Generative Adversarial Networks for Decorative Initial Image Generation

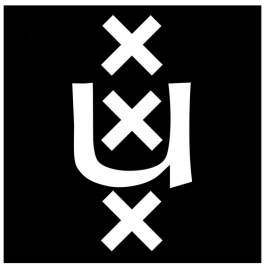
SUBMITTED IN PARTIAL FULLFILLMENT FOR THE DEGREE OF MASTER OF SCIENCE

Corine Jacobs
10001326

MASTER INFORMATION STUDIES
HUMAN CENTERED MULTIMEDIA

FACULTY OF SCIENCE
UNIVERSITY OF AMSTERDAM

July 16, 2018



1st supervisor
MSc Gjorgji Strezoski
University of Amsterdam

2nd supervisor
Dr. Nanne van Noord
University of Amsterdam

Generative Adversarial Networks for Decorative Initial Image Generation

Corine Jacobs
University of Amsterdam, Netherlands
corine.jacobs@student.uva.nl

ABSTRACT

In this thesis we will apply a generative machine learning model to a dataset of annotated images of decorative initials. The dataset is annotated with the following attributes: the depicted letter, the country of creation, the city of creation and the name of the printer. The generative model we use is the Generative Adversarial Network (GAN). GANs consist of two models (usually neural networks) which can be seen as adversaries: one model (the generator) learns to generate realistic looking data points, whilst the other model (the discriminator) learns to discriminate between the real and generated data points [20]. The models then iteratively improve each other in a minimax game.

The research question is formulated a follows: are GANs capable of generating (semi-)realistic images of decorative initials based on certain attributes, whilst capturing the stylistic characteristics related to those attributes? We trained two types of GANs on the dataset of annotated images: the Deep Convolutional GAN (DCGAN) [46] and the Auxiliary Classifier GAN (ACGAN) [41]. We show that both GANs are able to pick up the general shapes of letters and to a lesser extent their decorations. This is reflected as well in our quantitative evaluation (inception score, SSIM, classification accuracy) of the GANs (see Tables 1, 2 and 3). From the results we also noted that unconditional generation with the DCGAN, and conditional generation on the 'letter' attribute perform significantly better than conditional generation with the ACGAN on the 'country', 'city' and 'name' attributes. This corresponds to our expectations as letters are also easier for humans and classifiers (see Section 4.2) to detect and recognize than the country/city of creation or the printer. Finally we also show that training the ACGAN on a subset of the training set containing only the letter 'S', boosts the performance of conditional generation on the 'country' attribute (compared to the full dataset).

We conclude that the DCGAN and ACGAN are capable of generating semi-realistic images of decorative initials when the generation is not conditioned, or when the generation is conditioned on the depicted letter, and that generation conditioned on 'country', 'city' and 'name' is a more difficult task.

1 INTRODUCTION

Artificial intelligence has made great strides in recent years, with the continuing increase in computer power facilitating deep learning, but also with the constant development of new algorithms and machine learning models [30]. One of the AI fields which has made a great progress in recent years is that of generative modeling [24]. Generative models are a class of machine learning models which are used to generate data points. Examples of these types of models include the Restricted Boltzmann Machine (RBM) [50] and the Variational Auto-Encoder (VAE) [28].

One of the most groundbreaking generative models however is the Generative Adversarial Network (GAN) proposed by Ian Goodfellow in 2014 [20]. GANs consists of two models (usually neural networks) which can be seen as adversaries: one model (the generator) learns to generate realistic looking data points, whilst the other model (the discriminator) learns to discriminate between the real and generated data points. The models then iteratively improve each other in a minimax game. With the new game-theoretic approach, GANs greatly improved the quality of the output for generative models compared to their predecessors (e.g. RBM and VAE) [20].

GANs have been applied to numerous academic fields and real world problems as will be discussed in section 3. Among these fields is that of Art History. The studies that are available on GANs and art usually focus on style transfer (e.g. [69] and [67]). In this paper the focus will be on the less researched area of art generation using GANs.

Art history as an academic field concerns itself with the analysis of style, meaning and historical context of art works [23]. The analysis of style is hereby often used to date, localize and attribute works to certain artists (connoisseurship) [12]. Art historians look at symbols and motifs used (iconography), but also at harder to define properties of style like size of brush strokes, thickness of the paint application and the used color palette [17]. In general, these properties seems to show big differences across eras and geographical areas and minor differences between artists which have worked in the same time and location [12].

In this thesis the goal will be to condition the generation process of GANs on attributes related to style; country/city of creation and artist/printer. For both researchers in generative models and art history it is interesting to see if a GAN conditioned on these type of attributes is able to learn stylistic themes related those attributes. For example, will it be able to detect and reproduce the stylistic components of artworks which are specific to a certain country? This is usually the way art and book historians work to date and localize the origin of certain works. Will GANs be able to use these type of attributes to generate images which show stylistic characteristics related to those same attributes?

For this study we will use a dataset of decorative initials labeled with several attributes (see Section 4.1). An example of such a decorative initial can be seen in Figure 1. Decorative initials are a type of book decoration which are usually found at the beginning of sentences, paragraphs or chapters [8]. Often the decoration in and around the letter of the initial relate to the text surrounding text. An example of this is depicted in Figure 2, in which the initial depicts Adam and Eve. This is relevant to the surrounding text (Genesis) as it will later on describe the creation of Adam and Eve. In many other cases however, the decorations are not related to the text, thereby serving only a decorative function.

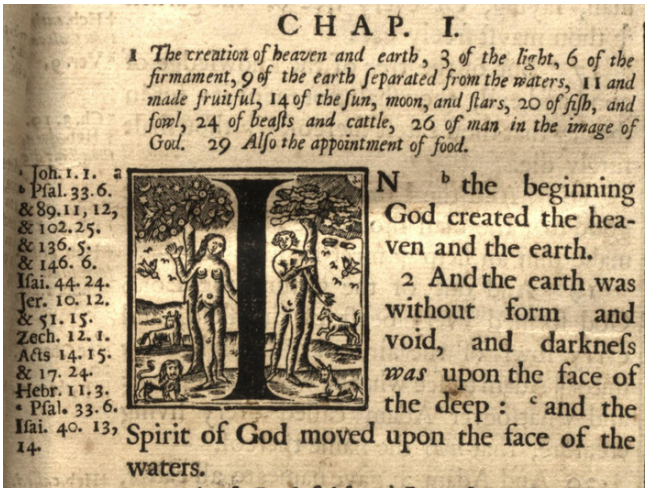
Figure 1: Example of a decorated initial. Giovanni Maria Viotti, date unknown, photo by the Special Collections of the UvA: [https://flic.kr/p/EFMMwb].

In this thesis the goal will be to tune a GAN to produce (semi-) realistic looking images of decorative initials conditioned on attributes, thereby contributing to the study of generative models and their use on art datasets. This can be summarized in the following research question: are GANs capable of generating (semi-)realistic images of decorative initials based on certain attributes, whilst capturing the stylistic characteristics related to those attributes?

Before moving on to the results, we will first discuss the related work and give more background on generative models and GANs in particular.

2 GENERATIVE ADVERSARIAL NETWORKS

Generative Adversarial Networks (GANs) were proposed by Ian Goodfellow et.al. in 2014 [20]. Since then a lot of research has been performed on GANs. Currently there are many different variations of GANs and there are many decisions to be made with regards to the architecture, hyper-parameters and loss function. In the remainder of this section we will more thoroughly examine generative models and GAN architectures in particular.



CHAP. I.

1 The creation of heaven and earth, 3 of the light, 6 of the firmament, 9 of the earth ſeparated from the waters, 11 and made fruitful, 14 of the ſun, moon, and ſtars, 20 of fiſh, and fowl, 24 of beaſts and cattle, 26 of man in the image of God. 29 Alſo the appointment of food.

a Joh. 1. 1.
b Pſal. 33. 6. & 89. 11, 12, & 102. 25. & 136. 5. & 146. 6. Iſai. 44. 24. Jer. 10. 12. & 51. 15. Zech. 12. 1. Acts 14. 15. & 17. 24. Hebr. 11. 3.
c Pſal. 33. 6. Iſai. 40. 13, 14.

IN b the beginning God created the heaven and the earth.

2 And the earth was without form and void, and darkneſs *was* upon the face of the deep: c and the Spirit of God moved upon the face of the waters.

Figure 2: First page of the Book of Genesis of a church bible. Printed by Charles Bill, 1701, London. Image taken from [9].

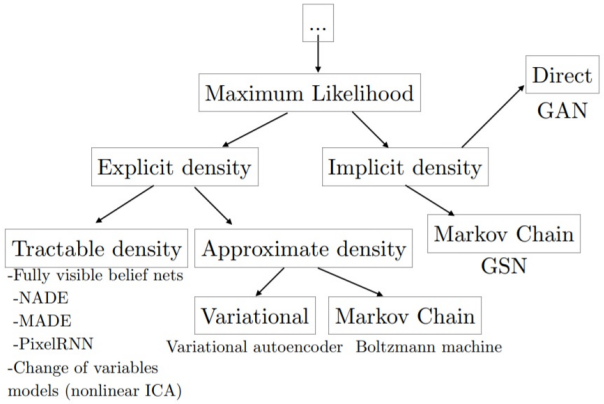


Figure 3: Taxonomy of generative models. Image taken from [21].

2.1 Generative Models in General

Generative modeling is an area of machine learning in which researchers try to model the underlying distribution of a dataset [16]. Once this distribution is found, it is possible to draw new samples from this distribution to create new data points which are similar to those in the training set [16]. In the case of images, each image is a data point, and drawing new data points from the underlying distribution can be seen as generating 'new' or 'unseen' images.

Traditionally, generative modeling focused on explicitly finding the distribution of the data. If this is a tractable problem, the distribution can be calculated directly (e.g. PixelRNN or PixelCNN [42]). If it is not tractable, the distribution can be approximated (e.g. Variational Auto-Encoders (VAEs) [28]). For an illustration of the taxonomy of generative models see Figure 3. With the invention of Generative Adversarial Networks however, the underlying distribution will no longer be explicitly approached, but implicitly, as will be described in the next section.

2.2 GANs

In a Generative Adversarial Network (GAN), two models (usually neural networks) are implemented to compete against each-other in a minimax game [20]. One model functions as the forger and is usually called the 'generator'. The job of the generator is to create as realistically as possible new data points (e.g. images). The other model can be thought of as the connoisseur and is usually called the 'discriminator'. The job of the discriminator is to differentiate real data points from those created by the generator. As the generator and discriminator are competing in the minimax game, they drive the performance of the GAN, iteratively improving each other. It is now also straightforward to see why GANs only implicitly work with the underlying distribution. The GAN does not approximate a density, but is more directly focused on the main task: generating realistic looking data points via a game-theoretic approach.

The architecture described in the GAN paper by Goodfellow [20] is usually referred to as the 'vanilla GAN' (e.g. [29], [57]). In this architecture both the generator (G) and discriminator (D) are multilayer perceptrons and the entire system can be trained using

backpropagation. The objective function for both models can be formulated as follows:

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

With x as the input/training data and z as random noise. Generated images should be labeled as fake, images from the training data should be labeled as real. This means that the discriminator will try to maximize V. The generator on the other hand wants to trick the discriminator as much as possible, thereby trying to drive the value for V downward. In this sense G and D are adversaries in a minimax game. The objective function of the GAN can then be summarized as follows:

$$\min_G \max_D V(G,D)$$

Another way to look at this objective is that it tries to minimize the difference (divergence) between two distributions: that of the real data (p_{data}) and of the generated data (p_g). In most of the practical applications of GANs (e.g. [46]) this minimization of divergence between p_{data} and p_g is implemented by using Binary Cross Entropy loss (BCE loss). In the original GAN paper however Janson-Shannon divergence (JS-divergence) was used [20] and alternatively the Kullback-Leibner divergence (KL-divergence) is also sometimes used [40]. These functions measuring the distance between two distributions are generally referred to as f-divergences [32].

2.3 Training GANs

Though GANs are very successful, they are not easy to train. There are several reasons as to why the training of a GAN is unstable:

- Its hard to achieve Nash equilibrium in the minimax game [48]. Updating the gradients of both D and G at the same time cannot guarantee convergence.
- GANs are perceptible to mode collapse [3]. They might fail to learn the complex diversity of the dataset and get stuck in outputting data points with very low variety. This is likely due to the fact that the optimization functions (f-divergences) used by GANs are generally mode-seeking, rather than mode-covering [45] (see fig. 4). This means that they are more focused on capturing one mode really well, instead of approximating over all modes.
- Gradients might vanish if the discriminator performs without error [2]. The loss function will be zero, after which there is no gradient to update. On the other hand, if the discriminator performs badly, the generator does not receive appropriate feedback to base its adjustments on.
- In general, f-divergences won't converge nicely, making training also less stable [54].

A different loss function was suggested in the paper by Arjovksy et al. in 2017 [3]; the Wasserstein-1 distance or Earth-Mover (EM) distance. GANs implementing this loss function are generally referred to as WGANs (Wasserstein GANs). Using Wasserstein distance is said to improve the stability of learning, get rid of mode collapse and is to provide meaningful learning curves for debugging and hyper-parameter searches.

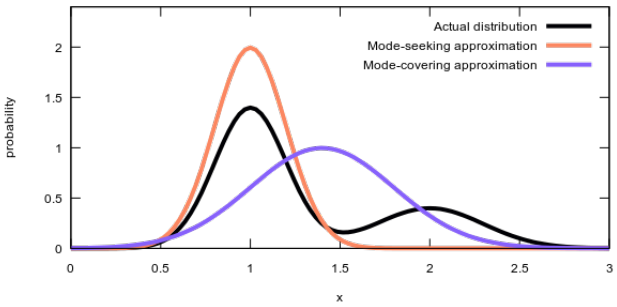


Figure 4: Mode-seeking versus mode-covering f-divergences. Image taken from [38].

WGAN did indeed make progress in stabilizing training, but WGAN would in some cases still fail to properly converge [22]. A study by Arora et al. has additionally shown that WGAN still suffers from problems with generalizability and that it lacks the prove for the existence of an equilibrium [4].

Researchers have also focused on improvements to the training of GANs in other ways than through the loss function. Salimans et al. for example [48] introduced several techniques intended to encourage convergence of GANs. Taken directly from their paper:

- Feature matching: Instead of directly maximizing the output of the discriminator, the new objective requires the generator to generate data that matches the statistics of the real data, where it uses the discriminator only to specify the statistics that are worth matching.
- Minibatch discrimination: any discriminator model that looks at multiple examples in combination, rather than in isolation, could potentially help avoid mode collapse.
- Historical averaging of parameters: When applying this technique, we modify each player's cost to include a term $||\Theta - \frac{1}{t}\sum_{i=1}^{t}\Theta[i]||^2$, where $\Theta[i]$ is the value of the parameters at past time i. This appears help with finding Nash equilibrium in the minimax game.
- One-sided label smoothing replaces the 0 and 1 targets for the discriminator with smoothed values, like .9 or .1. This is only applied to the positive labels and therefore called 'one-sided'. It is said to reduce the vulnerability of the adversarial networks.
- Batch normalization was shown to greatly improve the performance of GANs [46]. It however caused the output to be very dependent on the other inputs from the same mini batch. This is where VBN (Virtual Batch Normalization) comes in. In VBN Each data sample is normalized based on a fixed batch (*reference batch*) of data rather than within its mini batch. The reference batch is chosen once at the beginning and stays the same through the training.

Having a better understanding of the training process allows us to explore different types of architectures for GANs; most notably the convolutional GAN and the conditional GAN.

2.4 GANs for Image Generation

The GAN from the original paper by Goodfellow et al. was trained on MNIST, TFD, and CIFAR-10 (all image datasets). Since their GAN used fully connected networks, the images in these datasets were flattened to form appropriate model input.

The use of convolutional neural networks (CNNs) instead of fully connected networks followed naturally as GANs were mostly used for image generation. The use of CNNs is however not without problems as training convolutional GANs is generally even less stable than fully connected GANs [13].

The first GAN architecture which attempted to improve the training of convolutional GANs was the LAPGAN by Denton et al. [15]. LAPGAN uses a Laplacian pyramid of adversarial networks. The idea is that, instead of having just one generator CNN that creates the whole image, we have a series of CNNs which increasingly generate images with a higher resolution. The LAPGAN helped to improve the quality of generated images, but training was still quite unstable.

In 2015, Radford et al. introduced changes in the conventional convolutional GAN architecture that contributed to a more stable training process [46]. In the paper they explore the optimization of GAN architectures by imposing several constraints. They call GANs which follow the constraints Deep Convolutional Generative Adversarial Networks (DCGANs). The constraints allowed for the training of deeper models and higher resolution output images. The recommended constraints are as follows:

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

These constraints were not based on mathematical deduction, but rather on an extensive model exploration. According to the evaluation in [46], the DCGAN constraints resulted in stable training across a range of datasets (LSUN, Imagenet-1k, and a newly assembled Faces dataset). The DCGAN constraints are still widely used today as a basis for a relatively stable convolutional GAN. Examples of GANs following the DCGAN rules include the infoGAN [11] and DiscoGan [27].

2.5 Conditional GANs

The GANs discussed thus far are capable of generating images which resemble images from a training dataset. This is an interesting objective in itself, but for the application of GANs it would be increasingly useful if we have greater control over the generation process. Conditioning generation on certain attributes can give us more tools to generate meaningful new datapoints. For example, it is not very useful to generate a random image of a face, but it might be more meaningful to take an image of a face and generate a new version of it in which we change the hair color.

Besides convolutional GANs for better image generation, researchers thus also quickly started to expand the GAN architectures with possibilities to condition data generation. The original

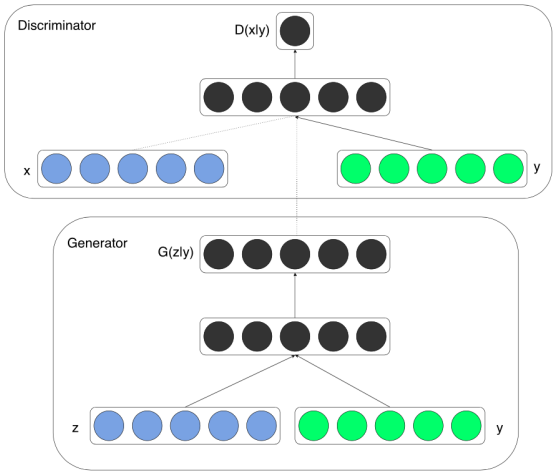


Figure 5: Schematic representation of a conditional GAN with x as the input data, y the attribute vector and z random noise. Image taken from [37].

GAN paper by Goodfellow et al. even already suggested this in their future work [20]. The first conditional GAN was introduced by Mirza et al. [37]. Conditional GANs differ from vanilla GANs in that they supply both the generator and the discriminator with an extra vector of data/attributes (see Figure 5).

Taking conditional generation of images as an example; for training we need a dataset with annotated/labeled images (e.g. the celebA dataset [1]). In such a scenario x would be a set of images from the training set (a minibatch) and y the set of corresponding attribute values (Figure 5). In a fully-connected cGAN the images from x can be flattened, after which the corresponding attribute vector y can be appended [37]. In a convolutional GAN the attributes can be added to the images as an extra channel.

The trick in a conditional GAN is to learn to disentangle the attributes from the attribute vector in the latent space of the neural network [36]. This means that the values of a specific set of latent variables will correspond to a certain attribute y. The neural network is then able to uniquely encode each attribute in the latent space.

3 RELATED WORK

The applications of GANs are very varied and are certainly not restricted to image generation. In the following section we shall first describe some of the main areas in which GANs have been applied. Next we will give a more comprehensive overview of applications of GANs which are more relevant for this thesis: art and fonts.

3.1 Applications of GANs in general

Next to images, GANs are used for other types of data like text (e.g. poems or speeches [65]), music (e.g. [62] and [65]) and video (e.g. [58]). These areas mostly focus on artistic data, but GANs have for example also been applied to medical data (e.g. [39], [49] and [63]) and physics data (e.g. [14] and [43]). Since this thesis will

focus on image data these research field will however not be further explored.

As will be discussed in Section 2, GANs come in various architectures. Many GANs are capable of directing the data generation by conditioning it on certain labels or text (text-to-image generation, e.g. [37] and [19]) or conditioning it on other images (image-to-image generation). In this way an adversarial network can, for example, learn to generate realistic images based on a (human drawn) sketch or transform a day-scene into an night-scene (e.g. [25] and [26]).

GANs have also been used for tasks other than purely generating new samples, some examples include; style transfer (e.g. [34] and [25]), super-resolution (e.g. [31]), image blending (e.g. [61]) and image in-painting (e.g. [44] and [64]).

3.2 GANs and Art

The application of GANs on art usually concern style transfer, super-resolution and image in-painting. They are less often applied to the conditional generation of artworks. Not much research has for example been done into art generation conditioned on stylistic properties like country of creator or name of the artist. This might be in part due to the lacking a proper, well annotated, art dataset; researchers might simply prefer to work with well known and high-quality datasets like MNIST, celebA, CIFAR, etcetera. Another advantage of the continuing use of those datasets is that performance between different models can be appropriately compared if they are applied to the same datasets.

There are some studies which have (in part) been dealing with their own art datasets. For example the CylceGAN proposed in [69], which transfers the artist style. Some of their results can be seen in Figure 6. Another example is the Auto-Painter by Liu et al. which is capable of transforming sketches into colored cartoon images [33] and the model as proposed in [67] to transfer coloring style between manga/anime images.

3.3 GANs and Fonts

For this thesis we use a dataset consisting of decorative initials (e.g. see Figure 1 and Section 4.1). These decorative initials do not only share properties with art datasets (style of the initial decorations), they also share some similarities with the application of GANs on font generation (the depicted letters in the initials). Thus far, there are no publications on the application of GANs on decorative initials. However font generation using GANs has been explored in two academic publications [5, 7] and a few blogposts [6, 56] which explore font generation using GANs.

The first academic publication introduces the Multi Content GAN (MCGAN) [5]. In this model missing letters from a subset of the alphabet are reconstructed. The generation in the MCGAN is split up into two sub-tasks: (1) learning the mask (outline) for the letters and (2) learning the color and texture of the images (as they use very colorful fonts with gradients and several other effects). The models starts with generating the mask and then uses this mask as a regularizer during the generation of the color and texture of the letters.

A second academic study we found did not focus on the style of separate characters, but performed font-to-font translation at word/sentence level [7]. The advantage of this approach is that it involves less preprocessing as there is no need for character segmentation.

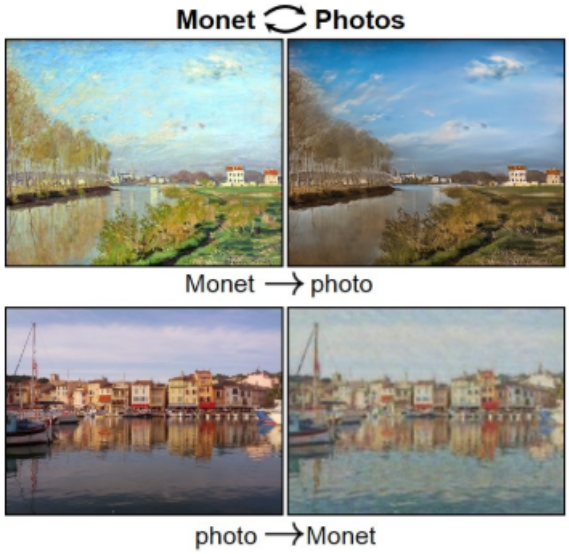


Figure 6: Some of the image-to-image translations results from the CycleGAN. Image taken from [69].

The first blog on the application of GANs on font generation shows a GAN trained on a large set of digital fonts (50,000) scraped from the web. One of the results presented in blog [6], can be seen in Figure 7. In this model, the GANs are trained on a whole alphabet of letters in the same font. At training time, some of the letters are left out, after which it is the job of the generator to reconstruct these missing letters. In this case the fonts were all in black-and-white so there was no need to learn the color and texture of the letters. Another blogpost [55], shows the same type of results on Chinese calligraphy instead of digital Latin fonts. This model is dubbed the zi2zi model and was based on (and named after) the pix2pix model [25].

4 METHODOLOGY

The research question was formulated as follows: are GANs capable of generating (semi-)realistic images of decorative initials based on certain attributes, whilst capturing the stylistic characteristics related to those attributes? The following subsections will further explain the dataset, models and evaluation methods that are used to answer the research question.

4.1 Dataset

The dataset used in this thesis consists of 30,650 annotated images of decorative initials from the collection of the UvA Special Collections. The annotations contain the following attributes:

- The depicted letter - 26 classes
- Country of creation - 11 classes
- City of creation - 65 classes
- Name of the printer - 374 classes

Figure 7: Analyzing 50k fonts using deep neural networks. For each pair of characters above, the real character (held out from the training set) is on the left and the generated output of the GAN on the right. They were generated based on two attributes: which font we want and what letter we want. Image generated by Erik Bern, [https://erikbern.com/2016/01/21/analyzing-50k-fonts-using-deep-neural-networks.html].

The dataset is not uniformly distributed among these classes. An overview which shows how many images are annotated as each of the classes can be found in Appendix A. To get a better sense of the contents of the dataset (and the great inner-class variety) some samples labeled as the letter 'S' are shown in Figure 8.

The original images are in high resolution and not nicely cropped. For this study the images were therefore first preprocessed: cropped to a square and then downscaled to 64x64 pixels. As initials were traditionally printed using only black ink [18] are all used in grayscale, making the full dimensions for each image 64x64x1. The dataset described in this section will from here on out be referred to as the *initials dataset* for brevity sake.

For our experiments we have extracted a second dataset from the initials dataset. This dataset consist of all the images from the initials dataset which are labeled as the letter 'S'. This *'S'-dataset* is used to also train GANs on. The idea behind this dataset is that we hope for the generator to be better able to generate the stylistic differences between countries, cities and names, if the depicted letter is kept static. The letter 'S' was picked as this is the most occurring letter in the initials dataset (also see Appendix A). In total, the 'S'-dataset consists of 2564 images.

4.2 Models

Experiments are done in two models: an unconditional GAN and a conditional GAN. With the unconditional GAN (DCGAN) we hope to see if a GAN is capable of generating (semi-)realistic images of decorative initials in the first place. The conditional GAN (ACGAN) is then used to evaluate the capability a GAN to capture style characteristics related to certain attributes (e.g. country of creation).

Figure 8: Nine images from the training dataset which are labeled as the letter 'S', showing the great inner-class diversity of this dataset.

Both the conditional and unconditional GAN use convolutional architectures. For the unconditional GAN a custom coded DCGAN[1] is used following the architecture constraints as suggested in [46]. For the conditional GAN the ACGAN model is used [41]. Both models will be further discussed below.

Deep Convolutional GAN (DCGAN)
For the DCGAN we use the suggested architecture by Radford et al. [46]. This is includes the architecture constraints they suggest and which were discussed in Section 2.4.

Auxiliary Classifier GAN (ACGAN)
The Auxiliary Classifier GAN (ACGAN) is a specific type of conditional GAN introduced by Odena et al. [41]. With minor changes to the architecture of the cDCGAN the ACGAN architecture managed to greatly stabilize training.

In the ACGAN the label information is not directly fed to the discriminator, but the discriminator is required to deduce it [41]. In that sense the discriminator will thus have an auxiliary decoder network that outputs the class label for the training data. The idea behind this approach is that forcing a model to perform additional tasks is shown to improve performance on the original task [47, 51, 52].

A schematic overview of the ACGAN can be seen in Figure 9. Especially note the differences between the conditional GAN as proposed by Mirza et al., and the ACGAN. The discriminator is not given any knowledge on the class label. Instead, it will output two probability distributions: that over the source (is it a real or fake image; $P(S|X)$) and that over the class labels ($P(C|X)$). The objective function of the ACGAN then also has two parts: the log-likelihood of the correct source, L_s, and the log-likelihood of the correct class, L_c.

[1] https://github.com/C0rine/InitialsGAN

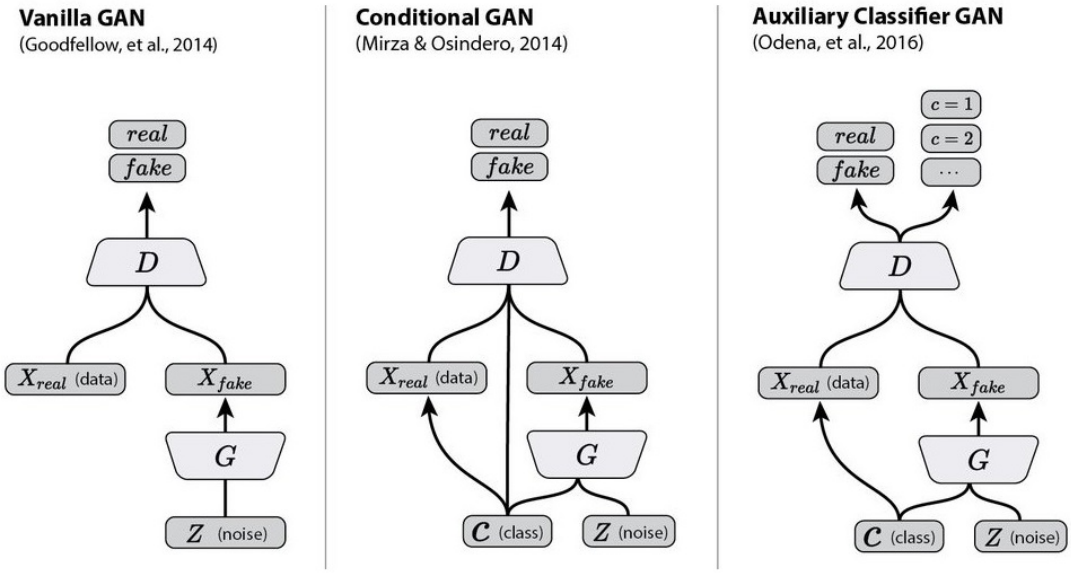


Figure 9: Schematic generalization several GAN architectures, with Z as random noise vector, X_{real} as images from the training set, X_{fake} as images generated by the generator and C as the set of classes for conditional generation. Image taken and adapted from [https://github.com/lukedeo/keras-acgan/blob/master/acgan-analysis.ipynb].

$$L_s = E[logP(S = real|X_{real})] + E[logP(S = fake|X_{fake})]$$
$$L_c = E[logP(C = C|X_{real})] + E[logP(C = c|X_{fake})]$$

Compared to the previously seen objective functions for GANs (section 2.2), the L_c term is new. In the ACGAN the discriminator and generator are in a minimax game over term L_s (which is similar to the $V(G, D)$ from section 2.2), but with the introduction of the L_c term the full objective for the networks will be as follows:

$$\max_D L_c + L_s$$
$$\max_G L_c - L_s$$

the intuition behind this can be explained as follows: the discriminator will want to be good at both predicting whether an image is fake or real and what type of class it was conditioned on. The generator will also want to make sure the discriminator recognizes the correct class, but does not want the discriminator to recognize the difference between real and fake images.

Model settings

Both models use the same hyper-parameter settings:

- Loss function: BCE loss
- Optimizer: Adam
- Batchsize: 32
- Learning rate: 0.0002
- Betas: 0.5, 0.999

An additional parameter for the ACGAN is the size of the latent space. After trying multiple values for the size of the latent space and evaluating the results empirically, we found that 500 worked best.

4.3 Evaluation methods

To evaluate the models described above we need some way to quantify performance. There are several pitfalls we need to look out for:

- A GAN might output very realistic looking images, but in reality is just reproducing images from the training set.
- The output of the GAN might lack diversity if it only learns to capture one mode of the dataset (mode-collapse).
- A GAN might create images that don't look like anything to humans, but can still fool the discriminator.

Additionally, for conditional GANs:

- We need to watch out for mode collapse per class. A conditional GAN that only outputs one type of image per class is also a bad performing GAN.
- A GAN might output a realistic looking image for a certain class but that does not necessarily mean that it looks like an image from the class it was conditioned on (e.g. condition training on the letter 'A', but we get a very realistic looking 'B').

We will try to recognize and quantify the susceptibility of our models to these pitfalls using following metrics: nearest neighbor comparison, inception score, SSIM and classification accuracy, each of which will be further explained below.

Nearest Neighbor Comparison

As stated above, it is important to note we do not only want a GAN to output realist looking images, we also want them to be 'new'. In other words: we do not want the model to copy images from the training set. Therefore most of the studies with new implementations of GANs, compare the generated images to their nearest

neighbor(s) from the training set (e.g. [26], [66]). The generated image should differ sufficiently from its nearest neighbor. We applied this type of evaluation to both the DCGAN and ACGAN. For our research, the three nearest neighbors are retrieved based on Euclidean distance.

Inception Score

The inception score was introduced in the paper on improved techniques for training GANs [48], but consequently used by multiple other authors for quantitative evaluation of the performance of GANs (e.g. [10], [35], [66] and [68]). It is said to correlate well with human judgement [48]. The inceptions score is calculated as follows [48]:

$$I = exp(\mathbb{E}_x KL(p(y|\mathbf{x})||p(y)))$$

Where x is one generated sample and y the label predicted by the inception model [53]. The KL stands for the Kullback-Leibler divergence. The intuition behind this metric is that good models should generate diverse but meaningful images [66]. 'Diverse' meaning that all classes are covered by the output of the GAN and 'meaningful' meaning that the output images can be easily classified (probabilities from the softmax show that the model is very certain about one class).

Usually the inception model is used to calculate the inception score [53]. This model is trained on the ImageNet dataset. When applying the inception model to other datasets it is usually only necessary to retrain the final layer of the model [48]. For this thesis however a different classifier ('Initials Baselines' developed by G. Strezoski) was used which was trained on the initials dataset from scratch.[2] The output of this classifier is softmaxed, after which the inception formula as suggested in [48] was used. We also follow the same experiment set-up as in [48], which includes the 50,000 output images which are evaluated in 10 splits.

SSIM

The Structural SIMilarity (SSIM) index is a metric originally used in image quality analysis [59]. It measure the difference between two images. In contrast however to metrics like MSE and PSNR, it is a perceptual similarity metric, meaning it discounts those aspects of the image that are not important for human perception [41].

The SSIM index is calculated on windows of an image. The formula to measure the distance between the windows x and y of same size (NxN) is as follows:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

With:

- μ_x the average of x
- μ_y the average of y
- σ_x^2 the variance of x
- σ_y^2 the variance of y
- $c_1 = (k_1 L)^2, c2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator, with:
  - L the dynamic range of the pixel values
  - $k_1 = 0.01$ and $k_2 = 0.03$ by default

The SSIM score ranges between -1 and 1, 1 meaning the images are identical, -1 meaning they are each-others inverse [60].

In this thesis the SSIM index is used to evaluate the inner-class diversity of the conditional GAN. As mentioned before, we do not want a conditional GAN which only outputs one type of image per class. The use of the metric follows that of the ACGAN paper [41]; for each class 100 random pairs of images are taken for which the SSIM is calculated. The SSIM index for that class is then the mean SSIM across the 100 pairs of images. The SSIM index for the whole conditional GAN is the the mean SSIM index across the all the classes. In short, we aim to get a low mean SSIM score across the classes, as this would indicate that, on average, the conditional GAN will output diverse images within the same class.

Classification Accuracy

The classification accuracy is applied to conditional GANs to see if the generated images conditioned on a certain class will also be classified as such. For example, if we try to generate the letter 'A', we will also want a well trained classifier to also classify it as such. The classification accuracy is measured on the same pre-trained classifier as the inception score.[2] The classification accuracy will calculate which percentage of 50,000 images generated by the cGAN are classified as the same class as the generation was conditioned on.

It is important to note for both the classification accuracy and the inception score that the classifier we use is not perfect. It performs well on the classification task of letters, but less on the other attributes:

Accuracy of the classifier on each of the attributes:

- Letters: 93.84%
- Countries: 68.34%
- Cities: 52.84%
- Names: 50.37%

The fact that the classifier does not perform very well on several of the attributes already indicates that generation task will probably also be difficult for those attributes.

5 RESULTS

In this sections the results for the two models as described in Section 4.2 will be discussed. An overview of the evaluation results can be found in Table 1, 2 and 3.

5.1 DCGAN results

For the unconditional GAN we followed the architecture constraints as suggested for DCGANs in [46]. Some additional experimentation was done to find the best model settings;

[2] https://github.com/gstrezoski/initials_baselines

Attribute	Incep. Mean	Incep. Std.
All	9.9996	0.1455

Table 1: Results of the evaluation of the DCGAN on the full dataset.

Attribute	SSIM	Incep. Mean	Incep. Std.	Class. Acc.
Letter	0.0759	8.1065	0.0528	38.08%
Country	0.1056	1.3457	0.0045	9.97%
City	0.0630	1.2467	0.0022	1.47%
Name	0.0475	2.4136	0.0071	0.25%

Table 2: Results of the evaluation of the ACGAN on the full dataset.

Attribute	SSIM	Incep. Mean	Incep. Std.	Class. Acc.
Country	0.1191	1.6764	0.0011	12.28%
City	0.0672	1.5702	0.0060	1.43%
Name	0.0589	2.7782	0.0011	0.25%

Table 3: Results of the evaluation of the ACGAN on the part of the dataset labeled as the letter 'S'.

- Training on this model was initially started with a batchsize of 128 and batchnorm (see Figure 17 in Appendix B.1). It was noticeable that the output was not very diverse so training was repeated with smaller batchsize of 32. This seemed to yield better results (see fig. 10).
- We've replaced the batch normalization layers of the model with instance normalization. As seen in appendix B.2 however, this significantly decreased the quality of the output.
- We replaced the BCE loss with the Wasserstein distance, but found that the performance of the GAN significantly decreased as well as can be seen in appendix B.4. The model would not properly converge.

The DCGAN architecture which eventually performed best was that with batch normalization, a batchsize of 32 and BCE loss. From an empirical analysis (looking at Figure 10), we see that this model is able to generate shapes that look like letters and generate noisy shapes around them which are reminiscent of the initial decorations as seen in the original dataset. In some cases we also see that the model learned to reproduce the lines of text surrounding the initials (as they were not in all cases cropped out during preprocessing of the training set).

As shown in Table 1, for the quantitative analysis of this model we could only calculate the inception score. In itself, these scores do not yet tell us much as we have no other models or papers which have worked with the same dataset. This inception score however can be compared to the inception score of the conditional GAN (ACGAN), which will be reported in the next section.

When looking at the nearest neighbors (based on Euclidean distance) of some of the output for this model we see that the images look quite similar in style, but different enough to be considered 'new' (see Figure 11).

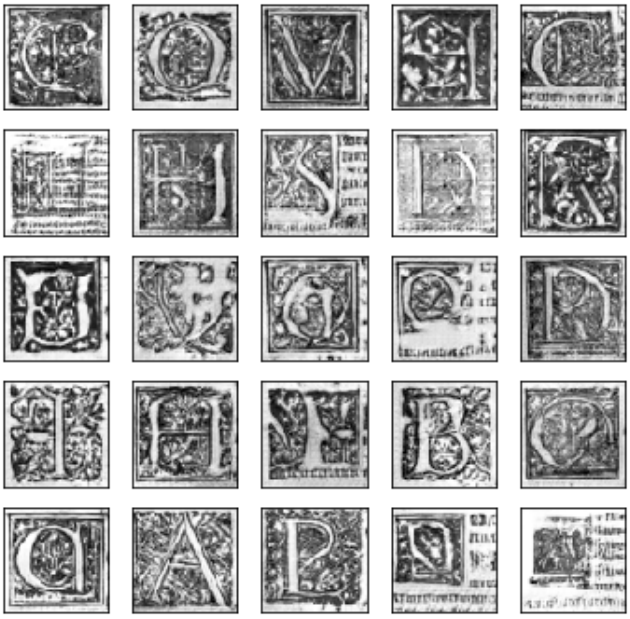


Figure 10: Some examples of DCGAN results with batchnorm, BCE loss and a batchsize of 32.

Figure 11: DCGAN results with their nearest neighbors. In each row, the rightmost image is an image generated by the DCGAN (batchsize 32, BCE loss), the consequent three images are the nearest neighbors from the initials dataset.

The results of this model gave us an idea of how well GANs could generate new images of decorative initials. It sets a certain baseline to compare our conditional results against. In the next step of our research we developed the conditional model; the ACGAN.

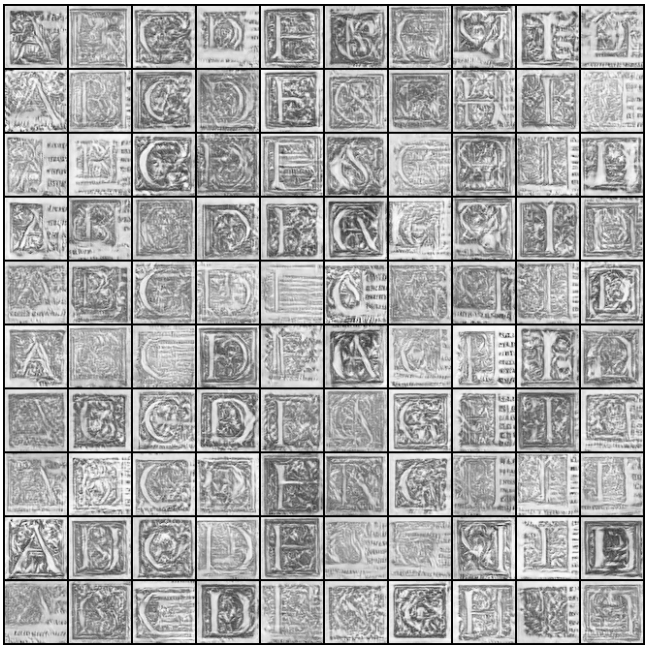


Figure 12: Some output examples for the ACGAN when trained on the full dataset. Each column shows images conditioned on a certain letter; from left to right A-J.

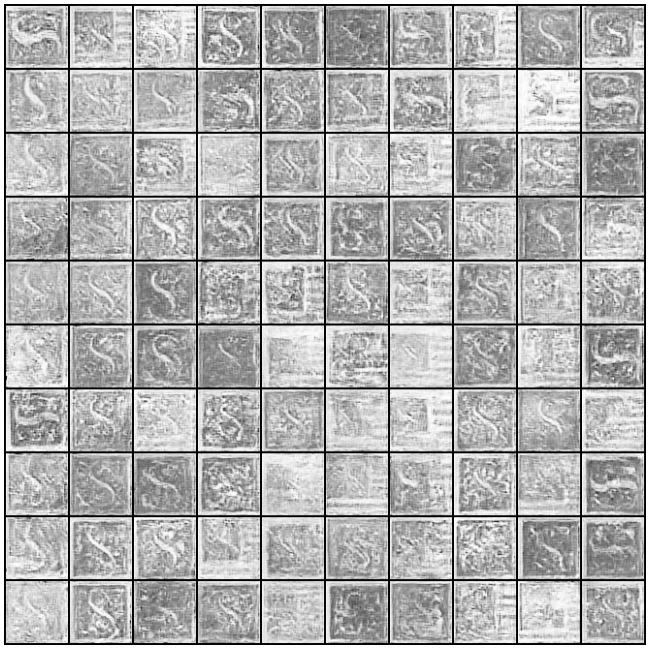


Figure 13: Some output examples for the ACGAN when trained on the 'S'-dataset. Each column shows images conditioned on a certain country.

5.2 ACGAN results

For the ACGAN we kept to the model architecture and model settings as suggested in the original ACGAN paper [41] and as discussed in Section 4.2.

We started our experimentation with the full dataset whilst conditioning the image generation on the 'letter'-attribute. Some results for this are shown in Figure 12. Most of the letters can be quite accurately generated, whilst the model seems to have more difficulty with others. Looking at the last column for example, for the letter 'J', we see that the model is not able to produce coherent samples. This is likely due to the fact that in the whole initials dataset there are only 3 images labeled as 'J' (compared to 2415 for the letter 'A' for example, see Appendix A). So there probably is just not enough training data available for the model to learn what the letter 'J' looks like.

We continued our experimentation for the other attributes (country, city and names), the results of which can be viewed in Appendix C.1. As we are comparing different letters in these images, it is hard to empirically determine if the ACGAN outputs initials with a different style for each country/city/name. It is for that reason we make use of the 'S'-dataset. By fixating the letter we can better determine if there is a difference is style between images that were for example conditioned on different countries.

Some results for the generation of images conditioned on country with the 'S'-dataset can be viewed in Figure 13). In this image we see that the model has learned the general shape of the letter 'S' and that it has found some variations on that shape. We however also see that the output for some countries look very similar and that these images look a lot more noisy than the images that were conditioned on the 'letter' attribute. The output for the ACGAN trained on the 'S'-dataset for the other attributes can be found in Appendix C.2.

An overview of evaluation results for the output of the ACGAN when trained on the full dataset can be viewed in Table 2. For the 'S'-dataset these can be viewed in Table 3. There are several interesting points to note in these tables:

- What is interesting to see in the SSIM column is that there apparently is more similarity between the images conditioned the same country than between the images conditioned on the same city. The SSIM index for images that are conditioned on the same printer (name) is even lower. This is the case both when trained on the full dataset and when trained on the 'S'-dataset. This is counter-intuitive as one would expect that there is more stylistic (and therefore visual) similarity between images made by the same artist, than images made in the same country. We have several ideas on reasons for these counter-intuitive SSIM scores:
  - Stylistic similarity might not be properly captured by the SSIM index, as it is originally also not developed for this purpose [59].
  - We see that the classification accuracy for these attributes is also quite low. This might indicate that the ACGAN outputs very noisy samples and noisy samples just tend to get a lower SSIM index.
  - Some classes (especially within the 'names' attribute) only had a few images from the training set attributed to them (see Appendix A). The images for such classes might all be very similar. The generator could have picked up on

	Classifier	Random
Letter	38.08%	3.85%
Country	9.97%	9.09%
City	1.47%	1.54%
Name	0.25%	0.27%

Table 4: Comparison of the classification accuracy of the classifier versus random classification of the output of the ACGAN when trained on the full initials dataset.

	Classifier	Random
Country	12.28%	9.09%
City	1.43%	1.54%
Name	0.25%	0.27%

Table 5: Comparison of the classification accuracy of the classifier versus random classification of the output of the ACGAN when trained on the full initials dataset.

this similarity, outputting very similar images for such classes as well. These generated images would then boost the SSIM a lot as they hardly differ from each other.

- For the inception score we note that the DCGAN and the ACGAN conditioned on the 'letter' attribute perform the best, which concurs with our empirical findings (also see Figure 11 and 12). What again seems counter-intuitive however is that the 'name' attribute scores better than the 'country' and 'city' attributes. This is again both the case for the full dataset and the 'S'-dataset. We do not have a good explanation for this as yet.
- The classification accuracy scores are compared to the accuracy we would get when we assign each image a random class; e.g. random classification of images to a certain letter would yield an accuracy of 100/26 (letters/classes) $\approx$ 3.85%. The classification accuracies versus random classification can be viewed in Table 4 and 5.
  In Table 4 we see that the classification accuracy is significantly better than random classification at the 'letter' attribute. We also see here that the classification accuracy on the 'country' attribute is better when trained on the 'S'-dataset, than when trained on the full initials dataset and in both cases is a bit higher than random classification. This could be an indication that it did help to separate one letter from the dataset to better generate the stylistic differences between countries.
  In both Table 4 and 5 we however see that classification accuracy for the 'city' and 'name' attribute actually score below random classification. From this we can deduce that these attributes are probably the most difficult to perform conditional generation on.

Figure 14 shows some random images generated by the ACGAN when trained on the full dataset for the attribute 'letter'. In each row, the first image is an image generated by the ACGAN, the consequent three images are the nearest neighbors from the dataset. We can deduce from this image that the ACGAN does not simply reproduce images from the training set. It is however notable that the nearest neighbors for each of the images outputted by the GAN not often contain the same letter as that the generation was conditioned on. Nearest neighbor results for the other models can be viewed in Appendix C.

Figure 14: Some of the results of the ACGAN when trained on the full dataset for the attribute 'letters'. In each row, the rightmost image is an image conditionally generated by the ACGAN (from top to bottom: 'A', 'B', 'C' and 'D'), the consequent three images are the nearest neighbors from the initials dataset.

6 DISCUSSION AND FUTURE WORK

In the introduction we formulated our research question as follows: are GANs capable of generating (semi-)realistic images of decorative initials based on certain attributes, whilst capturing the stylistic characteristics related to those attributes? We trained two GAN models on a dataset of images of decorative initials (annotated with: depicted letter, country of creation, city of creation and name of the printer) to answer this question: the unconditional DCGAN and the conditional ACGAN.

For both the unconditional DCGAN as the conditional ACGAN we have seen that it is possible to create semi-realistic looking images of initials when focusing on the letters. The GANs are in both cases able to pick up on the general shapes of letters and to a lesser extent the decorations around it. This is reflected as well in the inception scores and classification accuracies (see Tables 1, 2 and 3). We however also see that when we train the conditional ACGAN on countries, cities or names that the evaluation scores are a lot lower. This corresponds to our expectations as letters are indeed also easier for humans and classifiers (see Section 4.2) to detect and recognize than the country/city of creation or the name of the printer. The conclusion is therefore that the GANs are capable of generating semi-realistic images of decorative initials

when the generation is not conditioned, or when the generation is conditioned on the depicted letter.

For future work we would therefore suggest that more research can be done into specifically generating the decorative elements of the initials as this seems to be the more difficult task. Other, more modern, GAN architectures could be applied to the dataset. Stacked GANs especially seem promising for this dataset as the generation process could then be broken down into multiple smaller and more simple steps. We also think that the training of GANs for this dataset would benefit from a bigger training set as some of the classes a severely underrepresented in the dataset (see Appendix A). In general however, we also feel that more research needs to be done with regards to coherent evaluation metrics for GANs, as the current metrics are hard to interpret, especially when they cannot be compared to other models.

REFERENCES

[1] [n. d.]. Large-scale CelebFaces Attributes (CelebA) Dataset. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. ([n. d.]). (Accessed on 05/12/2018).

[2] Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862* (2017).

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

[4] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. 2017. Generalization and Equilibrium in Generative Adversarial Nets (GANs). *CoRR* abs/1703.00573 (2017). arXiv:1703.00573 http://arxiv.org/abs/1703.00573

[5] Samaneh Azadi, Matthew Fisher, Vladimir G. Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. 2017. Multi-Content GAN for Few-Shot Font Style Transfer. *CoRR* abs/1712.00516 (2017). arXiv:1712.00516 http://arxiv.org/abs/1712.00516

[6] Erik Bernhardsson. [n. d.]. Analyzing 50k fonts using deep neural networks - Erik Bernhardsson. https://erikbern.com/2016/01/21/analyzing-50k-fonts-using-deep-neural-networks.html. ([n. d.]). (Accessed on 01/25/2018).

[7] Ankan Kumar Bhunia, Ayan Kumar Bhunia, Prithaj Banerjee, Aishik Konwer, Abir Bhowmick, Partha Pratim Roy, and Umapada Pal. 2018. Word Level Font-to-Font Image Translation using Convolutional Recurrent Generative Adversarial Networks. *CoRR* abs/1801.07156 (2018). arXiv:1801.07156 http://arxiv.org/abs/1801.07156

[8] Charles Bigelow. 1999. Robert Bringhurst, The elements of typographic style. *Written Language & Literacy* 2, 1 (1999).

[9] George Chalmers, Robert D. Murray, Thomas Riches, and John Sturt. 1701. *The Holy Bible, containing the Old Testament and the New : newly translated out of the original tongues, and with the former translations diligently compared and revised : by His Majesties special command : appointed to be read in churches.].* Printed by Charles Bill and the Executrix of Thomas Newcomb.

[10] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. 2016. Mode Regularized Generative Adversarial Networks. *CoRR* abs/1612.02136 (2016). arXiv:1612.02136 http://arxiv.org/abs/1612.02136

[11] Xi Chen, Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 2172–2180.

[12] William George Constable. 1938. *Art history and connoisseurship: their scope and method.* CUP Archive.

[13] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. 2017. Generative Adversarial Networks: An Overview. *CoRR* abs/1710.07035 (2017). arXiv:1710.07035 http://arxiv.org/abs/1710.07035

[14] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. 2017. Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. *Computing and Software for Big Science* 1, 1 (2017), 4.

[15] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. 2015. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. *CoRR* abs/1506.05751 (2015). arXiv:1506.05751 http://arxiv.org/abs/1506.05751

[16] C. Doersch. 2016. Tutorial on Variational Autoencoders. *ArXiv e-prints* (June 2016). arXiv:stat.ML/1606.05908

[17] Max J Friedlander. 2013. *On art and connoisseurship.* Read Books Ltd.

[18] Philip Gaskell and Ronald Brunlees McKerrow. 1972. *A new introduction to bibliography.* Clarendon Press Oxford.

[19] Jon Gauthier. 2014. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester* 2014, 5 (2014), 2.

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

[21] Ian J. Goodfellow. 2017. NIPS 2016 Tutorial: Generative Adversarial Networks. *CoRR* abs/1701.00160 (2017). arXiv:1701.00160 http://arxiv.org/abs/1701.00160

[22] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5767–5777. http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf

[23] Michael Ann Holly. 1985. *Panofsky and the foundations of art history.* Cornell University Press.

[24] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. On Unifying Deep Generative Models. *CoRR* abs/1706.00550 (2017). arXiv:1706.00550 http://arxiv.org/abs/1706.00550

[25] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *CoRR* abs/1611.07004 (2016). arXiv:1611.07004 http://arxiv.org/abs/1611.07004

[26] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. 2016. Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts. *CoRR* abs/1612.00215 (2016). arXiv:1612.00215 http://arxiv.org/abs/1612.00215

[27] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. *CoRR* abs/1703.05192 (2017). arXiv:1703.05192 http://arxiv.org/abs/1703.05192

[28] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[29] Naveen Kodali, Jacob D. Abernethy, James Hays, and Zsolt Kira. 2017. How to Train Your DRAGAN. *CoRR* abs/1705.07215 (2017). arXiv:1705.07215 http://arxiv.org/abs/1705.07215

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[31] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2016. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint* (2016).

[32] Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory* 37, 1 (1991), 145–151.

[33] Yifan Liu, Zengchang Qin, Zhenbo Luo, and Hua Wang. 2017. Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks. *CoRR* abs/1705.01908 (2017). arXiv:1705.01908 http://arxiv.org/abs/1705.01908

[34] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. 2017. Deep photo style transfer. *CoRR, abs/1703.07511* (2017).

[35] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2813–2821.

[36] Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. 2016. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 5040–5048.

[37] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *CoRR* abs/1411.1784 (2014). arXiv:1411.1784 http://arxiv.org/abs/1411.1784

[38] Aiden Nibali. 2016. The GAN objective, from practice to theory and back again. http://aiden.nibali.org/blog/2016-12-21-gan-objective/. (2016). (Accessed on 05/22/2018).

[39] Dong Nie, Roger Trullo, Jun Lian, Caroline Petitjean, Su Ruan, Qian Wang, and Dinggang Shen. 2017. Medical image synthesis with context-aware generative adversarial networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 417–425.

[40] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 271–279.

[41] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2016. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585* (2016).

[42] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759* (2016).

[43] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. 2018. CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. *Phys. Rev. D* 97 (Jan 2018), 014021. Issue 1. https://doi.org/10.1103/PhysRevD.97.014021

[44] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2536–2544.

[45] Ben Poole, Alexander A. Alemi, Jascha Sohl-Dickstein, and Anelia Angelova. 2016. Improved generator objectives for GANs. *CoRR* abs/1612.02780 (2016). arXiv:1612.02780 http://arxiv.org/abs/1612.02780

[46] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR* abs/1511.06434 (2015). arXiv:1511.06434 http://arxiv.org/abs/1511.06434

[47] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. 2015. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072* (2015).

[48] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. 2016. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 2234–2242. http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf

[49] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. 2017. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*. Springer, 146–157.

[50] Paul Smolensky. 1986. *Information processing in dynamical systems: Foundations of harmony theory*. Technical Report. COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE.

[51] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[52] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[53] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *CoRR* abs/1512.00567 (2015). arXiv:1512.00567 http://arxiv.org/abs/1512.00567

[54] Lucas Theis, Aäron van den Oord, and Matthias Bethge. 2015. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844* (2015).

[55] Yuchen Tian. [n. d.]. zi2zi: Master Chinese Calligraphy with Conditional Adversarial Networks. https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html. ([n. d.]). (Accessed on 02/21/2018).

[56] TJTorres. [n. d.]. A Fontastic Voyage: Generative Fonts with Adversarial Networks | Stitch Fix Technology - Multithreaded. https://multithreaded.stitchfix.com/blog/2016/02/02/a-fontastic-voyage/. ([n. d.]). (Accessed on 01/25/2018).

[57] Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann SIMON-GABRIEL, and Bernhard Schölkopf. 2017. AdaGAN: Boosting Generative Models. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5424–5433. http://papers.nips.cc/paper/7126-adagan-boosting-generative-models.pdf

[58] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. 2016. Generating Videos with Scene Dynamics. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 613–621. http://papers.nips.cc/paper/6194-generating-videos-with-scene-dynamics.pdf

[59] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (April 2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

[60] Zhou Wang, Alan C Bovik, and Eero P Simoncelli. 2005. Structural approaches to image quality assessment. *Handbook of Image and Video Processing* 7 (2005), 18.

[61] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. 2017. Gp-gan: Towards realistic high-resolution image blending. *arXiv preprint arXiv:1703.07195* (2017).

[62] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. 2017. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIRâĂŹ2017), Suzhou, China*.

[63] Qingsong Yang, Pingkun Yan, Yanbo Zhang, Hengyong Yu, Yongyi Shi, Xuanqin Mou, Mannudeep K Kalra, Yi Zhang, Ling Sun, and Ge Wang. 2018. Low dose CT image denoising using a generative adversarial network with wasserstein distance and perceptual loss. *IEEE Transactions on Medical Imaging* (2018).

[64] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. 2017. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5485–5493.

[65] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.. In *AAAI*. 2852–2858.

[66] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. 2016. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. *CoRR* abs/1612.03242 (2016). arXiv:1612.03242 http://arxiv.org/abs/1612.03242

[67] Lvmin Zhang, Yi Ji, and Xin Lin. 2017. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier GAN. *arXiv preprint arXiv:1706.03319* (2017).

[68] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. 2016. Energy-based Generative Adversarial Network. *CoRR* abs/1609.03126 (2016). arXiv:1609.03126 http://arxiv.org/abs/1609.03126

[69] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593* (2017).

7 APPENDIX

A DATASET

Letter	No. samples
A	2415
B	580
C	2141
D	1968
E	2090
F	888
G	527
H	1517
I	2417
J	3
K	83
L	1142
M	1326
N	1543
O	1100
P	2180
Q	2198
R	755
S	2564
T	1249
U	5
V	1653
W	115
X	48
Y	25
Z	118

Table 6: Amount of images per 'letter' class of the initials dataset.

Country	No. samples
FR	11393
CH	6140
DE	4457
ES	211
IT	7075
BE	849
GB	202
POR	59
NL	188
DK	45
CZ	31

Table 7: Amount of images per 'country' class of the initials dataset.

City	No. samples
Lyon	5623
Paris	5102
Basel	4921
Zwickau	10
Valencia	29
Frankfurt	1132
Brescia	169
Madrid	60
Antwerpen	815
Venezia	5115
Firenze	528
Oppenheim	64
Munchen	76
Magdenburg	2
Strasbourg	430
London	202
Sevilla	80
Koln	2036
Louvain	32
Zurich	656
Worms	43
Bologna	908
Ingoldstadt	63
Ferrara	86
Marburg	46
Geneve	563
Pavia	23
Bergamo	49
Neustadt	17
Mainz	125
Heidelberg	63
Hagenau	108
Leipzig	135
Jena	51
Augsburg	95
Wesel	275
Macerata	58
Como	30
Nuremberg	176
Douai	20
Vincenza	35
Alcala	1
Wittenberg	15
Dortmund	2
Morges	18
Lucca	13
Lisboa	59

Continued on next page...

City	No. samples
Dillingen	2
Rome	36
Alkmaar	12
Amsterdam	150
Hamburg	14
Leiden	26
Ieper	2
Kopenhagen	45
Freiburg	2
Zaragozza	9
Troyes	51
Salamanca	32
Erfurt	2
Poitiers	21
Wurzburg	11
Genova	25
Caen	20
Praha	31

Table 8: Amount of images per 'city' class of the initials dataset.

Name	No. samples
Simon Bevilaqua	78
Matthieu David	14
Valentin Curio	52
Gabriel Kantz	10
Diego Gumiel	29
Jean Pillehotte	53
Andreas Cratander	294
Sebastien Cramoisy	23
Seitz Heirs Peter Main	44
Andre Bocard	98
Comino Presegni	102
Pedro Madrigal	23
Oudin Petit	29
Willem Vorsterman	23
Gaspare 2 Bindoni	105
Matthias David	14
Pre Galliot Du	46
Philippo 2 Giunta	60
Jakob Kobel	64
Adam Berg	76
Philippo 1 Giunta	236
Antoine Vincent	83
Wolfgang Kirchener	1
Gabriel Buon	22
Johann Knobloch	32
Guillaume Cavellat	80
Gilles Gourbin	68

Continued on next column...

Name	No. samples
Vincenzo Sabbio	42
Becker Matthias Main	78
Robert Barker	108
Bonetus Locatellus	175
Jacob Cromberger	80
Bartholomaeus Vincent	91
Mylius Crato	62
Antoine Augerelle	42
Robert Chaudiere	64
Hero Fuchs	75
Mace Bonhomme	64
Lucantonio 1 Giunta	168
Charlotte Guillard	69
Lucantonio 2 Giunta	264
Claude Servain	30
Peter Quentell	36
Jacques Huguetan	154
Arnold Heirs Birckman	252
Dirck Martensz	6
Giovanni Battista Ciotti	210
Bernardino Gerrualda	72
Claude Chevallon	72
Andreas Gessner	89
Simon Beys	28
Peter Schoeffer	55
Johann Oporinus	376
Heinrich Petri	579
Sebastien Honorat	18
Giovanni Rosso	298
Girolamo Scoto	406
Robert Winter	129
Bartollomeo Carampello	50
Alexander Weissenhorn	9
Vittorio Baldini	8
Candido Benedetto	1
Simon Colines	378
Adam Petri	84
Jean Roigny	169
Jacques Sacon	226
Christian Egenolff	46
Sebastian Henricpetri	310
Giovanni Maria Bonelli	247
Johann Soter	115
Eustache Vignon	61
Giacomo Pocatello	23
Antoine Ry	56
Ludwig Ostenius	52
Jamet Mettayer	69
Jean Louis	39
Jean Tournes	224

Continued on next page...

Name	No. samples
Feyerabend Johann Main	37
Joannes Moylin	72
Etienne Gueynard	375
Gerwin Calenius	117
Francesco Rampazetto	123
Comino Ventura	49
Johann Schott	35
Pierre Marechal	68
Wilhelm Harnisch	17
Claude Fremy	21
Sebastien Griffo	123
Gregorio Gregoriis	408
Josse Bade	586
Lorenzo Torrentino	232
Francesco Franceschi	589
Franz Behem	73
Melchior Neuss	445
Francesco Rossi	11
Cyriacus Jacob Main	30
Andreas Cambier	44
Domenico Farri	108
Ivo Schoeffer	42
Heinrich Gran	13
Michael Blum	81
Heirs Thomas Rebart	51
Peter Perna	133
Melchior Lotter	27
Ludwig Alectorius	219
David Sartorius	44
Wechel Heirs Andreas Main	60
Paganinus Paganinis	141
Ernhard Ratdolt	192
Johann Walder	141
Heinrich Stayner	84
Porte Hugues La	75
Hans Braker	24
Johann Miller	11
G M Beringen	32
Johann Feyerabend	50
Thielman Kerver	39
Ulrich Gering	51
Arnold Birckman	73
Vincent Portonariis	191
Robert Estienne	305
Aldus 1 Manutius	23
Jaques Marechal	63
Braubach Peter Main	444
Nicolas Benedictis	93
Valentin Kobian	25
Sebastiano Martellini	58
Prez Nicolas Des	24
Jerome Olivier	53
Francois Regnault	114

Continued on next column...

Name	No. samples
Michael Sonnius	299
Bartolommeo Zanis	57
Pierre Fradin	86
Johann Herwagen	640
Egenolff Christian Main	62
Arnoul Angelier	87
Laurent Sonnius	117
Claude Senneton	68
Thomas Anshelm	67
Theodore Rihel	45
Andre Wechel	70
Guillaume Chaudiere	13
Symphorien Barbier	18
Melchior Sessa	218
Joannes Crispinus	62
Christian Wechel	63
Johan Bebel	215
Antoine Harsy	81
Eucharius Cervicorius	260
Lechler Martin Main	42
Anselmo Giacarelli	91
Jean le Preux	80
Michael Hillenius	1
Porte Sybille La	31
Gottardo da Ponte	30
Berthold Rembold	245
Johann von Berg	137
Blaise Guido	39
GB Bellagamba	374
Willem Sylvius	8
Georg Hantsch	14
Damiano Zenaro	116
Michel Vascosan	211
Jan Loe	91
Adrien Tunebre	18
Giovanni Battista Phaelli	54
Conrad Bade	40
Jeune Martin Le	40
Allesandro Benacci	27
Jan 1 Roelants	98
Jean Bogard	20
Nicolaus Episcopius	103
Vincent Valgrisi	52
George Bishop	25
Gilbert Villiers	60
Johann Quentell	66
Theobaldus Ancelin	23
Richter Wolfgang Main	233
Noir Guillaume Le	22
Michael Isengrin	348
Guillaume Rouille	950
Jean Clein	217
Joannes Platea	59

Continued on next page...

Name	No. samples
Pierre Vidoue	40
Sebastien Nivelle	69
Louis Cynaeus	18
Jean Le Rouge	3
Christophorus Zelle	1
Giorgio Greco	35
Arnaldo Guillen Brocar	1
Hieronymus Verdussen	113
Laurentius Annison	25
Jakob Kundig	55
Guillaume Guillemot	34
Christoph Rodinger	1
Symon Cock	136
Claude Davost	57
Charles Estienne	45
Jan 1 Moretus	59
Johann Schwertel	14
Francesco Bolzetta	19
Fevre Francois Le	123
Melchior Soter	2
Denys Janot	5
Johann Schoeffer	4
Guillaume Lairmarius	18
Venturina Rosselini	2
Eichorn Andreas Oder	5
Jacques Faure	56
Barthelemy Ancelin	39
Vincenzo Busdraghi	13
Martin LEmpereur	32
Johann Neuss	54
Hartmann Friedrich Oder	2
Pierre Galterus	3
Hieronymus Wallaeus	1
Rutgerus Velpius	1
Levinus Hulsius	35
de Ferrara Gabriele Giolito	206
Antonio Padovani	43
Jean Cavelleir	41
Bassaeus Oder	25
Johann Faber	3
Ambrosius Froben	35
Guichard Julieron	3
Hieronymus Commelin	19
Antoine Blanchard	31
Heirs Symphorien Beraud	33
Jacques Myt	144
Godefriedus Kempen	6
Puys Jacques Du	12
Andrea Arrivabene	121
Heinrich Quentell	1
Froschauer Christopher 1 CH	360
Miguel Eguia	37
Pierre Roussin	1

Continued on next column...

Name	No. samples
Caspar Behem	6
Luis Rodrigues	59
Henry Estienne	106
Samuel Konig	20
Guglielmo Fontaneto	144
Giovanni Guerigli	19
Mathieu Berjon	17
Anton Hierat	5
Sebald Mayer	2
Johann Prael	32
Widow Gabriel Buon	124
Heinrich Gymnicus	19
Thomas Courteau	30
Symphorien Beraud	33
Bonaventura Nugo	229
Joannes Criegher	52
Jan van Keerbergen	10
Bartolomeo Bonfadini	36
Petrus Colinaeus	18
Heirs Sebastien Griffo	5
Francois Fradin	248
Guillaume Morel	53
Hieronymus Froben	454
Widow Martinus Nutius	16
Johann Froben	678
Nicolaus Brylinger	144
Vivant Gautherot	86
Widow Hendrick Peetersen	48
Joachim Trognesius	8
Nicolaus Faber	12
Jean Petit	323
Thibaud Payen	23
Johann Setzer	3
Jean Crespin	131
Norton Johann Main	22
Jacques Androuet	6
Giovanni Bariletto	17
Hans Luft	251
Jean Ogerolles	73
Andrea Poleti	9
Jacob Meester	12
Christoffel Cunradus	70
van Waesberghe Joannes Janssonius	80
Maternus Cholinus	66
Thomas Raynald	39
Johann Ruremundensis	1
Jean Laon	3
Giorgio Angelieri	103
Andreas Angermaier	10
Jean Gerard	43
Martinus Verhasselt	3
Federic Morel	27
Elisabetta Rusconi	21

Continued on next page...

Name	No. samples
Georg Papen	1
Jean Dalier	15
Jean Bienne	45
Heirs Hieronymus Benedictis	1
Barezzo Barezzi	200
Thomas Wolf	10
Martinus Gymnicus	1
Etienne Dolet	11
Francesco Suzzi	67
Froschauer Christopher 2 CH	136
Clerc David Le	3
Konrad Mechel	3
Christoffel Guyot	15
Arnout Brakel	6
Marcus Zaltieri	80
Joannes Degaram	9
Joannes Masius	3
Henri Estienne	167
Horace Cardon	101
Hadrianus Perier	77
Joos Destree	2
Peter Seitz	1
Benoit Prevost	68
Johann Gruninger	256
Mats Vingaard	45
Konrad Caesarius	9
Jean Marion	156
Michel Cotier	6
Jacob Roussin	110
Breisgau Emmeus Johann im	2
Pierre Mettayer	3
Francisco Baba	3
Guillaume Julianus	52
Johann Birckman	1
Catharina Gerlach	3
Pedro Bernuz	9
Rouge Nicolas Le	51
Jacob Stoer	28
Guillaume Foquel	32
Konrad Waldkirch	46
Paul Frellon	42
Fredericus Lydius	2
Bartollomeo Alberti	91
Pasqier Le Tellier	3
Joannes Grapheus	37
Guillaume Lairmarie	38
Paulus Queckus	4
Hendrik Connix	18
Raben Georg Main	48
Johann Gymnicus	52
Pietro Maria Marchetti	25
Wolfgang Sthurmer	2
Philippe Tinghi	46
Pamphilius Gengenbach	1
Johann Schoenstenius	1
Jasper Gennep	85
Herman Moller	13
Pierre Gautier	14
Bernardino Vitalis	44
Jean Blanchet	21
Thomas Brumennius	11
Guillaume Merlin	54
Bartholomaeus Westheimer	9
Elzeviers	11
Heinrich von Aich	11
Georg Defner	1
Antoine Chuppin	17
Johann Gemusaeus	1
Antonio Bellona	25
Girard Angier	20
Giovanni 1 Griffio	20
Ottavio Scoto	137
Francois Arnoullet	12
Heirs Johann Quentell	4
Robert Field	30
Andreas2 en HJ Gesner	71
Wilhelm Lutzenkirchen	3
Bolognino Zaltieri	98
Georg Schwartz	31
Christoffel Plantin	99
Gilles Huguetan	2
Baldassare Constantini	2
Gerard Morrhe	9
Jacques Giunta	110
Bocchiana Nova Academia	1

Table 9: Amount of images per 'name' class of the initials dataset.

B DCGAN

B.1 Batch Normalization with batch size of 32

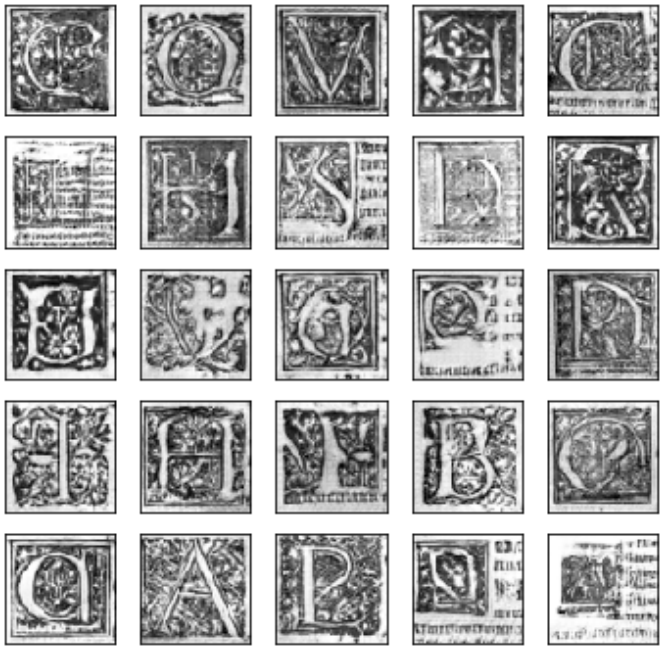


Figure 15: Some DCGAN results for Batchnorm, BCE loss and a batch size of 32

Figure 16: Nearest neighbors for the DCGAN with Batchnorm, BCE loss, batch size 32. In each row, the rightmost image is an image generated by the GAN and the consequent three images their nearest neighbors from the initials dataset.

B.2 Batch Normalization with batchsize of 128

Figure 17: Some DCGAN results for Batchnorm, BCE loss and a batch size of 128

Figure 18: Nearest neighbors for the DCGAN with Batchnorm, BCE loss, batch size 128. In each row, the rightmost image is an image generated by the GAN and the consequent three images their nearest neighbors from the initials dataset.

B.3 Instance Normalization

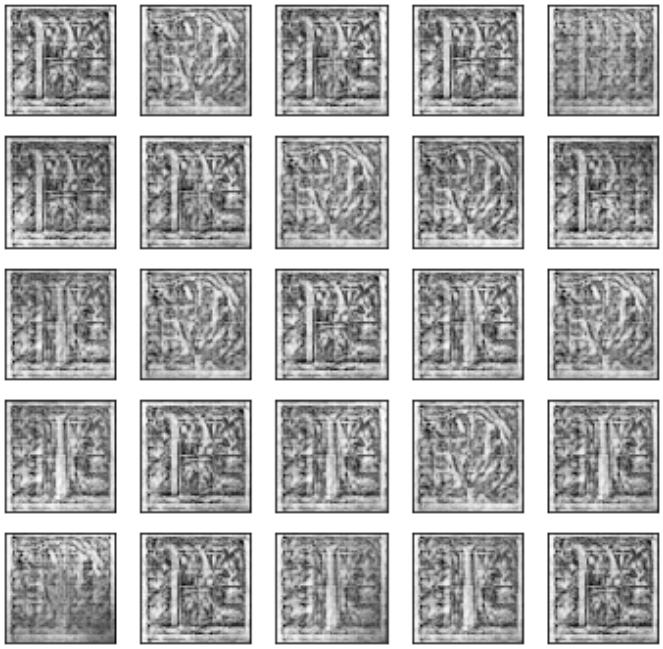


Figure 19: Some DCGAN results for Instancenorm, BCE loss and a batch size of 32

Figure 20: Nearest neighbors for the DCGAN with Instancenorm, BCE loss, batch size 32. In each row, the rightmost image is an image generated by the GAN and the consequent three images their nearest neighbors from the initials dataset.

B.4 Wasserstein Distance

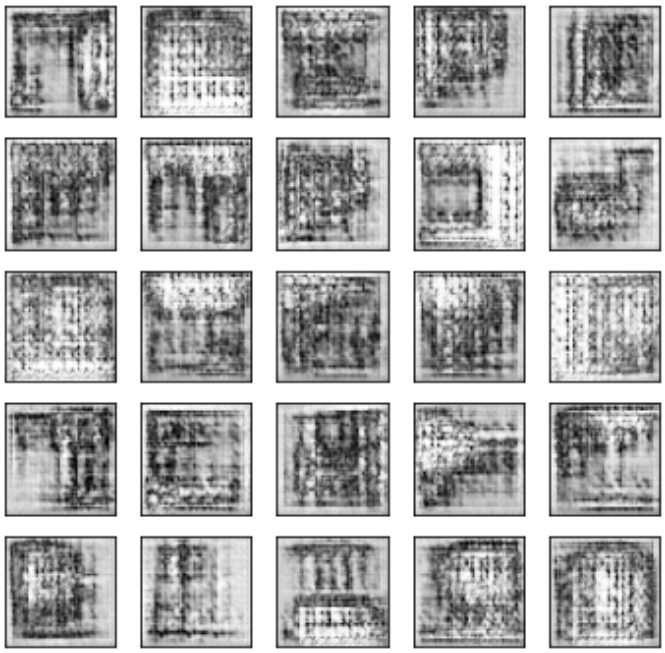


Figure 21: Some Wasserstein DCGAN results with Batchnorm, BCE loss and a batch size of 32

Figure 22: Nearest neighbors for the Wasserstein DCGAN with Batchnorm, BCE loss, batch size 32. In each row, the rightmost image is an image generated by the GAN and the consequent three images their nearest neighbors from the initials dataset.

C ACGAN

C.1 Full dataset

C.1.1 For 'letter' attribute:

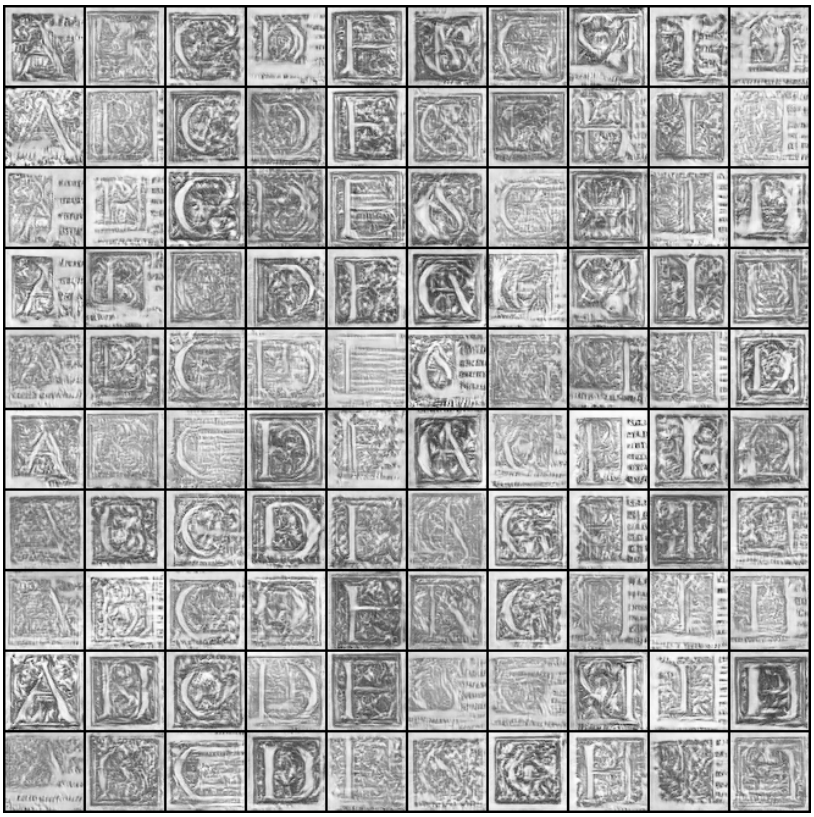


Figure 23: ACGAN results when trained on the full dataset for the 'letter'-attribute. Each column corresponds to a value for the attribute, from left to right: A, B, C, D, E, F, G, H, I, J.

Figure 24: In each row, the rightmost image is conditionally generated by the ACGAN (trained on the full dataset) on the 'letter' attribute. The consequent three images are the nearest neighbors from the initials dataset.

C.1.2 For 'country' attribute.

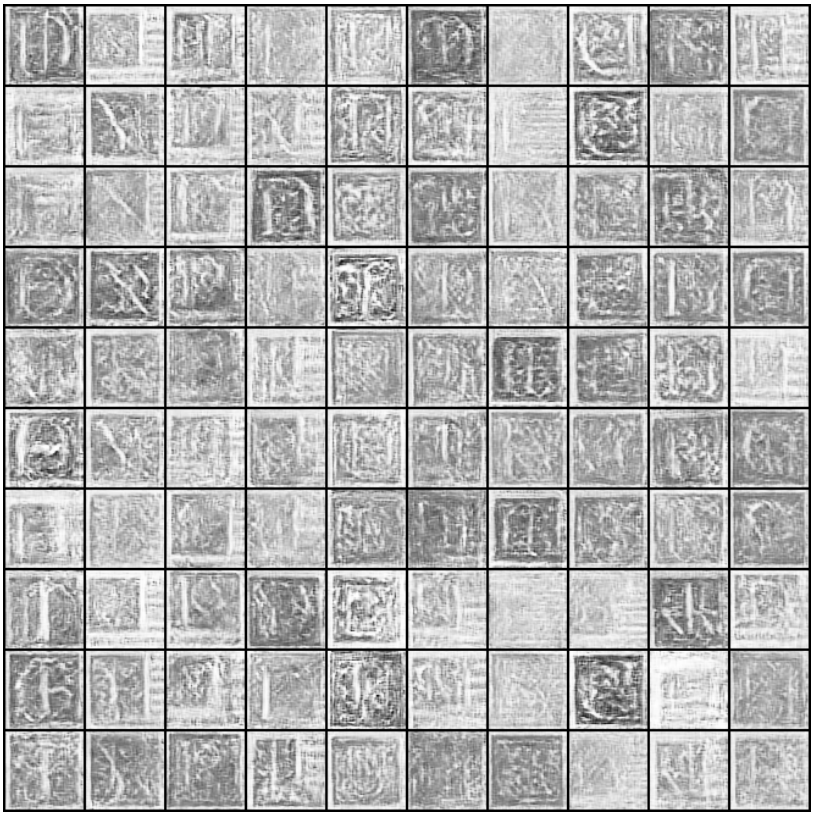


Figure 25: ACGAN results when trained on the full dataset for the 'country'-attribute. Each column corresponds to a value for the attribute, from left to right: FR, CH, DE, ES, IT, BE, GB, POR, NL, DK.

Figure 26: In each row, the rightmost image is conditionally generated by the ACGAN (trained on the full dataset) on the 'country' attribute. The consequent three images are the nearest neighbors from the initials dataset.

C.1.3 For 'city' attribute.

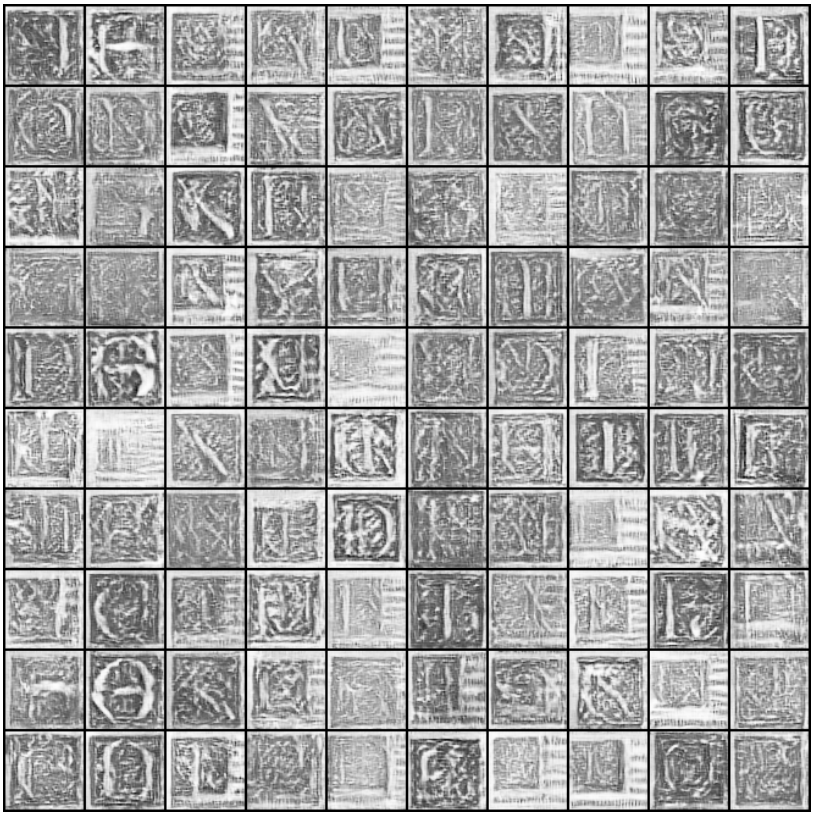


Figure 27: ACGAN results when trained on the full dataset for the 'city'-attribute. Each column corresponds to a value for the attribute, from left to right: Lyon, Paris, Basel, Zwickau, Valencia, Frankfurt, Brescia, Madrid, Antwerpen, Venezia.

Figure 28: In each row, the rightmost image is conditionally generated by the ACGAN (trained on the full dataset) on the 'city' attribute. The consequent three images are the nearest neighbors from the initials dataset.

C.1.4 For 'name' attribute.

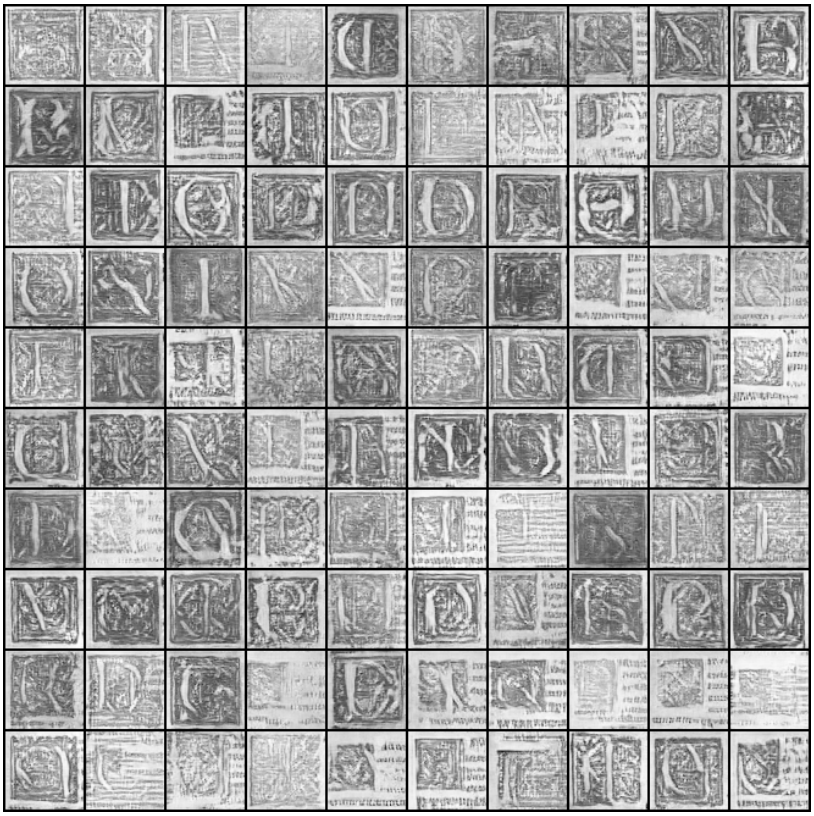


Figure 29: ACGAN results when trained on the full dataset for the 'name'-attribute. Each column corresponds to a value for the attribute, from left to right: Simon Bevilaqua, Matthieu David, Valentin Curio, Gabriel Kantz, Diego Gumiel, Jean Pillehotte, Andreas Cratander, Sebastien Cramoisy, Seitz Heirs Peter Main, Andre Bocard.

Figure 30: In each row, the rightmost image is conditionally generated by the ACGAN (trained on the full dataset) on the 'name' attribute. The consequent three images are the nearest neighbors from the initials dataset.

C.2 'S'-dataset

C.2.1 For 'country' attribute.

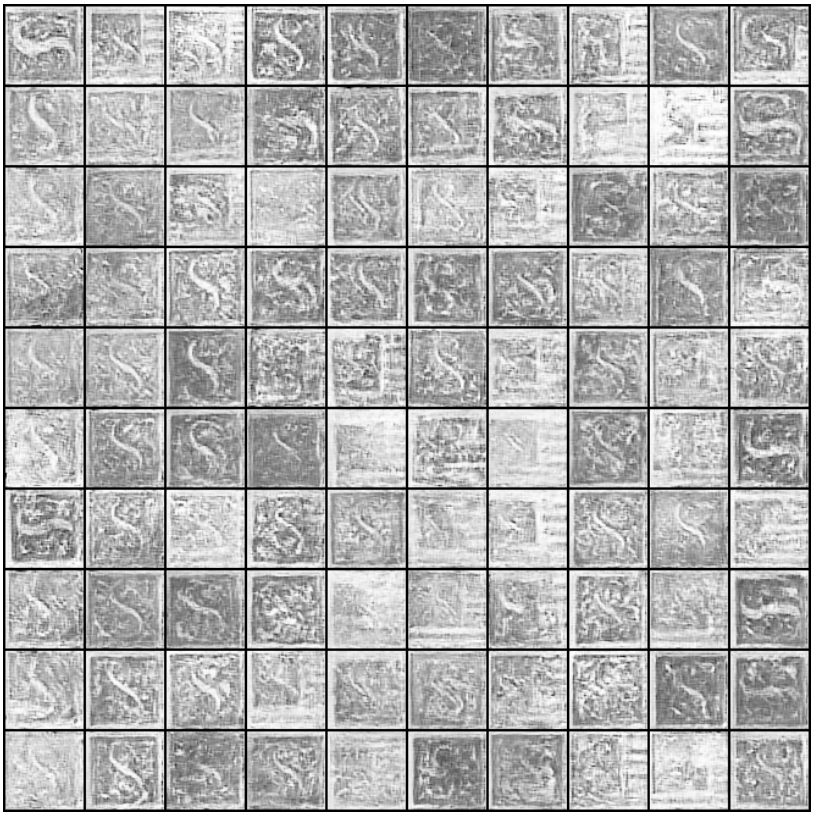


Figure 31: ACGAN results when trained on the 'S'-dataset for the 'country'-attribute. Each column corresponds to a value for the attribute, from left to right: FR, CH, DE, ES, IT, BE, GB, POR, NL, DK.

Figure 32: In each row, the rightmost image is conditionally generated by the ACGAN (trained on the 'S'-dataset) on the 'country' attribute. The consequent three images are the nearest neighbors from the initials dataset.

C.2.2 For 'city' attribute.

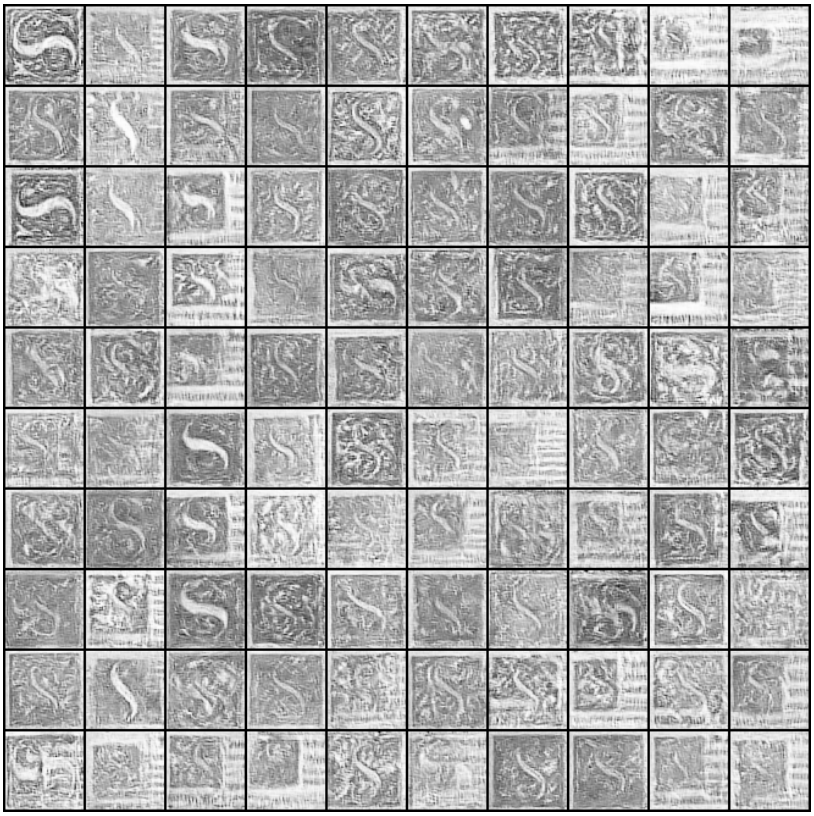


Figure 33: ACGAN results when trained on the 'S'-dataset for the 'city'-attribute. Each column corresponds to a value for the attribute, from left to right: Lyon, Paris, Basel, Zwickau, Valencia, Frankfurt, Brescia, Madrid, Antwerpen, Venezia.

Figure 34: In each row, the rightmost image is conditionally generated by the ACGAN (trained on the 'S'-dataset) on the 'city' attribute. The consequent three images are the nearest neighbors from the initials dataset.

C.2.3 For 'name' attribute.

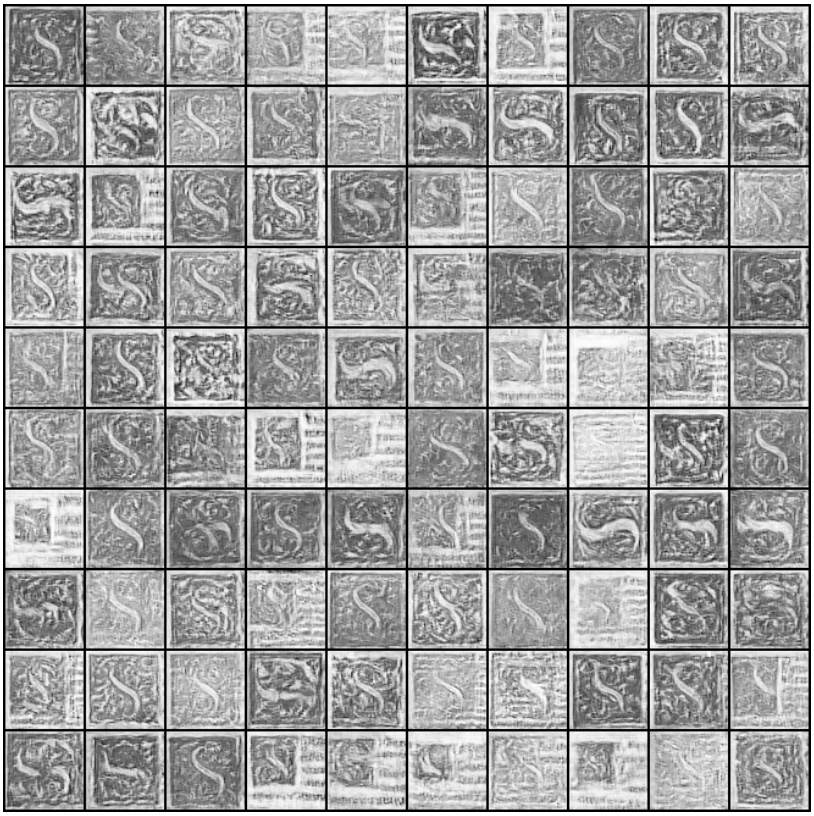


Figure 35: ACGAN results when trained on the 'S'-dataset for the 'name'-attribute. Each column corresponds to a value for the attribute, from left to right: Simon Bevilaqua, Matthieu David, Valentin Curio, Gabriel Kantz, Diego Gumiel, Jean Pillehotte, Andreas Cratander, Sebastien Cramoisy, Seitz Heirs Peter Main, Andre Bocard.

Figure 36: In each row, the rightmost image is conditionally generated by the ACGAN (trained on the 'S'-dataset) on the 'name' attribute. The consequent three images are the nearest neighbors from the initials dataset.