Exploiting synthetic images for real-world image recognition

Max Maton
Delft University of Technology
Delft, The Netherlands
https://in.maxmaton.nl/

Jan van Gemert
Delft University of Technology
Delft, The Netherlands
http://jvgemert.github.io/

Miriam Huijser
Aiir Innovations
Amsterdam, The Netherlands
https://aiir.nl/

Osman Kayhan
Delft University of Technology
Delft, The Netherlands
o.s.kayhan@tudelft.nl

Abstract

This paper shows that the gap between synthetic and real-world image distributions can be closed by using GANs to convert the synthetic data to a dataset which has the same distribution as the real data. Training this GAN requires only a fraction of the dataset traditionally required to get a high classification accuracy. This converted data can subsequently be used to train a classifier with a higher accuracy than a classifier trained only on the real dataset.

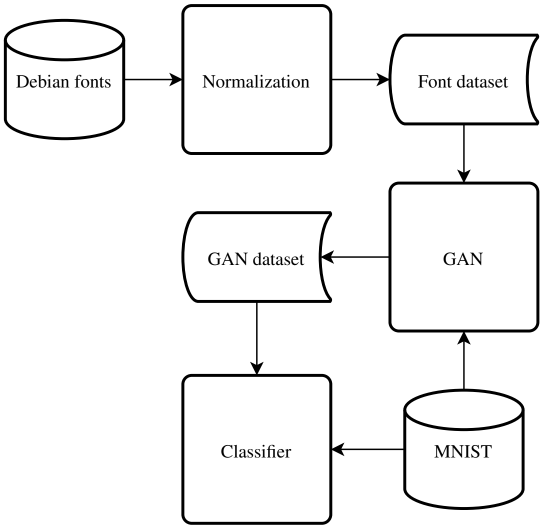

Figure 1. Technique used to convert the rendered dataset for use in training.

1. Introduction

Deep learning has revolutionised visual object recognition. Thanks to huge datasets and fast hardware (GPUs), current object recognition approaches have near-human accuracy. Because creating big datasets is often very expensive, people are starting to turn to rendered images to augment their datasets. However, training networks on rendered images may not achieve the desired accuracy due to a gap between synthetic and real image distributions [8]. Another development in current research is the increased focus on Generative Adversarial Networks (GANs) to generate images that look similar to the images they were trained on [7]. In this paper the impact of the distribution gap between synthetic and real image distributions is decreased by using a GAN to modify the rendered images to have the same distribution as real images. This technique is shown to be useful for inflating very small datasets to a level where they can be used to create accurate classifiers.

2. Related work

Rendered data can sometimes be used to train networks, i.e. using rendered images to segment images of indoor scenes [4], font character classification trained on interpolated real samples [8] and facial expression analysis using rendered faces [2]. These networks are trained by creating a rendered dataset that is as close as possible in statistical distribution as the real dataset. Generally this is very difficult or expensive to achieve.

The problem of creating rendered datasets with a distribution close to a real dataset can be solved by using domain adaptation. This is generally done using Generative Adversarial Networks [9, 3, 6] trained to create samples based on images in the rendered dataset that are indistinguisable from images in the real dataset. We decided to perform this research on the GAN as described by Bousmalis et al [3][1].

3. Inflating datasets using rendered data and GANs

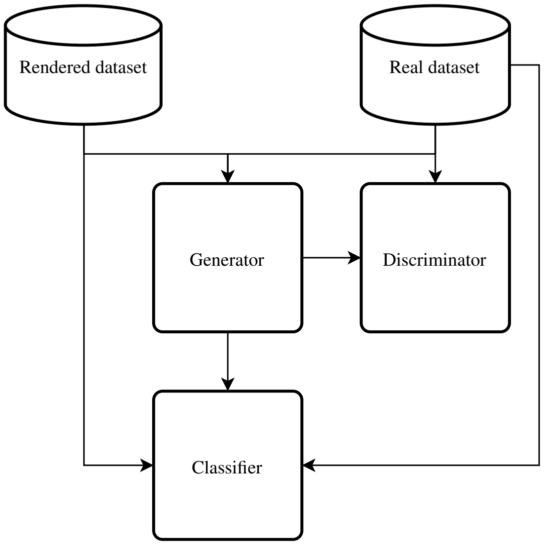

Figure 2. Overview of the GAN architecture

The basis of this research is the GAN that converts images from a rendered dataset distribution to an image that appears to come from the distribution of a real dataset. This GAN consists of three parts. The first part is a generator network that does the image conversion. The second part of the GAN is a discriminator, which tries to distinguish between the output of the generator and images from the real dataset. The third part of the GAN is a classifier that predicts the label of images coming from either the real dataset, the generator or the rendered dataset. All these networks are trained in parallel. The discriminator and classifier are trained to reduce the amount of misclassifications. The generator is trained to minimise the amount of pixels changed in the image, to minimise the loss in classification by the classifier and to try to fool the discriminator to classify the image as real.

To test the effects of inflating datasets when using this technique, we first trained the GAN with a real and rendered dataset. We then apply the trained GAN on the rendered dataset to create a new synthetic dataset. A combined dataset consisting of the synthetic dataset and the real dataset was used to train a second classifier.

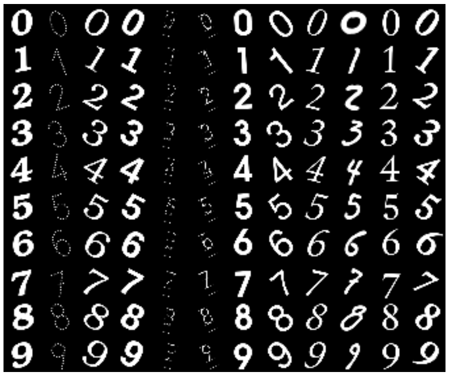

Figure 3. Example images of the MNIST font dataset

4. Experiments

We evaluated this technique on the MNIST dataset [5] together with a rendered dataset generated by rendering open source font digits in multiple rotations for all number classes. The rendered dataset contains samples from 149 fonts, with each digit rendered having 47 variations each with a distinct rotation between -47 and 46 degrees. All images were normalised using the same algorithm used to normalise the MNIST samples, the result of which can be seen in figure 3. A random sample of $10,030$ of those images were put in a test set and the remaining $60,000$ images were used as the rendered dataset.

The GAN was modified to use 10% of the training data as a validation set instead of a constant $1,000$ samples as this allowed for smaller sample sizes.

For multiple ratios[2] r, where $0 > r > 1$, we created the following datasets:

MNIST$_{\text{original}}$
$60,000 \times r$ images from the original MNIST dataset.

MNIST$_{\text{font}}$
$60,000 \times (1-r)$ images from the rendered font dataset.

We used these two datasets to train the GAN using 12 million training samples and subsequently applied the trained gan GAN on the MNIST$_{\text{font}}$ dataset to create the MNIST$_{\text{GAN}}$ dataset. This made the MNIST$_{\text{GAN}}$ dataset have the exact same size as MNIST$_{\text{font}}$.

With these datasets we trained five instances of the following classifiers:

$5 \times$ C-MNIST$_{\text{original}}$
Classifier only trained on MNIST$_{\text{original}}$

$5 \times$ C-MNIST$_{\text{font}}$
Classifier only trained on MNIST$_{\text{font}}$

[1] We chose to base our research on this paper due to the available TensorFlow[1] implementation

[2] Ratios used were: 0.001, 0.003, 0.005, 0.007, 0.008, 0.024, 0.056, 0.1, 0.3, 0.5, 0.7 and 0.9.

$5 \times$ C-MNIST$_{\text{GAN}}$
Classifier only trained on MNIST$_{\text{GAN}}$

$5 \times$ C-MNIST$_{\text{original+GAN}}$
Classifier only trained on MNIST$_{\text{original+GAN}}$

$5 \times$ C-MNIST$_{\text{original+font}}$
Classifier only trained on MNIST$_{\text{original+font}}$

Each classifier was tested on the MNIST test set which resulted in an accuracy percentage.

5. Results

To validate whether the GAN was creating useful images, we looked at the resulting images in MNIST$_{\text{GAN}}$:

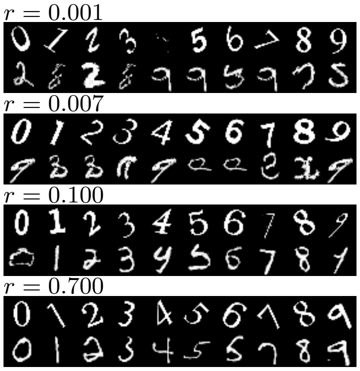

Figure 4. Example images of the MNIST font dataset. Top of each image is from MNIST$_{\text{font}}$, bottom is the generated image in MNIST$_{\text{GAN}}$

From visual inspection it is apparent that the GAN has trouble keeping the labels consistent if there is not enough real training data. An example of this are the 1 and the 2 of $r = 0.007$ in figure 4: they both result in something that looks like the same 8 instead of something that looks like a 1 or 2 respectively. This means that the classifier trained on this new dataset will receive two very similar samples with conflicting labels. We measured the accuracy on MNIST of C-MNIST$_{\text{GAN}}$ and found low accuracy for lower ratios. This supports the conclusion that there is an issue with labeling when the GAN is not trained with enough real data.

5.1. C-MNIST$_{\text{font}}$ performance

To make sure we actually test whether the GAN is able to close the distribution gap we looked at the accuracy of C-MNIST$_{\text{font}}$. For this dataset the error shows an inverse relation between the amount of samples and the performance of the classifier as can be seen in figure 5. This indicates that a gap exists between the rendered font dataset and the MNIST test dataset.

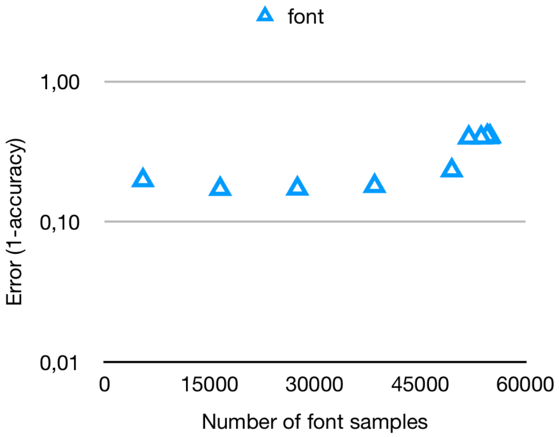

Figure 5. Performance of C-MNIST$_{\text{font}}$ with varying amounts of MNIST$_{\text{font}}$ training data. Performance decreases with more training samples indicating that MNIST$_{\text{original}}$ and MNIST$_{\text{font}}$ have different characteristics

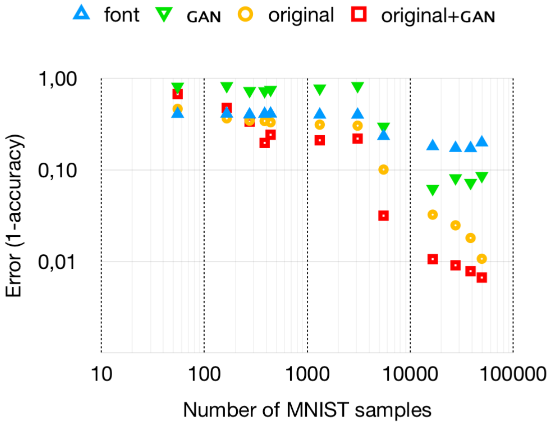

Figure 6. Performance of classifiers with different amounts of MNIST training data. Training with the GAN data improves results after a minimal amount of original MNIST samples

5.2. C-MNIST$_{\text{original}}$ against C-MNIST$_{\text{original+GAN}}$

We compared the accuracy of C-MNIST$_{\text{font}}$, C-MNIST$_{\text{GAN}}$, C-MNIST$_{\text{original}}$ and C-MNIST$_{\text{original+GAN}}$ to see whether training with the MINST$_{\text{original+GAN}}$ dataset is better than training with one of the individual datasets. The result of this comparison can be seen in figure 6.

There seems to be a minimum amount of real samples after which the GAN starts to produce meaningful data. In this case C-MNIST$_{\text{original+GAN}}$ is only more accurate than the C-MNIST$_{\text{original}}$ when the GAN is trained on more than 385 real images ($r = 0.007$).

Using MNIST as a test case, this technique is able to close the distribution gap even with as little 0.7% real data in the resulting dataset.

We also compared the difference in wrong predictions

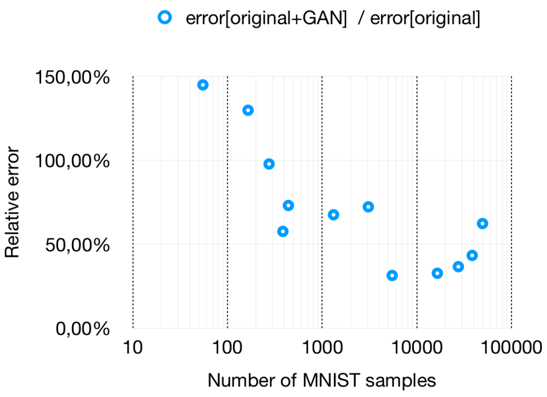

Figure 7. Error of C-MNIST$_{\text{original+GAN}}$ compared to the error of C-MNIST$_{\text{original}}$, lower is better. Lowest relative error was found at $5,500$ MNIST samples.

between C-MNIST$_{\text{original}}$ and C-MNIST$_{\text{original+GAN}}$. As is shown in figure 7, the highest reduction in error was achieved with $5,500$ MNIST images and $49,500$ font images. This indicates that this technique becomes more effective when the ratio between real and rendered images becomes less extreme.

5.3. C-MNIST$_{\text{original}}$ against C-MNIST$_{\text{original+font}}$

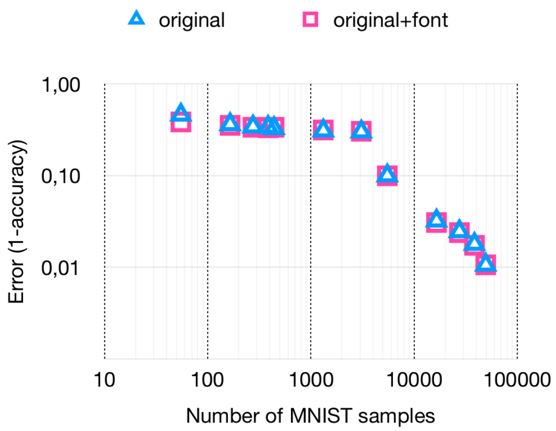

Figure 8. Performance of C-MNIST$_{\text{original+font}}$ versus the performance of C-MNIST$_{\text{original}}$ for varying amounts of training data. Performance is almost the same indicating no advantage to including font data in the training set directly.

To confirm that the improved results of C-MNIST$_{\text{original+GAN}}$ can not be solely explained by the addition of more data, we compared C-MNIST$_{\text{original}}$ with C-MNIST$_{\text{original+font}}$ to see if the addition of font data increased the accuracy. From the results in figure 8 we had to conclude that C-MNIST$_{\text{original+font}}$ has performance roughly equal to C-MNIST$_{\text{original}}$ at the ratios where C-MNIST$_{\text{original+GAN}}$ is more accurate. The addition of more data cannot explain the additional accuracy of C-MNIST$_{\text{original+gan}}$ indicating this increase is a property of the transformation made by the GAN.

6. Discussion

Due to a limited amount of time, no optimization has been done on the hyperparameters of the GAN. Optimizing these hyperparameters might further improve classification accuracy which should result in a further reduction in the amount of required samples from the real dataset.

When measuring the performance of C-MNIST$_{\text{original+GAN}}$ versus C-MNIST$_{\text{original}}$ we found that even though the generated samples are not good samples for their target class, C-MNIST$_{\text{original+GAN}}$ is much more accurate than C-MNIST$_{\text{original}}$. This effect can partially be explained by results shown by Sukhbaatar et al. that indicate that deep learning is robust to massive label noise. [10] This would mean that the network is still capable of learning higher level features from the mislabeled samples and is able to succesfully ignore the bad labeling.

7. Conclusion

Using MNIST as an example, we were able to show that it's possible to create a rendered dataset. We've shown that a distribution gap exists between the dataset we created and the real MNIST dataset and were able to close this gap using our described technique.

We've shown that GANs can be used to inflate trainingsets by reducing the gap between synthetic and real datasets. Furthermore, they can do so with very little real training data.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] I. Abbasnejad, S. Sridharan, D. Nguyen, S. Denman, C. Fookes, and S. Lucey. Using synthetic data to improve facial expression analysis with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[3] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *2017 IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 95–104, July 2017.

[4] J. Cano-Perez, J.-C. Pérez-Cortes, J. Arl, and R. Llobet. Training set expansion in handwritten character recognition. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops SSPR 2002 and SPR 2002 Windsor, Ontario, Canada, August 6–9, 2002 Proceedings*, 01 2003.

[5] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, Nov 2012.

[6] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. *Domain-Adversarial Training of Neural Networks*, pages 189–209. Springer International Publishing, Cham, 2017.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[8] H. S. B. Jean Nonnemaker. Using synthetic data safely in classification. In *PROCEEDINGS VOLUME 7247 IS&T/SPIE ELECTRONIC IMAGING — 18-22 JANUARY 2009*, volume 7247, pages 7247 – 7247 – 11, 01 2009.

[9] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1857–1865, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[10] S. Sukhbaatar, J. Bruna, M. Paluri, L. D. Bourdev, and R. Fergus. Training convolutional networks with noisy labels. In *ICLR Workshop track*, 2014.

A. Raw experiment results

table