Fingerprint Inpainting with Generative Models

s1739461 s1313429 s1771851 (Group 107)

Abstract

We propose and test two generative models to perform inpainting on fingerprint data. These models fill in missing patches of fingerprint images based on surrounding context. The first uses a deep residual architecture with a combination of reconstruction loss and adversarial loss, inspired by Generative Adversarial Networks, thereby encouraging mode selection as opposed to mode averaging. The second model we experiment with is a PixelCNN++. It produces qualitatively different fingerprint images. We demonstrate that they perform equally well when classifying inpainted fingerprints from 220 *subjects with AlexNet. Our models significantly outperform the classification of non-inpainted images, which indicates their potential use in modern identification systems where fingerprints are often partly occluded. Finally, we present an example of a real-world application and a break-down test of our inpainting models.*

1. Introduction

While supervised deep learning has shown remarkable results in many application domains (Szegedy et al., 2016), semantically labelling data by hand is time and labour-intensive. As such, the unsupervised deep learning paradigm, which tries to learn a suitable representation of data without using any labels, has seen a new wave of popularity in recent years. Here we focus our attention on a specific unsupervised task: *inpainting*. This was recently demonstrated by Pathak et al. (2016), who removed large patches from street view images and used a Convolutional Neural Network (CNN) to reconstruct the missing pixels. Their primary insight was to add an adversarial term to the loss function, as inspired by earlier work on Generative Adversarial Networks (GANs) Goodfellow et al. (2014). This approach was designed to encourage mode selection as opposed to mode averaging. The direct result thereof was more realistic and sharp inpainting as opposed to blurry inpainting. While inpainting has been shown to be a relevant feature extractor in the transfer learning paradigm, owing to time constraints we do not consider this aspect in this research.

Generative models are a particularly powerful set of unsupervised learning techniques that learn a low-dimensional representation of the data and sample from this to produce new data. The idea is that the new data exists in close vicinity to the training data on a high-dimensional hyper-plane specific to the problem domain. GANs are recognised as the current state-of-the-art for such generative tasks and have been shown to produce convincingly realistic images by learning data distributions implicitly (e.g. (Radford et al., 2015)). They can be described as the competition between a generator, which learns to produce images that look real, and a discriminator, which learns to discriminate fake and real images. These two components learn in unison. Cao & Jain (2018) showed that GANs alone can produce random, fake fingerprints virtually indistinguishable from real ones.

GANs often show unstable learning behaviours; an alternative class of generative models are *autoregressive models*. These estimate data distributions explicitly by minimising the negative log-likelihood of some well-formulated probability distribution estimation. One such model, the PixelCNN++ (Salimans et al., 2017), generates images pixel by pixel by decomposing the probability density over colours as conditionals on previously generated pixels. They are, however, notoriously slow to train.

In our earlier work (Group107, 2018), we provided an autoencoder baseline for the inpainting task on fingerprint images. We achieved somewhat disappointing results because the autoencoder alone did not perform mode selection; instead, it mostly produced averaged-out, grey pixels, resulting in blurry images. In this work, we aim to improve inpainting performance by using a combination of deep Residual Neural Networks (ResNets) (He et al., 2016) and GANs, as well as a PixelCNN++. We show that the inpainting results are much more accurate with such models, and demonstrate how to quantify their relative performance by solving a classification problem using inpainted and non-inpainted fingerprints. We run two final experiments: First, we intentionally mask poor-quality fingerprint regions, showing how these can be inpainted to "fix" the fingerprints, and second, we perform a break-down test by inpainting iteratively larger masked regions.

Section 2 briefly discusses the fingerprint data set. Section 3 and Section 4 introduce the GAN and PixelCNN++ inpainting models respectively. We compare these models by means of a classification task in Section 5. Finally, we test our model on a real-world application in Section 6 and perform a breakdown study in Section 7.

2. Data: Fingerprints

We have provided a detailed description of the fingerprint datasets we use herein in our previous work (Group107, 2018). Thus, we only provide a brief summary of this description in what follows.

2.1. Overview

From smart-phone authentication to evidence in criminal trials, fingerprints are ubiquitous in modern biometric systems. However, the images at hand are a challenge in deep learning since they suffer from high intra-class variation (owing to varying skin moisture, scanning deviations, artefacts from ageing, etc.) and low inter-class variation (i.e., fingerprints often look similar). As such, modern fingerprint matching technology typically relies on heuristic algorithms (Maltoni et al., 2009). These look to extract a combination of three levels of features: (1) The overall shape of the fingerprint, (2) the minutiae points, namely where the ridge lines either come to an end (ridge-ending) or split in two (bifurcation), and (3) the sweat pores visible on the ridges of the fingerprint. We show that our inpainting can often reproduce the first two features. The third feature requires high resolution scanners and is often not available; it is not inpainted by our model.

2.2. The Data

There are several fingerprint datasets available for direct download or via request. We obtained a number of these in our earlier work. It is important to note that those datasets only contain a small number (eight maximum) of fingerprints per class, i.e per finger.

Figure 1. Examples of fingerprint crops with four random patches.

A detailed description of the data we have used to construct suitable datasets for this research can be found in Group107 (2018). Owing to the insights gained through this earlier research, we have made several changes regarding the usage of the data described therein:

1. **Input data size**. We are now using 128×128 pixel crops of the fingerprints instead of 384×288 pixel full fingerprints. This is because our earlier work showed that a wider context did not improve inpainting performances. Moreover, the smaller input image sizes allowed us to fit deeper models on the GPU memory.

2. **Data standardisation**. In our earlier work we standardised fingerprints on a dataset level, because they each used different scanners. The abovementioned cropping made this step less relevant, so we now follow a simple approach of rescaling the data to the range $[-1, 1]$. We match this scale in the output images by using $tanh$ activation function in the final layer of our generative models.

3. **Patching**. We have set the patching protocol to place four patches of size 32×32 pixels.

4. **Data split**. Two of the original databases were held out as validation and test data, to ensure that the inpainting is invariant to the physical scanner used during data collection. The validation set is used for comparison in Sections 3 and 4, and both sets are used for the classification task in Section 5.

No other changes from our earlier work have been made to the data. Figure 1 gives examples of the 128×128 patched fingerprint crops used as input data throughout this work. Note that many of the images displayed in this work are of full fingerprints as opposed to those shown in Figure 1. These are computed by processing overlapping regions of 128×128 pixels and averaging the resultant output, using soft contributions at the region edges.

3. Experiment 1: GAN Inpainting

3.1. Description

In our earlier work on fingerprint inpainting we used a patch autoencoder which produced blurry regions (Group107, 2018). This was mostly due to the limited depth of the model and to the autoencoder prioritising "mode averaging" rather than "mode selection". In other words, the autoencoder often found that predicting an average pixel value (grey) would result in a lower reconstruction error. Pathak et al. (2016) proposed to include a GAN component to the inpainting pipeline for that very purpose. What follows in this section describes how we

implemented a ResNet as the generator component for a GAN, and encouraged mode selection which improved inpainting.

3.2. Implementation

After some preliminary hyper-parameter search, we decided to use the architecture shown in Figure 2 for the ResNet generator and discriminator. Deeper architectures were shown to yield a lower reconstruction loss (Group107, 2018), and so we used the deepest architecture that would fit on the GTX 1060 GPU used. We then focused our experimentation on the addition of an adversarial loss.

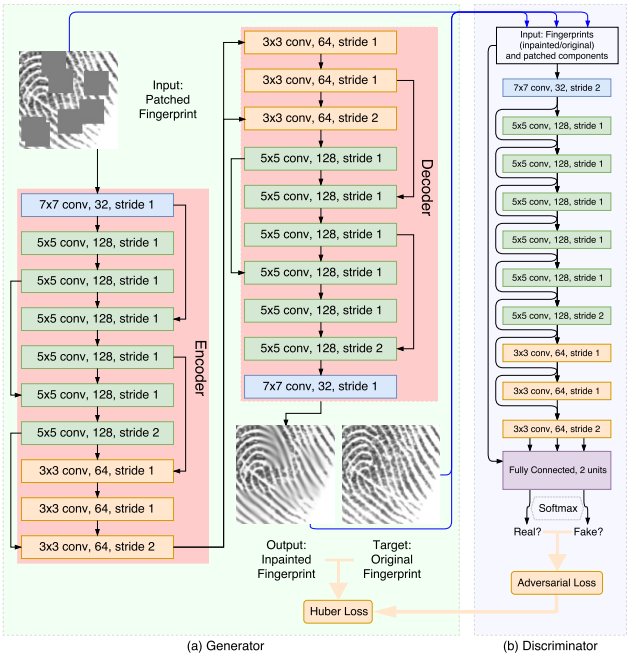

Figure 2. (a) and (b) are the ResNet architectures for the generator and discriminator components of the GAN. For each convolution we show the respective kernel size, number of output channels and stride length used.

Figure 2 (a) illustrates the ResNet generator architecture we used for this work, while Figure 2 (b) is the architecture of the discriminator component of the GAN. Note that the discriminator has a skip connection from the input layer to the fully-connected output layer. Its architecture, including the small residual skip connection length, was found through a brief hyper-parameter search. Each building block shown in Figures 2 (a) and (b) consists of a convolution, leaky ReLU activation and batch normalisation.

The output of the generator component – an inpainted fingerprint – is used as input to the discriminator network shown in Figure 2 (b) to compute the adversarial loss, which we define in what follows. Let $G(\mathbf{x}^{(i)})$ be the the inpainted images that are produced by the generator G, $\mathbf{x}_0^{(i)}$ be the original images, and $\mathbf{x}^{(i)}$ be the patched images. For a batch of size M, the "fake" images $G(\mathbf{x}^{(i)})$ and the "real" images $\mathbf{x}_0^{(i)}$ are both run through the discriminator D and classified. The adversarial loss is then defined as the negative cross-entropy (Goodfellow et al., 2014):

$$L_{adv} = \frac{1}{M}\sum_{i=1}^{M}\Big[\log(D(\mathbf{x}_0^{(i)})) + \log(1 - D(G(\mathbf{x}^{(i)})))\Big] \tag{1}$$

The arguments of the log terms being probabilities, we have $L_{adv} \leq 0$ and intuitively the generator and discriminator play a "mini-max" game on it respectively. This adversarial loss term is then added to a reconstruction loss L_{recon}, for which we used the Huber Loss (Huber, 1964), to yield the loss L_G of the generator component:

$$L_G = \lambda_{adv} L_{adv} + \lambda_{recon} L_{recon} \tag{2}$$

The discriminator loss L_D is given by (Goodfellow et al., 2014):

$$L_D = -\frac{1}{M}\sum_{i=1}^{M}\Big[\log(1 - D(G(\mathbf{x}^{(i)})))\Big] \tag{3}$$

The generator loss L_G and discriminator loss L_D are optimised in turn for each batch. Both training losses were optimised via the Adam Optimizer with a small learning rate decay of 2.5×10^{-4} % per training step.

The main hyper-parameters that have influence over the success or failure of the addition of adversarial loss are the loss mixing coefficients λ_{recon} and λ_{adv} for reconstruction and adversarial losses, respectively (see Equation 2). We tested many combinations of these over several weeks, three of which are compared to a ResNet without adversarial loss in the following section.

3.3. Results

Figure 3 shows the validation reconstruction losses between original, non-patched images and inpainted images for different values of λ_{adv}, where we kept $\lambda_{recon} = 1$ constant. In addition, we also included a model with $\lambda_{adv} = 0$ to show how the model would train without a GAN component. All the λ_{adv} values are kept extremely small because we quickly found out that larger values would result in basically no reconstruction at all.

By looking at Figure 3 and the qualitative images between inpainted images, we conclude that $\lambda_{adv} = 2 \times 10^{-6}$ and $\lambda_{recon} = 1$ performs best, with the exception of the pure ResNet autoencoder. This is to be expected, as the ResNet

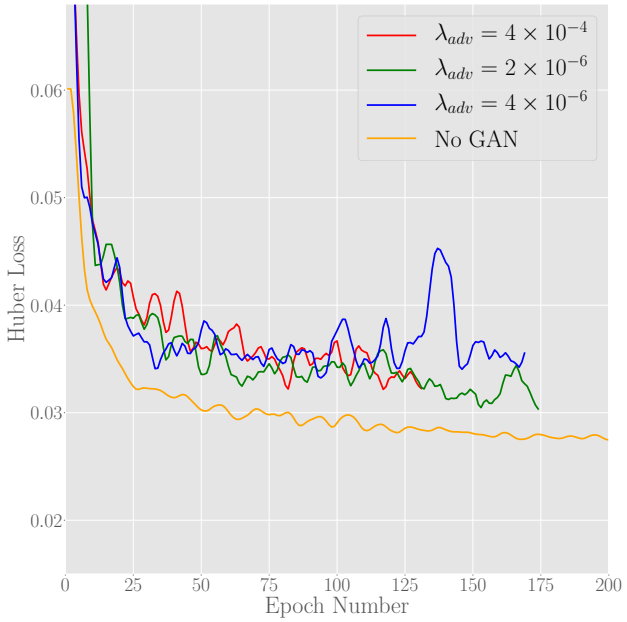

Figure 3. Huber Loss on the validation set for inpainting reconstructions with different values of λ_{adv}, as well as without the adversarial loss component ($\lambda_{adv} = 0$). A smoothing function is used to reduce the jitter of these curves.

autoencoder's aim is to reconstruct the patched pixels, unlike the adversarial models which also aim to generate realistic patches. In fact, while the model without GAN yields a lower loss, it doesn't yield images that are as visually sharp.

The inpainting performance of the aforementioned best GAN model is demonstrated in Figure 4 (e) and (f). While the inpainted patches are somewhat blurry, the generated lines are clearly recognisable as the contrast between ridges and valleys is much more prevalent than in our previous work (Group107, 2018).

4. Experiment 2: PixelCNN++ Inpainting

4.1. Description

A PixelCNN (van den Oord et al., 2016) is an image density model that uses autoregressive connections to model images pixel by pixel. The basic idea is to decompose the image distribution as a product of conditionals:

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i \mid x_1, ..., x_{i-1}) \tag{4}$$

where n is the image dimension and is 128 in our case.

A PixelCNN learns a distribution over the entire colour-

Figure 4. (a) and (b) are original fingerprints selected from the validation set. (c) and (d) are the same fingerprints after applying patches that cover up to 25% of the images. (e) and (f) are inpainted using the generator component of the GAN set-up. (g) and (h) are the inpainted results using the PixelCNN++.

space for each pixel. This approach does not leverage the knowledge that similar colours will have similar likelihoods of being present. This and several other pitfalls motivated the advances that resulted in the crafting of the PixelCNN++ (Salimans et al., 2017), which models the colour of each pixel as a mixture of K logistic sigmoid functions ($K = 10$ in the original paper). Thus, for any given pixel, the PixelCNN++ models:

$$p(x_i|\pi_i, \mu_i, s_i, x_{<i}) = \sum_{j=1}^{K} \pi_{i_j} \left[\sigma\left(\frac{x_i + 0.5 - \mu_{i_j}}{s_{i_j}} \right) - \sigma\left(\frac{x_i - 0.5 - \mu_{i_j}}{s_{i_j}} \right) \right] \tag{5}$$

where x_i is the pixel colour being estimated, π_{i_j} is the factor for the j^{th} logistic sigmoid for the i^{th} pixel, μ_{i_j} is the mean of the j^{th}logistic sigmoid, s_{i_j} is the scale thereof, and σ is the logistic sigmoid function. Furthermore, the colour channels are linked so that the green channel depends linearly on Equation 6 and the value of the estimated red channel, while the blue channel depends on its estimation and a linear combination of the red and green channels. Owing to the grayscale nature of fingerprints, we discard this component.

All parameters in the neural network and the loss function that defines the colour of each pixel are learned through the direct minimization of the negative log-likelihood of Equation 6, allowing the PixelCNN++ to define a powerful and flexible relationship between pixel intensities in an image.

4.2. Implementation

Our implementation is largely adapted from the original PixelCNN++ code published by OpenAI[1]. Nonetheless, some changes were needed, such as making the PixelCNN++ gray-scale friendly. We also set $K = 3$ as the model needs less flexibility to estimate gray pixels versus colour pixels. Other than this, the default parameters were used. The main challenge was the long training time of such a model, which could be up to over a week on a GTX 1060 GPU. This greatly limited the amount of hyperparameter fine-tuning we could perform.

4.3. Results

We trained a PixelCNN++ model for nearly 60 epochs using the parameters used in Salimans et al. (2017). We do not believe that the model has converged at this point, but the extremely slow training time is prohibitive. Figure 5 demonstrates the bits per dimension loss for the PixelCNN++ training. Bits per dimension is commonly used to compare explicit distribution estimators, such as autoregressive models, because the log-likelihood is directly accessible. It is, essentially, the loss per dimension in bits and is defined as:

$$bpd = \frac{1}{\log(2) n^2} \sum_{i}^{n^2} -\log p(\mathbf{x}) \tag{6}$$

for a single image, $\mathbf{x}$, with n^2 pixels. For instance, a value of $bpd = 2.92$ was obtained using PixelCNN++ on CIFAR-10 data (Salimans et al., 2017).

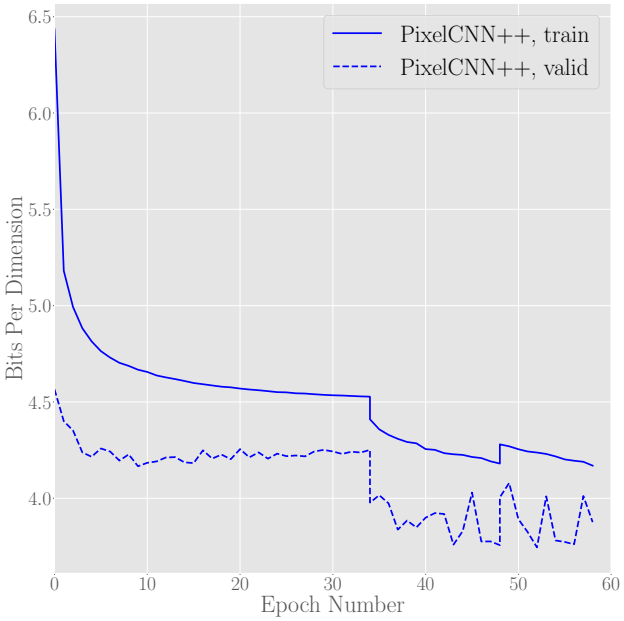

Figure 5. Bits per dimension loss for inpainting reconstructions using PixelCNN++.

The minimum validation bits per dimension we obtained was 3.74; it was at this point that the model was frozen and used for the inference tests of the following sections. Interestingly, the validation loss is lower than the training loss. This might be attributed to the fact that the held out validation data represents 'easier' fingerprints to model, but most likely is the result of strong regularisation during training but not for inference (i.e. dropout).

Figure 4 (g) and (h) give examples of the inpainting using PixelCNN++ on randomly placed patches. The quality of the inpainting is different from that of the GAN-based inpainting. In particular, the PixelCNN++ inpainting seems more noisy and "pixelated". The autoregressive nature of this approach means that the reconstruction errors compound from top to bottom (owing to the PixelCNN++ conditioning on pixels above in its receptive field). Nonetheless, the inpainting using PixelCNN++ is certainly able to

[1] github.com/openai/pixel-cnn

complete regions in a realistic fashion. The main advantage this autoregressive model has over an adversarial approach is that the resultant inpainting is not blurry. Upon close inspection it becomes evident that the ridge-valley line completions are more true to the original fingerprint when using the PixelCNN++. However, the associated speckle-type noise is a clear drawback compared to the smooth lines the GAN ResNet generates.

5. Experiment 3: Classification

5.1. Description

As mentioned previously, finding a metric to evaluate the quality of the inpainting task can be challenging. This is because the loss function does not necessarily indicate that the *relevant* features of the fingerprints have been inpainted. Here, we define "relevant" features as those features which can be used to discriminate one finger from another. Intuitively, this may correspond to the overall shape of the fingerprint and its minutiae points (see Section 2.1), although it is not constrained to be just those.

One simple way of quantifying the quality of our inpainting model is to run a classification task on patched images, both with and without inpainting. We expect the classifier to perform better on the inpainted images if the regions have been filled in with relevant features. Note that in this experiment we are not concerned about getting a state-of-the-art classification accuracy across classes. Indeed our approach is not amenable to real-world classification, since there could be millions of fingerprints in a given database, each of which would need to be a class in our classifier network. Instead, we are solely interested in the relative performance of the classifier for different inpainting cases.

5.2. Implementation

We perform a standard classification task on 220 subjects from datasets that were excluded from training in sections 3 and 4. As previously, there are 8 impressions per subject, taken by the same scanner but in different conditions, such as various moisture levels. These conditions are randomised as we split each class into 6, 1, and 1 impressions for the training, validation and test set respectively. We run the classifier on four versions of the dataset:

- without patches;
- with patches;
- with patches inpainted with the GAN model; and
- with patches inpainted with the PixelCNN+ model.

The latter three cases were patched using 27 patches of size 32x32 pixels, with overlapping allowed. This corresponds to a maximum of 25% of the total area of the input images, which are cropped 374x300 pixels (see Group107 (2018)).

Since we do not have much data and are interested in the relative classification accuracy, we consider a fairly superficial model: AlexNet (Krizhevsky et al., 2012). We implement this architecture exactly as per the original paper, which most notably splits the 2^{nd}, 4^{th} and 5^{th} convolutional layers into two groups, in order to reduce the number of parameters. We train this model with the Adam Optimizer and a decaying learning rate. This converges in about one hour on a GTX 1060 GPU.

The small number of images per class poses a serious challenge when training a deep CNN. We propose two solutions. First, we augment the input images with a random ± 20 degree rotation. Then, we perform cross validation on the training and validation sets: we perform 7 training runs, each using different images for the validation set. This allows us to average the validation accuracy across all those runs when comparing different cases of inpainting.

5.3. Results

The final validation accuracies, when averaged over all cross-correlation runs, are shown in Figure 6. All training runs (not shown in that particular figure) converge quickly to 100% training accuracy, which is a sign of overfitting. This was expected owing to the small number of fingerprint impressions per class. In fact, there are so few examples that even with data augmentation our shallow architecture is data-starved, meaning it can remember some of the training data without necessarily learning a relevant representation thereof.

Considering the above, it may seem counter-intuitive to try to classify fingerprints given the data we have access to. The reader is thus reminded, however, that this experiment is intended to show the relative performance of fingerprint inpainting procedures.

While the absolute validation accuracies are not particularly high, the relationship between those curves demonstrates the value of our inpainting models. Indeed, the accuracy is increased by over one third when inpainting the missing patches, which means that some relevant features for classification were inpainted correctly. The GAN and PixelCNN+ inpainting seemed to perform similarly well.

6. Real-World Use Case

To test the real-world applicability of the models learned through this research, we apply them to fingerprints where regions have been manually identified for replacement. These regions typically correspond to wrinkles or acquired scars, and may not be present in the fingerprints of the same

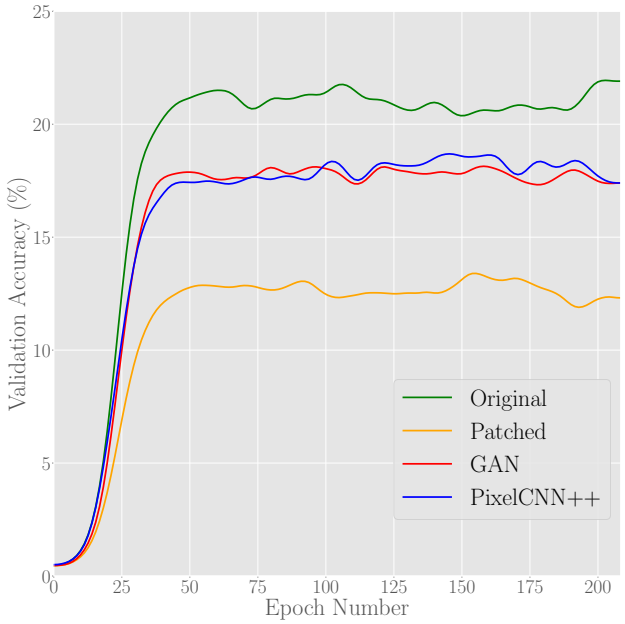

Figure 6. Validation accuracies for patched fingerprints with and without inpainting. These curves are averages of 7 cross correlated runs. The classification accuracy without any patching is also shown as a baseline. A smoothing function is used to reduce the jitter of these curves.

Figure 7. Real-world use case: fingerprints intentionally patched to encourage reconstructions in poor-quality regions. From left to right: original fingerprint, manually patched, GAN inpainting, and PixelCNN inpainting.

finger in a database. Examples can be seen in Figure 7.

Of course this inpainting procedure of poor-quality regions would not necessarily improve the classification accuracy of the fingerprints, should all impressions in the dataset contain the same wrinkles/scars, for instance. This remains to be explored as future work, however. Nonetheless, the reconstruction is highly encouraging in that the continuity of both inpainting approaches is good, resulting in realistic 'wrinkle-free' fingerprints.

7. Break-down Testing

The models we train must learn to leverage the surrounding context in order to perform inpainting. It is not immediately clear, however, how close the context must be to the inpainted pixels in order to be effective. A simple breakdown test is demonstrated in this section, as we inpaint a central patch of increasingly larger sizes, in order to assess at what point the inpainting procedures fail. This is demonstrated in Figure 8.

It can be seen that breakdown occurs at approximately the same patch size that was used for training, which is somewhat expected. However, it is also evidence that much of the advantage gained over our earlier work (Group107, 2018) is owing to the much deeper ResNet architecture employed instead of the adversarial loss that used to assist in training the generative component of the GAN. Indeed, a GAN alone would inpaint the squares with realistic fingerprint features even without any context. Interestingly, the GAN ResNet tends to merge lines together when the inpainting is ambiguous, which is often not the case in the real fingerprint counterpart.

The PixelCNN++ inpainting, however, fails in a fashion consistent with the autoregressive structure of the model. That is, each pixel is conditioned on those above it in a convolutional manner. This receptive field structure means that regions at the lower end of the patches in Figure 8 are more noise-prone. Overall, the GAN ResNet can inpaint larger regions than the PixelCNN++.

8. Conclusion

In this report we demonstrated the inpainting performance of a ResNet autoencoder with an adversarial loss and a PixelCNN++ model on a dataset of patched fingerprints. For the former model, we explored different weightings of the adversarial loss component, while we used standard parameters for the latter model. Both models showed promising inpainting performances: the adversarial model generated realistic but blurry patches, while the PixelCNN++ resulted in sharper but more pixelated inpainting. We evaluated the inpainting performances of both approaches by

framing a classification problem. Classifiers trained on fingerprints inpainted via both models worked similarly well and much better than a classifier trained on the patched fingerprints, albeit not as good as one trained on the original, non-patched fingerprints.

We have considered the real-world applications of our methods, in particular their use in denoising poor-quality fingerprints. This has a direct application in auto-completion of partial latent fingerprints in the field of criminal investigation. Lastly, we tested the robustness of our inpainting model by inpainting increasingly large patches. We found that the inpainting performance starts to become unreliable when the patch size becomes bigger than the patch sizes used in the training data.

The results of this particular work are encouraging and could, given minor adjustments, have some direct applications for the biometric sciences. However, due to the complexity of the models, there is much that we have not experimented with and which could be the basis for future work. For instance, if we were to continue this work, we would aim to use a siamese architecture for determining whether fingerprints originate from the same source. Albeit a possibly combinatorial rephrasing of the fingerprint classification problem, this is how current state-of-the-art fingerprint identification systems work. Moreover, posing the problem as such would circumvent the issue of low numbers of data samples per class.

If there was more time, we would have also liked to add an adversarial loss to the PixelCNN++ and blend our two presented approaches together. This is fundamentally difficult owing to the differences in how these models estimate data distributions and is thus interesting for general domains too.

The inpainting models could also be used as data-augmenters for deep learning classification. One could augment the entire image patch by patch and get an output fingerprint without scars or line discontinuities. One could also use our ResNet encoder model as a feature extractor and transfer the representation learned across datasets. At last, we would also like to test our model more thoroughly by showing how much context is actually used during inpainting. Perhaps a context much smaller than 128x128 yields satisfactory inpainting, thereby allowing for deeper model or faster training.

Figure 8. Breakdown test for incrementally larger patches. From left to right: fingerprints with patches, GAN processed result, and PixelCNN++ processed result.

9. Acknowledgements

Our Python *TensorFlow* codebase was partly adopted from the Machine Learning Practical GitHub (https://github.com/CSTR-Edinburgh/mlpractical) and amended with inspiration from Antreas Antoniou's GitHub (https://github.com/AntreasAntoniou).

References

Cao, Kai and Jain, Anil K. Fingerprint synthesis: Evaluating fingerprint search at scale. 2018.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Group107. Mlp coursework 3: Fingerprint inpainting. 2018.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huber, Peter J. Robust estimation of a location parameter. *The annals of mathematical statistics*, pp. 73–101, 1964.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

Maltoni, Davide, Maio, Dario, Jain, Anil K, and Prabhakar, Salil. *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.

Pathak, Deepak, Krähenbühl, Philipp, Donahue, Jeff, Darrell, Trevor, and Efros, Alexei A. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016. URL http://arxiv.org/abs/1604.07379.

Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL http://arxiv.org/abs/1511.06434.

Salimans, Tim, Karpathy, Andrej, Chen, Xi, and Kingma, Diederik P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Szegedy, Christian, Ioffe, Sergey, Vanhoucke, Vincent, and Alemi, Alex A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR 2016 Workshop*, 2016. URL https://arxiv.org/abs/1602.07261.

van den Oord, Aaron, Kalchbrenner, Nal, Espeholt, Lasse, Vinyals, Oriol, Graves, Alex, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pp. 4790–4798, 2016.