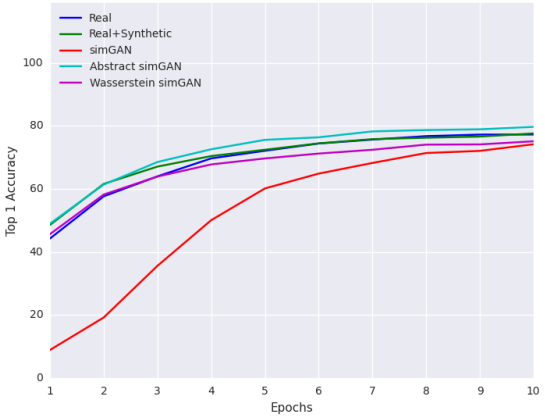
Furthermore, the improvement made by the data refinement process is clear. All three simGAN's boost the classification accuracy significantly on the Boeing dataset. The results also seem to indicate that the modified simGAN's improve slightly over the vanilla simGAN. The Wasserstein simGAN helps to improve and stabilize the training process, and it is the fastest to converge as



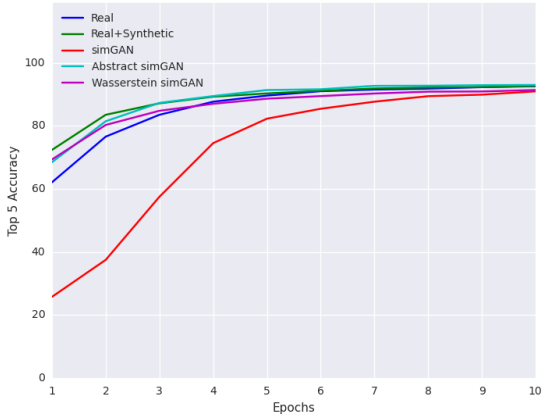
(a) Top 1 Accuracy on Boeing dataset



(b) Top 5 Accuracy on Boeing dataset



(c) Top 1 Accuracy on IAM dataset



(d) Top 5 Accuracy on IAM dataset

Figure 2: Performance during training over the two datasets

seen in Figure 2. Overall, the Abstract simGAN performs marginally better. This is likely because its loss function is better at retaining annotation information, a feature that is vital for a setting in which the refined images need to be used in a supervised manner. However, during training the Wasserstein simGAN appeared to be more robust to hyperparameter tuning. Specifically, the regularization constant λ used in the Abstract simGAN had a large impact on its performance. Using a constant that is too small leads to the network distorting the images to make them completely white, as it is not sufficiently penalized for changing the pixel values. This is a form of mode collapse, a common issue in GAN training. On the other hand, using a constant that is too large leads the network to keep the image almost unchanged, leading to no improvement in accuracy over the unrefined images.

For the IAM dataset, all of the models perform very similarly. This is likely due to the large amount of available real data. As the real data alone is sufficient to produce reasonable classification accuracy, the addition of synthetic data seems to either show no significant performance gain, or in some cases, even hinder the training process. Nevertheless, the Abstract simGAN does produce a marginal improvement in Top 1 and Top 5 accuracy over the models that do not perform any image refinement. It also does noticeably better than the other two simGAN variants on the same metrics. Additionally, we see that both modified simGAN's (Abstract and Wasserstein) converge to the optimum fairly quickly, showing an improvement over the vanilla simGAN in terms of the training process.

While the quantitative effect of the refinement process is clear, it is hard to explain the impact of these simGAN's qualitatively. To the naked eye, the refined image may look marginally sharper or blurrier than the unrefined image, but no general pattern can be observed. In order to better visualize the impact of each simGAN, the pixel-wise difference between the refined and the original image was plotted. Specifically, the pixels where the refined image is darker were plotted in blue, and the pixels where the refined image is lighter were plotted in red. Some of these illustrative visualizations are shown in Figure 3.

The Wasserstein simGAN (column 3) appears to "sharpen" the image. It makes the pixels at the corners lighters, and adds intensity to the pixels in the middle of the image. Conversely, the Abstract simGAN seems to have almost the opposite effect. It spreads out the word and makes it somewhat blurrier. For the vanilla simGAN, the effect could possibly be interpreted as a denoising of the image, but the pattern is harder to determine. More samples of this refinement analysis can be found in the appendix.

All of the effects appear to make the boundaries of the word clearer, and it can be argued that an effective classification algorithm for handwriting needs only the word contours. The pixel intensities within those contours do not matter as much. The simGAN variants thus take different approaches to make the classification process easier and more robust, which leads to improvements in accuracy as seen earlier.

9 Conclusion

In many deep learning settings, it is difficult to find large labelled real world datasets to train on. As a result, synthetic data generation has become very popular in recent years. In this paper we present and improve on the simGAN method that, given a black box data generator and a small amount of real data, can enable the generation of large amounts of data drawn from the same distribution as the real data. We analyze the performance of the simGAN method and its variants both quantitatively and qualitatively, and show that it allows for significant improvements in accuracy on an offline word-level handwriting recognition task.

Figure 3: Visualization of the refinement process. The first column shows the original image. The next 3 columns show the impact of the refinement process of the vanilla simGAN, Abstract simGAN and Wasserstein simGAN, respectively.

10 Acknowledgements

I would like to thank the Boeing Aerospace Data Analytics Lab for funding this project and related research projects in the Language Technologies Institute, under the title "Airplane Maintenance and Handwriting Recognition". I am extremely grateful to Barnabás Póczos and Ravi Starzl for their inputs and suggestions. I would also like to thank Hai Pham and Daniel Clothiaux for coding assistance and useful discussions regarding methods and frameworks to use.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[3] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay D. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *CoRR*, abs/1312.6082, 2013.

[4] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[5] Alex Graves, Marcus Liwicki, Horst Bunke, Juergen Schmidhuber, and Santiago Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 577–584. Curran Associates, Inc., 2008.

[6] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868, May 2009.

[7] Alex Graves and Juergen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. 2009.

[8] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015.

[9] Jianying Hu, Michael K. Brown, and William Turin. Hmm based on-line handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(10):1039–1045, October 1996.

[10] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *CoRR*, abs/1412.1842, 2014.

[11] S. Jaeger, S. Manke, and A. Waibel. Npen++: An on-line handwriting recognition system. In *in 7th International Workshop on Frontiers in Handwriting Recognition*, pages 249–260, 2000.

[12] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[15] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.

[16] Vu Pham, Christopher Kermorvant, and Jérôme Louradour. Dropout improves recurrent neural networks for handwriting recognition. *CoRR*, abs/1312.4569, 2013.

[17] R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, Jan 2000.

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

[19] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[20] Ray Smith. An overview of the tesseract ocr engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 629–633. IEEE, 2007.

[21] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 1457–1464, Washington, DC, USA, 2011. IEEE Computer Society.

[22] Tao Wang, David J. Wu, Adam Coates, and Andrew Y. Ng. End-to-end text recognition with convolutional neural networks.

11 Appendix

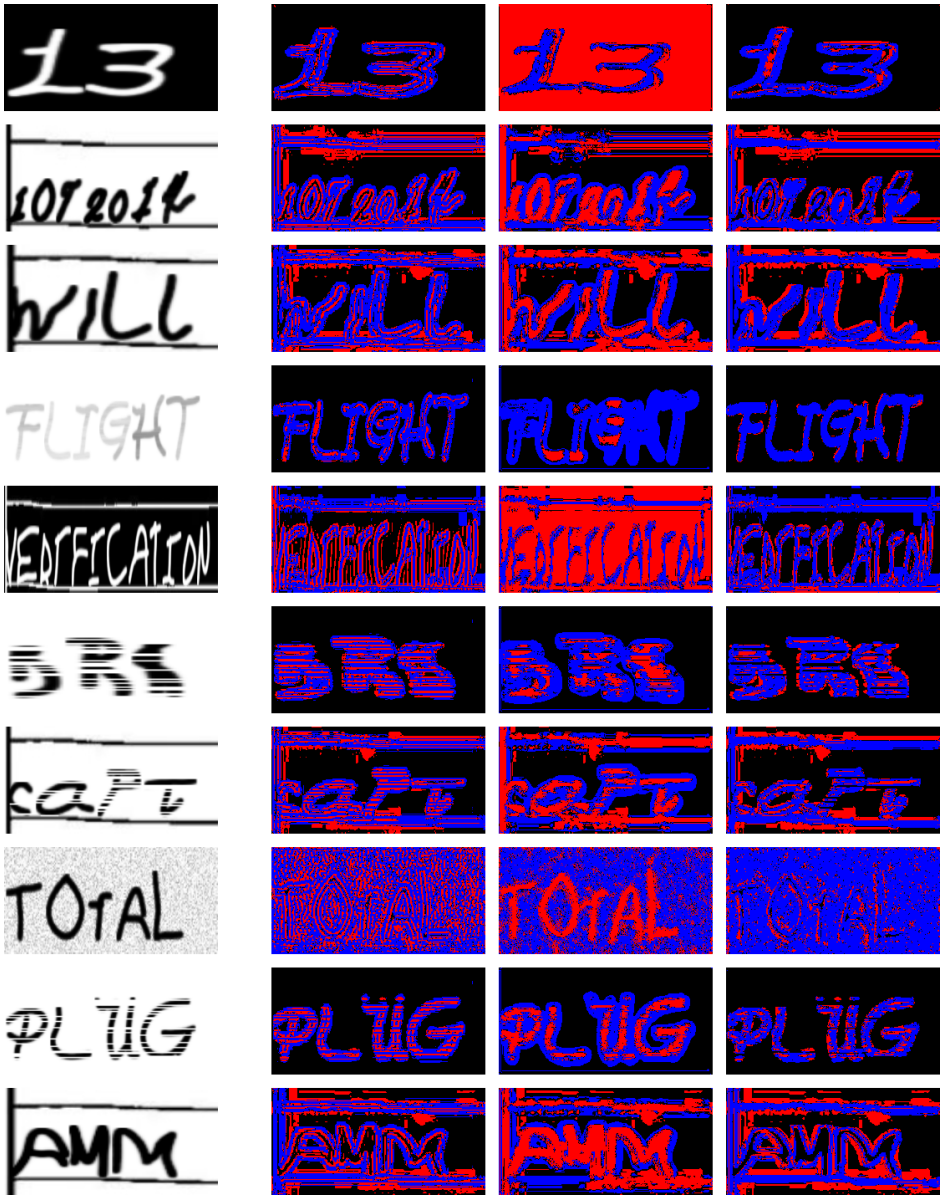


Figure 4: Additional samples of refinement process visualization. The first column shows the original image. The next 3 columns show the impact of the refinement process of the vanilla simGAN, Abstract simGAN and Wasserstein simGAN, respectively.

Figure 5: Additional samples of refinement process visualization. The first column shows the original image. The next 3 columns show the impact of the refinement process of the vanilla simGAN, Abstract simGAN and Wasserstein simGAN, respectively.