UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Farid Hasanov

Iterative versus amortized inference solutions to the constellation problem

Master's Thesis (30 ECTS)

Supervisor(s): Jaan Aru, PhD
Tarun Khajuria, MSc
Taavi Luik, MSc

Tartu 2022

Iterative versus amortized inference solutions to the constellation problem

Abstract:
Making sense of the visual inputs is an essential part of human intelligence. While processing in the human visual cortex has been observed to have recurrent nature, machine vision systems with one feedforward pass from input into prediction have dominated computer vision benchmarks. This discrepancy may be explained through lack of challenging datasets where gradual refinement of solution would be necessary to lead to correct solution. Such a dataset, where local information about the encoded objects has been erased, was recently proposed. The current thesis represents the first attempt to solve this novel dataset. We propose to use generative models DCGAN and VAE with optimization algorithm CMA-ME to refine the solutions as iterative inference, and use generative models Pix2pix and CycleGAN as feedforward or amortized inference. Through solving the problem posed in the novel computer vision dataset, we show the prevalence of iterative refinement of hypotheses over the single-prediction paradigm, encouraging further research in the field of iterative inference.

Keywords:
Deep Learning, Computer Vision, Convolutional Neural Networks, Generative Modeling, Image processing

CERCS: P176 - Artificial Intelligence, T111 - Imaging, image processing, B640 - Neurology, neuropsychology, neurophysiology

Iteratiivset ja amortiseeritud järeldamist kasutavad lahendused täht-kujude probleemile

Lühikokkuvõte:

Visuaalse sisendi mõistmine on inimmõistuse oluline osa. Kuigi inimese visuaalse korteksi töös on täheldatud tagasisisestatud töötlusahelaid, on masinnägemise andmestike lahendamisel domineerinud süsteemid, mis teevad ühe pärilevi arvutuse sisendist ennustusse. Seda lahknevust võib seletada keerukate andmekogumite puudumisega, kus õige lahenduseni jõudmiseks oleks vaja lahendust järk-järgult täiustada. Hiljuti pakuti välja selline andmestik piltidest, kust on kustutatud lokaalne informatsioon pildile peidetud objektide kohta. Käesolev lõputöö kujutab endast esimest katset selle uudse andmekogumi lahendamiseks. Teeme ettepaneku kasutada generatiivseid mudeleid DCGAN ja VAE koos optimeerimisalgoritmiga CMA-ME, et täiendada lahendusi iteratiivselt, ning kasutada generatiivseid mudeleid Pix2pix ja CycleGAN, et luua lahendusi pärilevi- või amortiseeritud järeldusena. Uudses masinnägemise andmestikus püstitatud probleemi lahendamise kaudu näitame iteratiivse hüpoteeside järeldamise üleolekut ühekordsest ennustamisest, julgustades edasisi uuringuid iteratiivse järelduse valdkonnas.

Võtmesõnad:
Sügavõpe, Tehisnägemine, Konvolutsioonilised Närvivõrgud, Generatiivne Modelleerimine, Pilditöötlus

CERCS: P176 - Tehisintellekt, T111 - Pilditehnika, B640 - Neuroloogia, neuropsühholoogia, neurofüsioloogia

Contents

1 Introduction

Human vision system is complex and exhibits characteristics of both feedforward [DZR12] and recurrent [LR00] behavior. While for the easier tasks, such as object recognition, human brain shows capability to solve them "via a cascade of reflexive, largely feedforward computations" [DZR12, p. 1], overall human brain has been shown to utilize 'feedback', or recurrent, connections between layers to process sensory input [KS20].

Computer vision has seen a development of both methods of computation, with deep feedforward methods dominating in a number of key visual inference benchmarks, such as CIFAR 100 [Kri09], ImageNet [DDS+09]. While feedforward paradigm makes sense in some tasks, often there may be multiple explanations for the provided visual input, in which case we may need a model that is able to generate a variety of good guesses to choose from. Imagine trying to find a shape (constellation) in the night sky. While good solutions obviously exist (such as famous constellations like Pleiades, Orion's belt), there could be a variety of different good shapes that can still fit with high resemblance to real objects. A person who is not aware of 'popular' shapes will most probably generate other, conceptually different solutions.

The most interesting part, though, is to hypothesize about exactly how a person would do it. Are they going to look at small clusters of stars and generate part of the solution first (like tail of the dog) and then try to expand on this tail to the full shape - dog? Will they imagine a full solution (like a dog) and then try to fit it somehow (in iterative manner) so that its contour matches stars? Or, perhaps, they will try to 'denoise', or erase, irrelevant stars out of the sky representation in the brain (stars, that are too far away from each other, or stars, that do not fit into any familiar shape) and leave only small subset of stars, out of which it is trivial to draw a shape?

1.1 Problem

Recently, Khajuria and colleagues [KTLA22] from the Computational Neuroscience lab at the University of Tartu presented a dataset called "Constellations", which was inspired by the star-constellations. This dataset is challenging for feedforward image recognition, as the images are hidden over random dots (Figure 1). The authors also conducted experiments with humans solving this task and demonstrated that humans iteratively try different hypotheses to solve these images [KTLA22]. In the present thesis we ask how would a machine vision system solve these images, as this could lead to new hypotheses in understanding of the human brain's processing of visual input. However, instead of directly tackling the "Constellations" dataset, which might be too hard for any current machine vision system due to the limited number of images per category, we have decided to work with datasets with more images per category – MNIST [LC10] and Google's Quick, Draw! [HE17] dataset. We have altered both original datasets in

the same way Khajuria and colleagues did by breaking the outline of shape into dots and uniformly adding noise on the image (Figure 1). This altered form is referred to as 'constellation' modality.

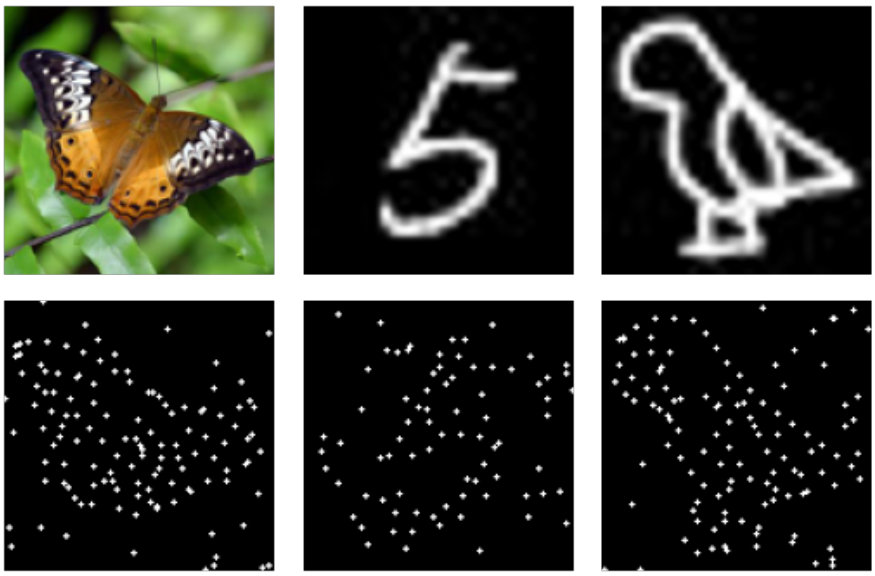

Figure 1. First row shows images in the original form from Constellations, MNIST, and Google Quick Draw! datasets respectively. Second row shows images in their altered form, where they are hidden among noisy dots.

1.2 Contribution

Through a series of experiments, we show how a framework allowing for generating guesses and improving them over time outperforms models that rely on single feed-forward pass for prediction. While the idea that theoretically iterative improvement is supposed to bring improvement to vision models has been proposed in literature [vK20], in practice it has been hard to demonstrate it on the existing datasets. With the novel dataset proposal from Khajuria and colleagues [KTLA22] we demonstrate superiority of iterative framework on the specific generative modeling task.

2 Background

This section will give an outlook on the field of generative modeling and provide background for both amortized inference (direct), as well as iterative inference models as they were used in our solution.

2.1 Iterative visual inference

Iterative inference can be defined as a hypothesis generation strategy that is refined over time by gradual improvements [vK20]. In order to implement this paradigm we have to tie together two processes: a hypothesis generation strategy and a way to make this strategy evolve over time.

We start the search with hypothesis generation strategy. In our case it means that we need to find a model that is able to generate images. The next section will give background on this family of models, called **generative** models.

2.1.1 Generative models

One of the main driving factors for innovation (and, arguably, for survival) in human society is competition. The training process where two (or more) models are trained in a competitive manner is called adversarial learning. **Generative Adversarial Networks** [GPAM$^+$14], or **GANs**, are based on the same idea - two neural networks, a generator and a discriminator, are trained simultaneously: the generator is trained to take random noise as input and output realistic images, and the discriminator is trained to discern real images from those produced by the generator (called 'fake' ones). Successive training helps both of them to improve at their goal. The generator is learning to capture the distribution of source images better and better even though it doesn't encounter it in training directly - during the training the generator only gets signal from the discriminator of how 'realistic' its samples are. The discriminator, at its turn, sees both fake and real images, and with time is learning to discern fake ones from reals better because the quality of fakes is increasing, too (Figure 2).

Mathematically, GAN training can be represented as a two-player zero-sum minimax game with the following value function V(D,G) (Eq. 1):

$$\min_G \max_D V(D,G) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z))), \tag{1}$$

where D(x) represents the probability of correct classification of real data as real, and D(G(z)) represents the probability of correct classification of data, produced by generator, as fake.

Architecture-wise, vanilla GAN can be represented by a fully connected neural network, which takes random noise vector as vector and learns to output images. The

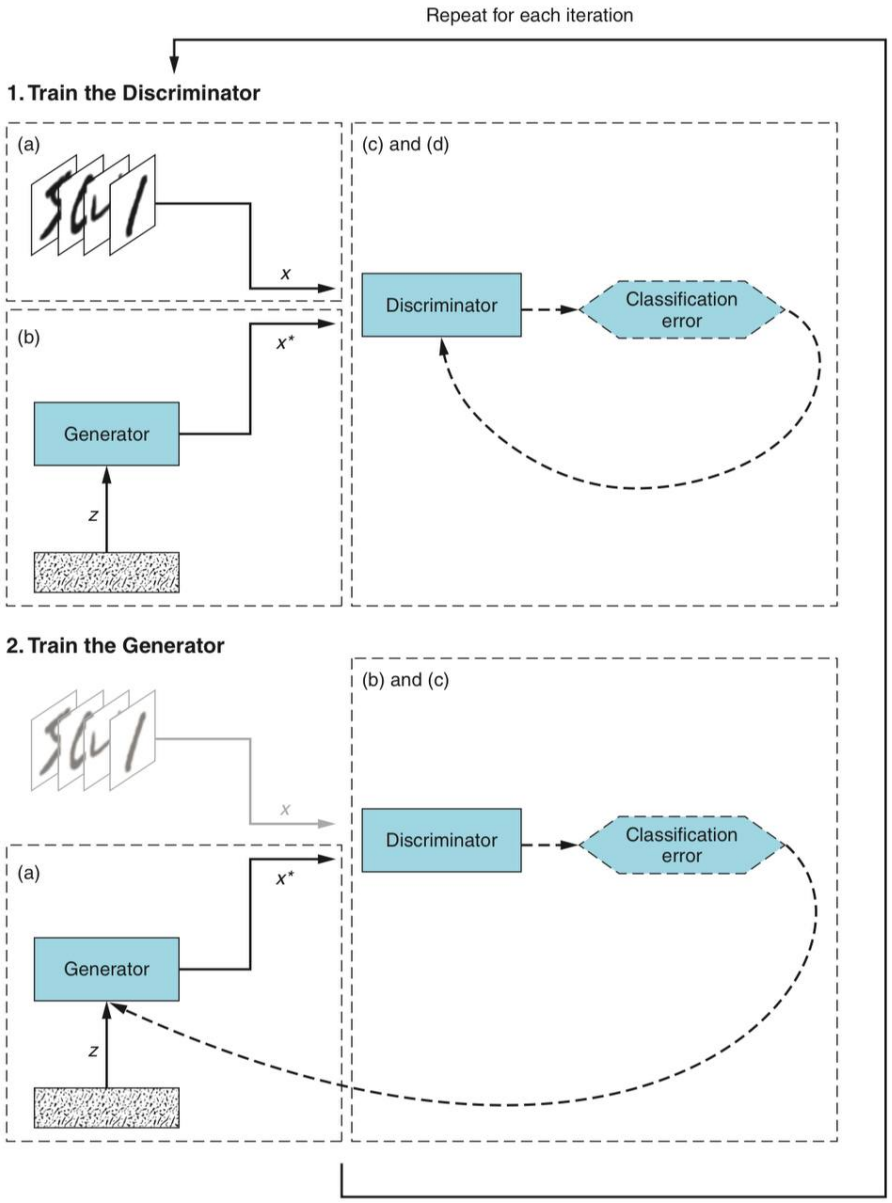

Figure 2. Typical training scheme for discriminator and generator. The discriminator is provided a set of real images and a set of images from the generator (so-called 'fake' images) and backpropagates classification loss based on how well it was able to predict the class of each set. The generator is provided a noise vector as input, and backpropagates classification loss of the images it generated (if they were all classified as 'real', then it does a good job) [LB19]

space this input vector spans is referred to as **latent** space of the generator. Discriminator, in its case, is also a fully connected neural network, that basically serves as a classifier - for the input image it outputs a label with indication of being 'real' or 'fake.

Allowing for random noise to serve as input gives greater variability among the generated images. Even though for simpler datasets the proposed architecture works, usually vanilla GAN suffers from unstable training (generator or discriminator may overfit one another), mode collapse (when generator only learns to produce only a limited variety of samples because of unfortunate local optima of weights), and vanishing gradients problem (due to which discriminator's output won't let generator update its weights). A number of improvements to basic GAN architecture and training have been proposed: **DCGAN** [RMC15], which replaces fully connected layers with strided convolution layers without pooling, **WGAN** [ACB] which proposed a superior measure for distance between real and fake data (Wasserstein distance), as well as using gradient penalty (or clipping), avoiding using Sigmoid function at the last layer of the discriminator, and others.

Another generative neural network, **Variational autoencoder** [KW13], or **VAE**, consists of two neural networks: encoder and decoder. The objective in training is to learn a continuous, lower-dimensional, latent representation of the images in training in order to be able to reconstruct the original image out of it. The general principle is similar to principal component analysis (PCA), where latent representation is generated by low-dimensional factorization of the covariance matrix of the data [BB21]. In VAE the input image is first encoded into a latent representation by encoder part of the network, and is then restored into normal size image by decoder part of the network. The usual architecture of autoencoder consists of downsampling convolutional layers in encoder, and upsampling transposed convolution layers in decoder part. Variational Autoencoder's objective can be represented by formula (Eq.2):

$$\log\left(p(x)\right) = KL(q,p) - \mathbb{E}_{q(Z|X)}[\log\left(q(z|x)\right) - \log\left(p(x|z)p(z)\right)], \tag{2}$$

where KL is Kullback Leibler divergence (mathematical expression measuring distance between distributions), and the second term is known as Variational Lower Bound (lower bound for log probability of real distribution). In the process of learning lower-dimensional representation on images the latent space spanning the training data is generated. Later, this learned latent space can be sampled for random latent vectors, from which decoder is going to generate new images [BB21].

By picking a generative model we have only fulfilled one piece of the puzzle - a hypothesis generation strategy. The second piece we need to make it work is a way to improve the hypotheses that are generated. For this we will turn to **optimization** algorithms.

2.1.2 Optimization algorithm

The problem statement was that generated shapes should be fitting at least partially the dots present on the image (whether the found shape is the same with the one that was 'encoded', is unimportant - creative solutions are good as well). In order to achieve this we need to walk in the latent space of our generative model and refine solutions, guided by a loss function, consisting of object realness and passing-through-dots loss. This search for the best solution, under constraints and with a well-defined objective function is called optimization. As there is an interest in many different solutions which all satisfy the objective to a certain extent, attention was paid to **Quality-Diversity optimization** algorithms.

Quality-Diversity optimization [CCVM20] is a branch of optimization algorithms that strives to produce optimum solutions that differ across given attributes [QDW]. This may be considered a paradigm shift from single-objective optimization, where the goal is to find the most optimal solution for the objective function that was defined. In reality it is usually more beneficial to see a range of different and novel solutions, which are marginally optimal, than having just one solution, which may be considered global optimal.

The algorithm we will be using for optimization was introduced in [FTNH19] and is called Covariance Matrix Adaptation MAP Elites (CMA-ME) . This is a quality-diversity algorithm that uses CMA-ES [Han16], a stochastic evolutionary optimization algorithm, to search for solutions, and uses MAP Elites [MC15] inspired archive mechanism to store and iterate through best solutions.

2.2 Amortized visual inference

Amortized inference solutions are defined as solutions, achieved by a single mapping from $X \rightarrow Y$ [vK20]. In this sense it may be viewed also as a hypothesis generation strategy, with the focus on generating 'right' solutions by doing a single pass through a feedforward network rather than improving guesses along the way.

It's also important to draw a distinction line between amortized and iterative inference solutions. While in both of them we actually have a feedforward convolutional network, having an optimization algorithm to improve on the next guess of the latent space vector is what makes the procedure iterative, not the architecture itself.

To perform amortized inference we need to find a model that can directly draw a shape on the image without the need (or capability) to improve it. Luckily there is a family of models that lets us do this, which are referred to as **conditional** GANs [MO14].

Naturally, the problem of finding a shape in an image with dots (see Figure 1) could be viewed as an image-to-image translation problem. One of the most frequently used models in this field is called **Pix2pix** [IZZE16].

Pix2pix is a GAN variation that takes images as an input and learns to transform

them into a target domain. As the generated image is 'conditioned' on the input image, it falls into the category of conditional GANs. This method has a number of successful implementations on various tasks, such as satellite image -> map correspondence [IZZE16], facade labels -> photo [TŠ13], semantic labels -> photo translation [COR$^+$16] and others. An example of its prediction versus ground truth can be seen on Figure 3. Notice that Pix2pix requires sets of paired images, meaning that for each image in the training set there should be a direct correspondent of it in the target domain.

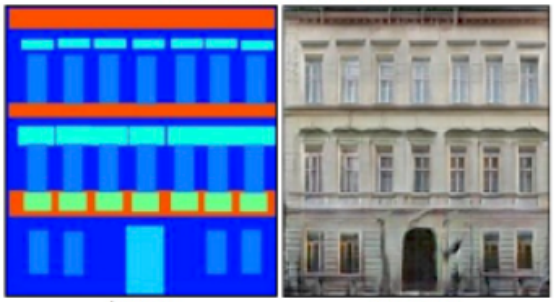

Figure 3. Facade label image is translated into realistic-looking image of facade in the real world [IZZE16]

Pix2pix uses a similar loss schedule foundation with feedback from discriminator as the other GAN network we studied previously. On top of that authors found it beneficial to add an additional term for increasing image sharpness, which is weighted L1-loss between generated and target sample.

Generator in Pix2pix is usually of U-Net architecture, which is quite similar to encoder-decoder setup, with addition of skip connections between layers. These skip connections allow sending important features not only between consequent layers, but also between each i^{th} layer in encoder and $(n-i)^{th}$ layer in decoder (where n is the total number of layers in the generator) [BB21]. See Figure 4 for visual representation.

Ordinary discriminator learns to output label per image with indication that it is real or fake image. Pix2pix paper [IZZE16] introduced the discriminator that learns to output NxN grid per image with indication of being real or fake for each of the small grid cells. This kind of discriminator is called PatchGAN (Figure 5). The reason for this shift is to take into account cases where the model learned to produce only parts of the image correctly; it allows a more accurate loss function for improvement of the generator.

CycleGAN [ZPIE17] architecture may be considered a step forward as it eases up the requirement of having paired images. The training dataset consists of two datasets, source and target, without exact one-to-one mapping in between them. In order to execute the strategy of unpaired image transfer, the authors propose a concept of cycle consistency. Imagine two generators, G and F, such that G learns to output mapping X -> Y and F

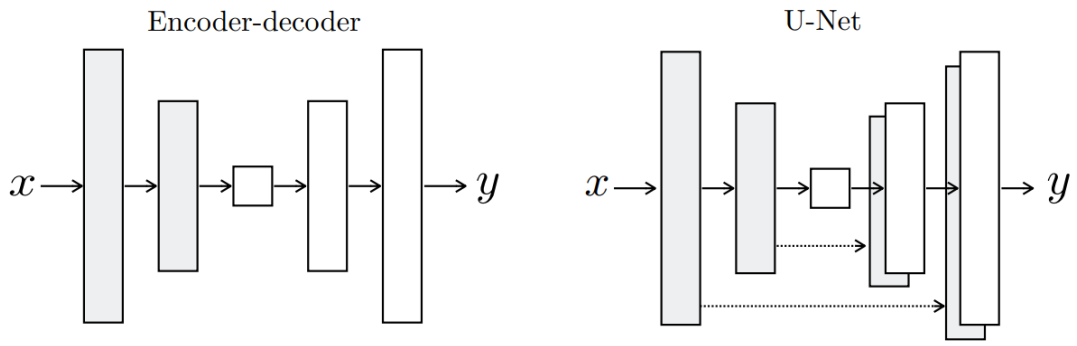

Figure 4. Typical U-Net architecture in comparison with Encoder-decoder scheme of variational autoencoder [IZZE16]

learns to output mapping Y -> X. During training generator G gets an image x which it transforms into $\hat{y}$, and $\hat{y}$ is transformed into $\hat{x}$ back using F generator. The loss between x and $\hat{x}$ is going to be propagated to update G and its respective discriminator, D_y, and is called cycle consistency loss. (Figure 6)

The same loss will be calculated for another generator, just in reverse order, and F with D_x is going to be updated. Overall loss for the training of CycleGAN is going to be a weighted sum of usual adversarial loss, just as for any GAN, with cycle consistency losses introduced here and identity loss (a form of regularization).

The usual architecture of CycleGAN generators consists of convolutional downsampling layers which make up encoder part of the network, convolutional layers that transform the image making up transformer part of network, and upsampling transposed convolutional layers that make up decoder part of the network . Architecture of discriminators is Patch-GAN, same with Pix2pix.

Figure 5. Comparison of ordinary discriminator (on the left) with PatchGAN discriminator (on the right) [BB21]. PatchGAN outputs a grid per image, where each part of the image has indication of being fake or real.

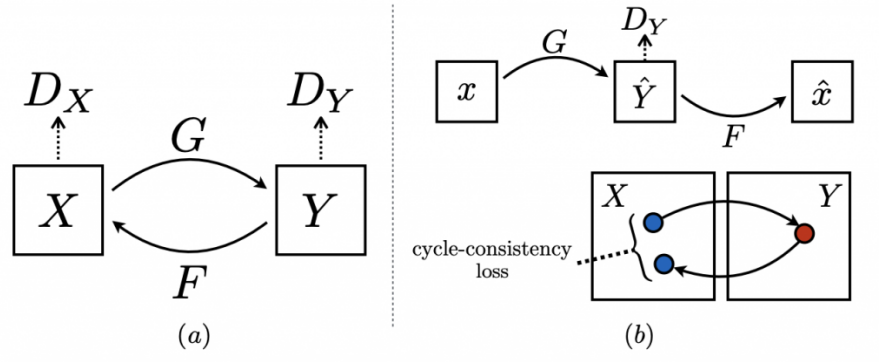

Figure 6. Training dynamics for CycleGAN [ZPIE17]

2.3 Denoising argument

In one way, we can also imagine applying VAE directly to solve our problem, as literature has shown autoencoder success in solving denoising problems [PKJ20]. However, this approach is not directly applicable in our case. Looking at Figure 7, we can see that the noise here is applied all over the image but the original contour of the shape that was present is not disrupted, while in constellation images case (Figure 1) the contour of the shape is broken down into dots, which makes the problem more complex. Experiments have proven that directly applying VAE architecture to solve constellation images leads to poor results.

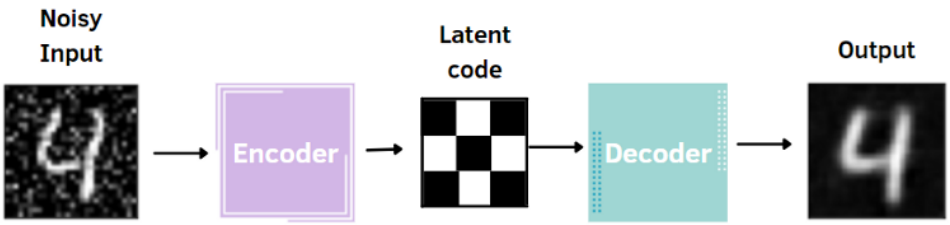

Figure 7. Autoencoder solving denoising problem [Ane22]

3 Data and Methods

This section will introduce datasets we have used, demonstrate exact neural network architectures selected, as well as overall training setup for each of them.

3.1 Datasets

The original version of this problem included a dataset, composed by Khajuria and colleagues [KTLA22]. Their dataset basically had two variations: one was an altered version of Things dataset [HDK+19] with about 3533 images of 1215 objects, and another was an altered version of TU-Berlin dataset [EHA12] with 25000 images over 250 categories. Figure 8 shows some examples of images from [KTLA22].

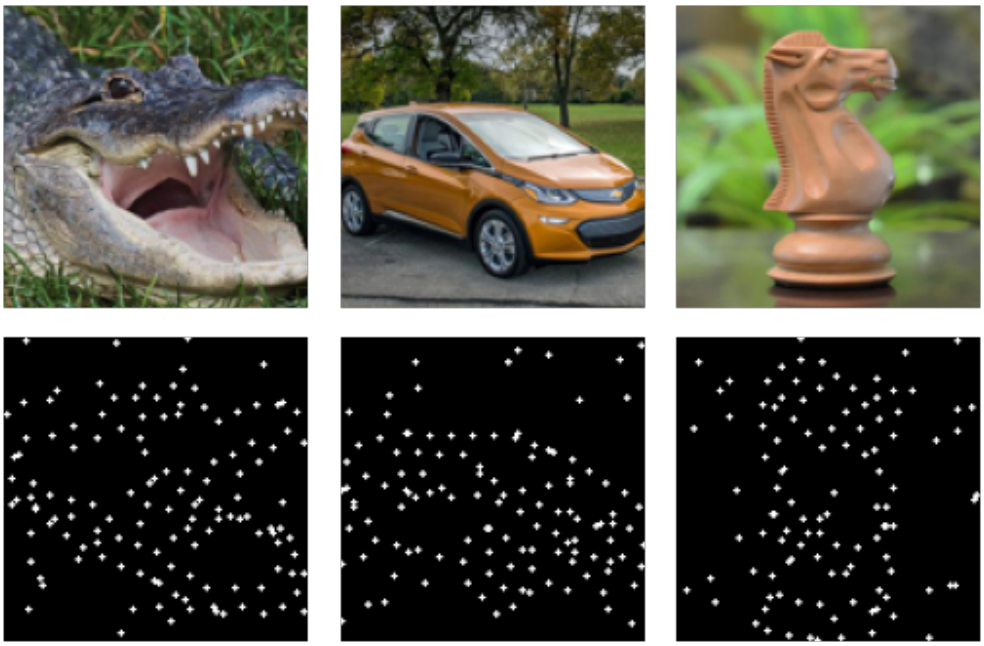

Figure 8. Images from 'Constellations' dataset [KTLA22]

As we wanted to train our models from scratch, we needed datasets that would have more images per category. Using the code published along the 'Constellations' dataset, we can alternate any image dataset into constellation format. Thus, we have selected two famous and image-rich datasets: MNIST [LC10] and Google Quick, Draw! datasets [HE17].

MNIST is a dataset of digital 28x28 images of hand-drawn digits from 0 to 9, assembled together in 1998 by Yan LeCun, Root Cortez and Christopher Burgess [LC10].

The dataset was built from a number of scanned document sets available at the National Institute of Standards and Technology (NIST). The images of the figures were taken from a variety of scanned documents, normalized in size and located in center. The database consists of 60000 images in the training set and 10 000 in the testing set. Figure 9 shows some examples of images from the original MNIST dataset.

Figure 9. Images from MNIST dataset [LC10]

The Quick, Draw! dataset [HE17] is one of biggest public datasets for sketch classification tasks. It was assembled from drawings of real players in Quick, Draw! online game and contains 50 million sketch images, divided into 345 categories (such as alarm clock, dinosaur, bird etc). Figure 10 shows some examples of sketches in the Quick, Draw! dataset.

Figure 10. Images from Quick, Draw! dataset [HE17]

3.2 Constellation images

For each of the datasets, a constellation version was created with different signal/noise ratios. This ratio can be affected either by perturbing distance between dots of the original outline or changing the amount of noisy dots added on the image. Lower distance between dots corresponds to easier-to-recognize shape, and thus corresponds to providing high signal. We sometimes substitute distance between dots of original outline as **d** in text for simplicity. Figures 11 and 12 show how constellation images look.

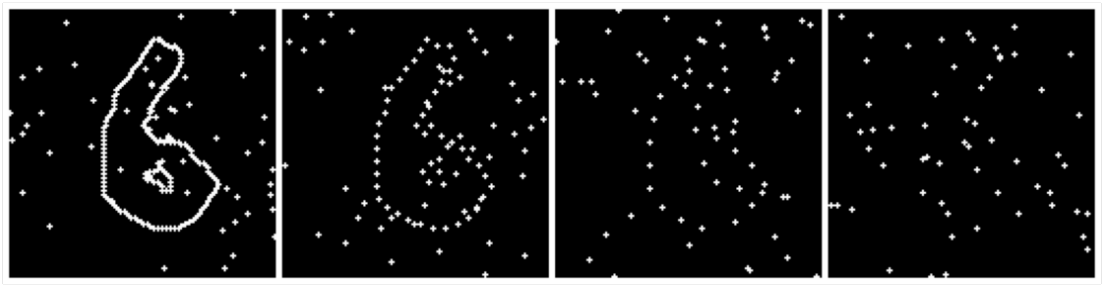

Figure 11. Constellation images with increasing distance between dots from left to right (1, 5, 10, 15) and noise level = 0.003. It can be observed how it becomes challenging for a human to 'decode' the hidden shape starting from **d** 10, and at 15 it is practically impossible.

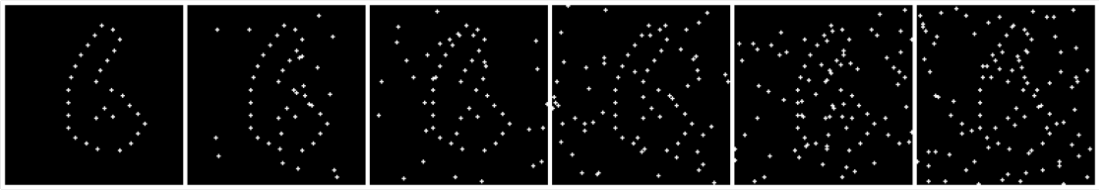

Figure 12. Constellation images with distance between dots 8 and noise level [0-0.006] with step 0.001. At higher levels of noise more random noisy dots are added all over the image.

We ran experiments with small noise ranges (up to 0.003) and varying **d** from 1 to 15. The reasons we decided not to improve in the direction of highly noise environments are these:

1. Any big enough shape, generated as a solution, crosses some amount of dots. In our experience, random shapes that are not conditioned to cross dots cover about 25% dots nevertheless (we allow a small margin of dot being close to the generated shape to also account for crossing), while good solutions, even the correct one (the one out of which the noised image was generated), covers about 45% of dots (at noise level 0.003). When we increase noise level even more, random realistic shapes become almost indistinguishable from shapes that were taught to cross dots in their generation. To account for this fact, it makes sense to experiment with a moderate amount of noise.

2. Increasing distance between dots measure (basically, deleting more and more dots from the contour outline of encoded shape) produces a more complex task rather than increasing noise around the image. It is easy to recognize the shape even in a highly noisy environment if **d** is low, but it is almost impossible to find out what it was if **d** is high in low noise environment (Figure 13)

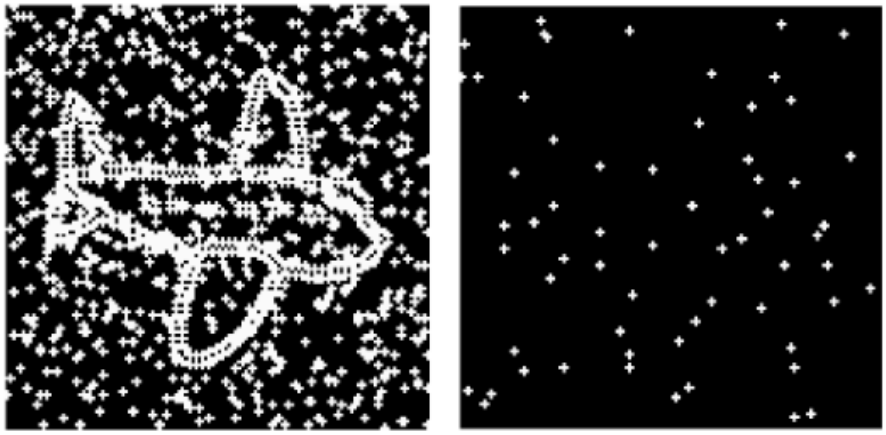

Figure 13. The airplane can be still clearly recognized on the left picture (despite high noise) while on the right picture it is very hard to imagine what was encoded originally (the right image is generated with high **d** measure).

At the start of the project we have decided to first run experiments only on an altered version of MNIST dataset, as a 'more simple' task, opposed to Google Quick, Draw! dataset, as this can give an idea about how good the proposed model is. If the model can't work well with MNIST dataset, it doesn't quite make sense to check it on more

complex, both in terms of amount of categories, as well as complexity of the shapes, dataset of Google Quick, Draw!

3.3 Amortized inference solutions

Pix2pix and CycleGAN were used as amortized inference solutions. They are trained on direct mappings from one image domain (constellation images) into another (shapes). In Pix2pix this direct mapping is ordered, meaning for each constellation image the 'correct solution', the image it was generated from, serves as its target, while in CycleGAN the mapping is unordered, meaning we have to supply two unordered sets of images - one is in constellation modality, another one is original.

3.3.1 Training setup of Pix2pix

Pix2pix was trained for 100 epochs with initial learning rate = 0.0002 and batch size = 4 using Adam optimizer with reducing learning rate on plateau (patience = 5). These hyperparameter choices were used following recommendations from the original paper [IZZE16] and tested in practice to ensure best results. Figure 14 shows our final architecture of the Pix2pix generator.

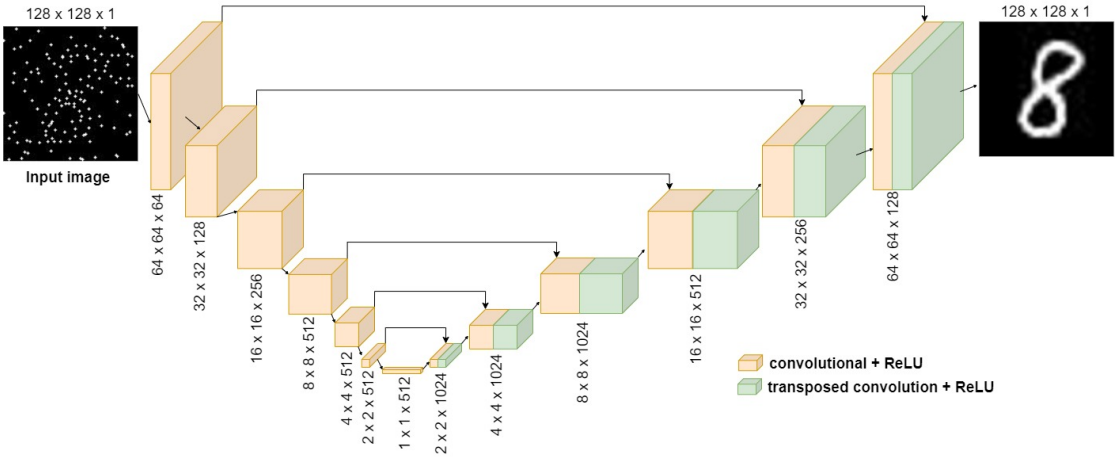

Figure 14. Pix2pix generator architecture consists of 6 convolutional layers, which shrink up the input image dimensions to 1 x 1 x 512 (contracting path), after which the image is being upsampled by 6 consecutive transposed convolution layers (expansive path). Transposed convolution layers receive input both from the previous layer, as well as from convolutional layers out of expansive path.

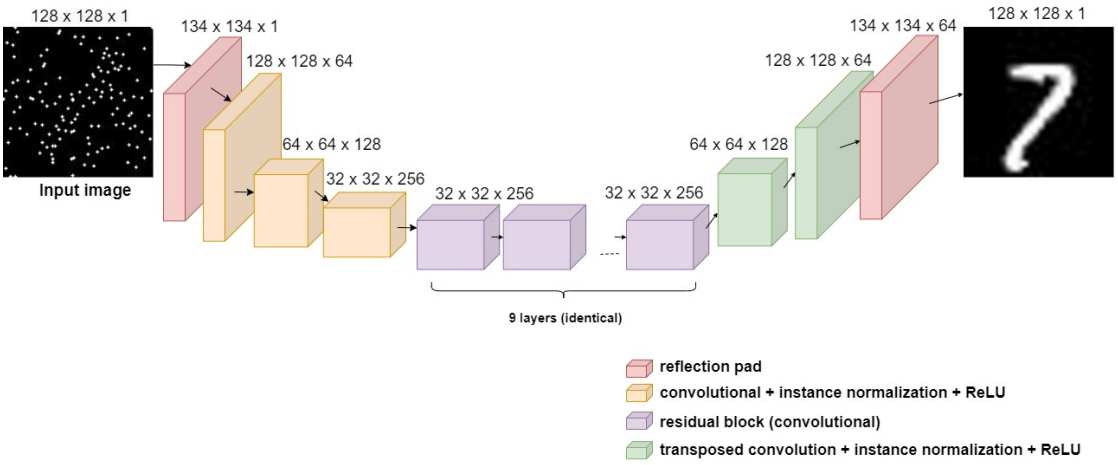

Figure 15. CycleGAN generator architecture consists of a Reflection pad layer, three convolutional layers, followed by 9 Residual block layers, 2 transposed convolution layers and Reflection Pad layer.

3.3.2 Training setup of CycleGAN

CycleGAN was trained for 50 epochs with initial learning rate = 0.0002 and batch size = 1 with Adam optimizer, and 50 more epochs with linear scheduling of learning rate to 0. In practice it seemed like the network didn't improve the quality of predictions after the first 50 epochs. Figure 15 shows the final architecture of the CycleGAN generator that we have used.

Keeping batch size low for both Pix2pix and CycleGAN training has been referenced in their respective papers [IZZE16], [ZPIE17], although being disputed later [LSO22]. We have observed that low batch sizes indeed lead to better sharpness in final images, at the cost of longer training times.

3.4 Iterative inference solutions

Convolutional VAE and DCGAN, combined with CMA-ME optimization, were both used as iterative inference solutions. It's important to note that both of these generative neural networks were trained separately on the original dataset without constellation images. By the time training is over both models can take random noise vectors as inputs, and generate images out of them.

In both cases this input vector space serves as a solution space for CMA-ME to optimize on (in VAE the noise vector is given as input to the decoder part of the network, and in DCGAN the noise vector is given as input to the generator). Generated images are evaluated based on realness of the generated shape (based on the feedback of the

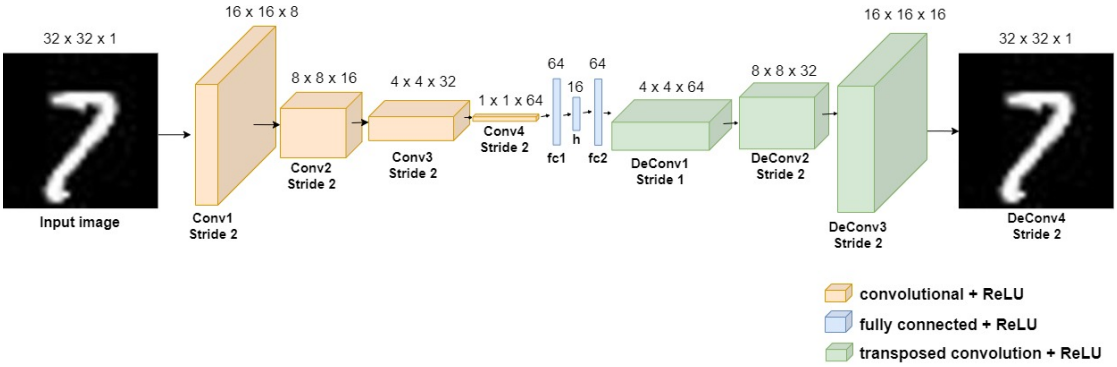

Figure 16. Convolutional VAE architecture consists of four convolutional layers in the encoder part, three fully connected layers in the middle, and four transposed convolution convolutional layers in the decoder part. h vector serves as a latent space that we will use for sampling. Notice that images are of 32 x 32 size.

discriminator) and the amount of dots crossed by the generated shape (weighted sum of two served as the objective function for optimization). CMA-ME is varying the input vector with the goal of maximization of the objective function. The advantage of the CMA-ME optimization algorithm over single-objective algorithms is that it can treat certain characteristics of objects as attributes and look for different solutions across these attributes - in our case variable attributes were selected to be class, rotation angle and number of fitted dots of generated shape.

3.4.1 Training setup of Convolutional Variational Autoencoder

Convolutional VAE has been trained for 200 epochs with initial learning rate = 0.001 and batch size = 128 using Adam optimizer with reducing learning rate on plateau (patience = 5). Figure 16 shows the exact architecture of the network we have used.

3.4.2 Training setup of DCGAN

DCGAN has been trained for 100 epochs with initial learning rate = 0.001 and batch size = 128 using Adam optimizer with reducing learning rate on plateau (patience = 5). Figure 17 shows the exact architecture of the network we have used.

To perform iterative inference, we were running an iterative process of improving initial guess with CMA-ME for 300 iterations per image.

Most of the computations were carried out using HPC Center of the University of Tartu resources with NVIDIA Tesla A100 16GB GPU.

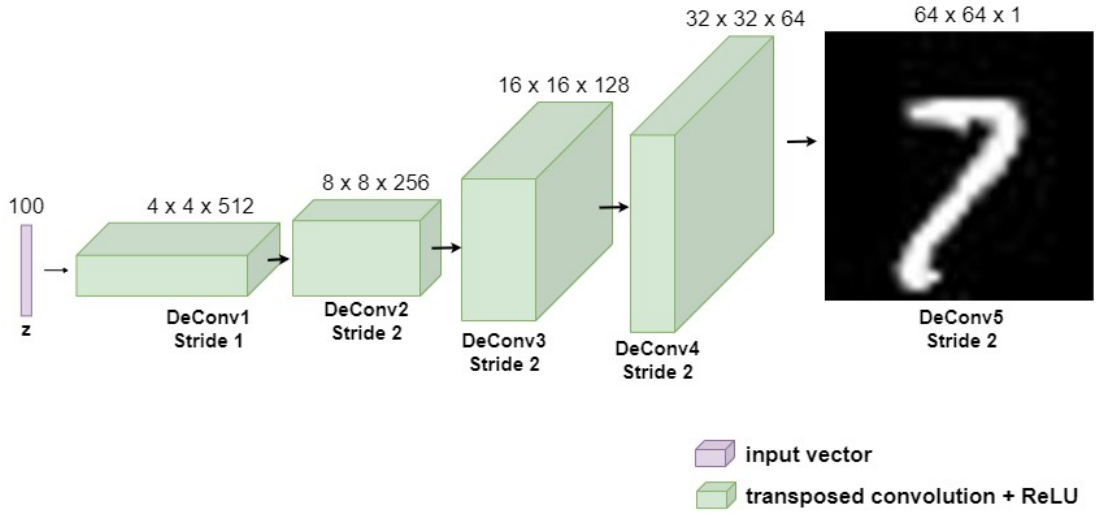

Figure 17. DCGAN generator architecture consists of five consecutive transposed convolution layers, where a vector of size 100 serves as an input to the first one layer. The z vector serves as a latent space that we will use for sampling. Notice that images are of 64 x 64 size.

3.5 Evaluation

The generated images were evaluated on two terms - based on generated object realness (determined by a well-trained discriminator) and the amount of dots covered (let's call it d.c loss). It is important to see that the model is trying to fit the shape into the dots, rather than simply locating the generated shape on any corner of the image without consideration of passing through dots. This produced some difficulties, as having the same weights for object realness and d.c loss doesn't produce good results. The reason for it is because we tend to focus too much on the dots coverage of the image and inevitably really bad non-realistic solutions with high dot coverage get good enough scores. On the other hand, only relying on the realness of the generated shape is not an option, as we specifically pose the problem as drawing a shape that passes through dots. By conducting studies with different weights per each term, we have found out that having 0.9 weight for object realness and 0.1 weight for d.c loss in the total loss equation serves as the most fair metric in our case.

4 Results

In this section we will take a look at the generated shapes out of constellations, and show comparison between models using evaluation metrics.

4.1 Amortized inference results

Amortized inference in this thesis was studied through applying two image-to-image translation neural networks: CycleGAN and Pix2pix. They both worked quite well on what we consider 'easier' conditions of images - where **d** is low. With increasing level of difficulty their performance started to worsen. We are going to look at each one in particular in the upcoming sections.

4.1.1 Performance of CycleGAN

It seemed natural to start the amortized image translation solution from CycleGAN, as it allows for learning from unpaired images. Theoretically it seems that if the model is learning to translate images in unpaired setting, it won't just decode the shape that we have hidden, but be more creative in its generation process (as it didn't see how hidden shape is 'decoded' into the correct shape in training), which will make it more suitable for our task.

As a sanity check, it was interesting to see if the model is able to fit a shape for an image without any noise dots. CycleGAN was able to do it perfectly, as can be seen on Figure 18.

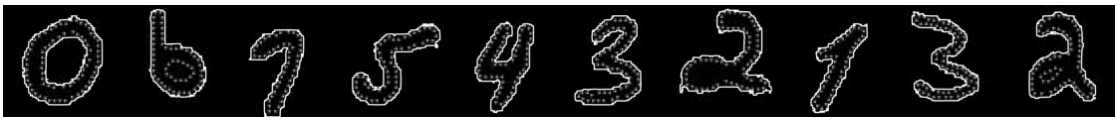

Figure 18. Prediction of CycleGAN on MNIST dataset with (**d**, noise level) = (8, 0) condition. All of the generated shapes look decent and cover dots. The overlay of the generated outline on the original constellation type image is done later by other piece of code, as the model is producing a shape without dots directly.

We see that with increasing amounts of noise CycleGAN predictions started to deteriorate. Because 0.003 level of noise (check Figure y for reference of how image looks like) is still a level at which it is pretty easy for a human to see the encoded shape and connect it without even creative thinking, while CycleGAN seems to already not be able to solve the given task, we have decided to continue our pursuit in a more robust direct inference model.

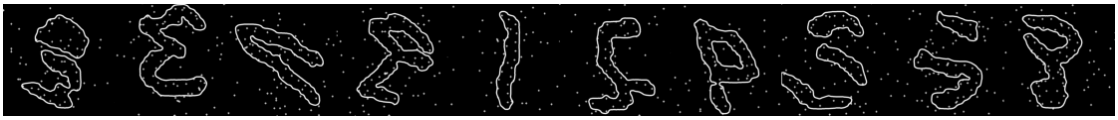

Figure 19. CycleGAN batch prediction on MNIST dataset with (8, 0.003) condition. Apart from one image, we can observe that CycleGAN is failing on all other samples. Even though some of the shapes can be creatively classified by humans into something, (e.g. the third from the left can be imagined to be a twisted snake!) CycleGAN was trained on digits, so it has to be able to produce digits only - everything else is considered to be a hallucination of some kind.

4.1.2 Performance of Pix2pix

Using Pix2pix has been observed to be a more robust method to solve constellation images than CycleGAN. After validating that it works on images without noise, we started experiments to see up to what levels of **d** it can work seamlessly. You can observe how it produces good shapes on (8, 0.003) condition on Figure 20, but still fails at (13, 0.003) level (Figure 21). The breaking point seems to be at around $\mathbf{d} \approx 12$.

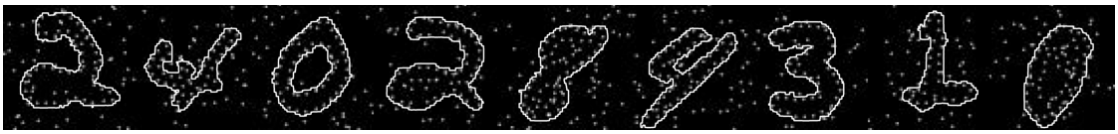

Figure 20. Pix2pix batch prediction on MNIST dataset with (8, 0.003) condition seems to work quite well in most cases.

Figure 21. Pix2pix batch prediction on MNIST dataset with (13, 0.003) condition seems to fail in most cases.

Seeing that Pix2pix worked better overall than CycleGAN, we have decided to check it on another dataset, which is Quick, Draw! With that dataset the results are more or less same, with it working decently $\mathbf{d} = 8$ but failing on $\mathbf{d} > 12$. Figure 22 shows how Pix2pix works on (8, 0.003) condition on Quick, Draw! dataset.

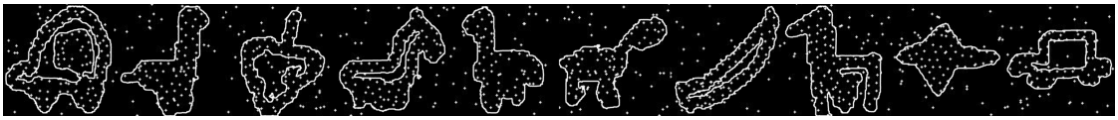

Figure 22. Pix2pix batch prediction on Quick, Draw! dataset with (8, 0.003) condition

4.2 Iterative inference solutions

Iterative inference framework was implemented through walking in latent space of two generative models: VAE and DCGAN. They were both robust against high **d**, but each with its own caveats that we will discuss in the upcoming sections. Due to the fact that the iterative inference solutions are evolving through iterations, we will also show the images that were generated during this path to display how it works.

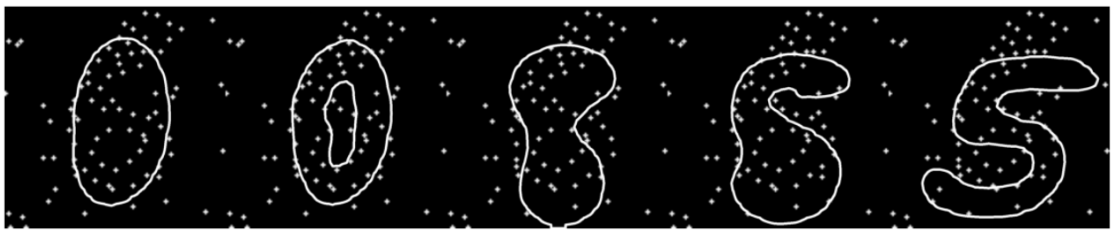

Figure 23. From left to right - evolution of the best solution generated per batch of VAE. First represents the initial guess, the last represents the final solution. (0 to 300 iterations)

4.2.1 Performance of VAE

Figure 23 shows the best solution evolution of VAE-based iterative framework over 300 iterations. Over time images are refined to become more realistic and to fit more dots. While the first image fits only 24 dots, the last image, presented as the final solution, fits 38 dots already, which confirms that algorithm works as expected.

We studied performance of VAE under different levels of difficulty (Figures 24,25,26). We observe how solutions that are generated by VAE are robust against high **d** values in terms of object realness. In some cases we observe VAE failing to fit obvious clusters of dots (such as the third image in Figure 24), which we refer to overall process stochasticity, and general inflexibility among images of VAE. We will talk about this more deeply in the Discussion section.

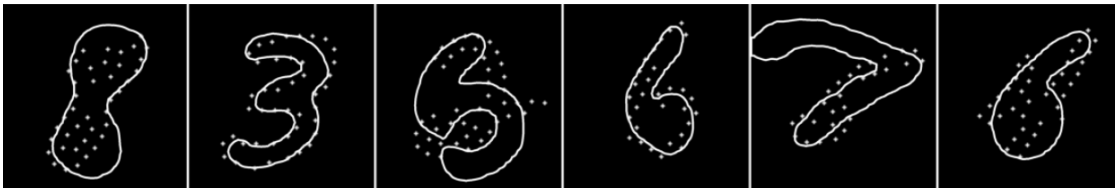

Figure 24. VAE best prediction on (8, 0) condition on MNIST dataset

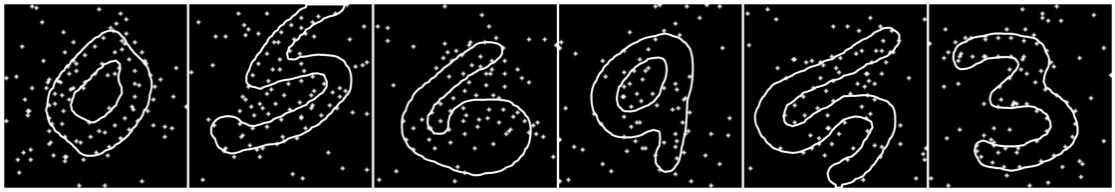

Figure 25. VAE best prediction on (8, 0.003) condition on MNIST dataset

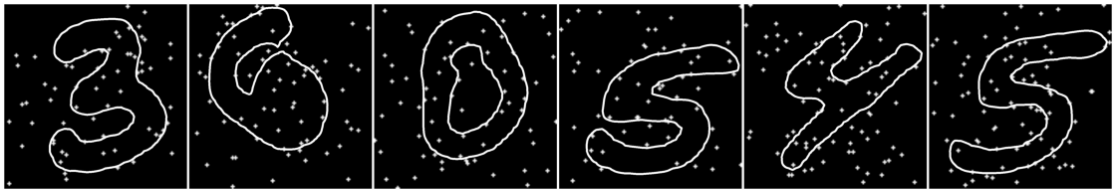

Figure 26. VAE best prediction on (13, 0.003) condition on MNIST dataset

4.2.2 Performance of DCGAN

We evaluated DCGAN-based iterative solution under different conditions to check its robustness. Figures 27 and 28 showcase how the best solution evolved throughout the iterative process of CMA-ME algorithm on DCGAN images. It is crucial to remember that the goal of improvement for these images is not only to be real-looking, but also fit the dots. For example, in Figure 28 the first image is generated using a random input vector and it passes through only 17 dots. As the iterations continue, shapes start to fit the dots more and more, with the last shape passing through 36 dots.

We observe how DCGAN generated images seem to be able to fulfill the goal both in easy condition of d = 8, as well as hard condition of d = 15 (see batch predictions on different datasets on Figures 29,30,31).We also noticed how as the iterations progress shapes start to try to 'cheat', by making their contours wavy with the hope to capture more dots along the way. We will theorize about why this happens in the 'Discussion' section.

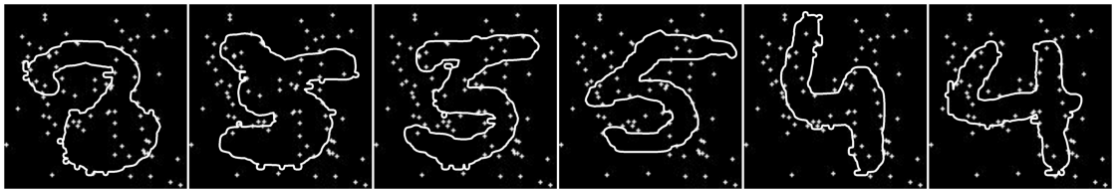

Figure 27. From left to right - evolution of the best solution generated per batch of DCGAN model on (8, 0.003) condition (0 to 300 iterations).

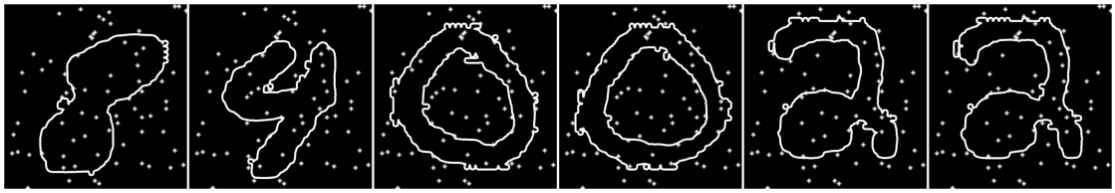

Figure 28. From left to right - evolution of the best solution generated per batch of DCGAN model on (15, 0.003) condition (0 to 300 iterations).

Figure 29. DCGAN best prediction on (8, 0.003) condition on MNIST dataset.

Figure 30. DCGAN best prediction on (15, 0.003) condition on MNIST dataset.

Figure 31. DCGAN best prediction on (15, 0.003) condition on Quick, Draw! dataset (airplane, key, giraffe, car, ant)

4.3 Comparison between models

Here we will take a look at predictions of the different models on the same image, and then dive into more quantitative measures to allow for better comparison across models.

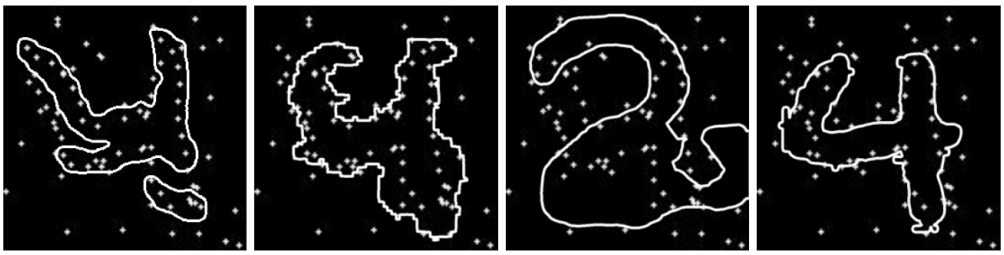

Figure 32. From left to right - CycleGAN (30), Pix2pix (43), VAE (40), DCGAN (38) prediction on (8,003) condition of the constellation images. The number in the brackets reflects the amount of dots crossed. The total number of dots of original shape that was encoded is 37 (without noise).

4.3.1 Prediction on the same image

Figure 32 shows results of the prediction for the same image to allow for better comparison. We observe CycleGAN guessing the upper part of the image correctly (in this case encoded number was 4), Pix2pix guessing the image correctly with a bit wavy contour, VAE drawing completely different shape which is still satisfying the main objective (realistic and crossing many dots), and DCGAN drawing also a bit wavy outline of the correct solution. We have encountered different solutions in training, but we believe that overall picture is best reflected through the results of exactly this particular comparison - CycleGAN drawing a somewhat unrealistic images, VAE failing to get accurate solution if the same image was not present in its training and hence not reflected in its latent

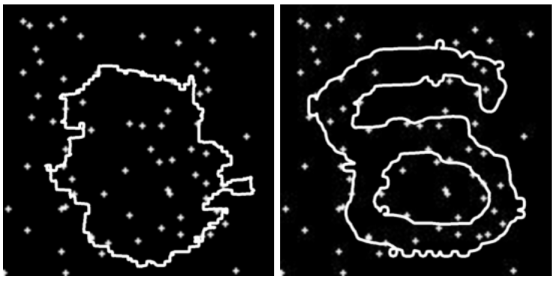

Figure 33. Pix2pix (30) and DCGAN (41) prediction on (13, 0.003) condition. The number in the brackets reflects the amount of dots crossed. The total number of dots of original shape that was encoded is 23 (without noise).

space, Pix2pix working quite well but generated image has noisy contour, and DCGAN delivering the best results among these four models (with slightly wavy contour, too).

We can observe and compare how two best performing models from the last round work on the more difficult case with **d** = 13 (Figure 33). Indeed, while DCGAN, with a wavy contour still manages to produce realistic shape, passing through the dots, Pix2pix produces vague shape (perhaps, close to 0) which doesn't fit the dots too much.

4.3.2 Quantitative comparison

We have evaluated our models based on the objective function described in section 3.5. We benchmark objective function score on 4 conditions: (1, 0.003), (5, 0.003), (8, 0.003), (13, 0.003) with increasing level of difficulty. Figure 34 summarizes our findings about the models. We see how CycleGAN fails to deliver good performance starting from **d** = 8 (and already sees a small drop of performance on **d** = 5). Pix2pix works well up until **d** = 13. VAE performs a bit worse on easier condition of **d** = 1, which is perhaps due to its inability to find exact matches for the encoded shape. Finding an exact match is especially important in easier conditions, as many of the dots present in the picture belong to the original outline. For harder conditions it becomes more important to be able to 'daydream' solution rather than finding exact encoded shape, which VAE can successfully do. Lastly, DCGAN performs almost equally well under any condition.

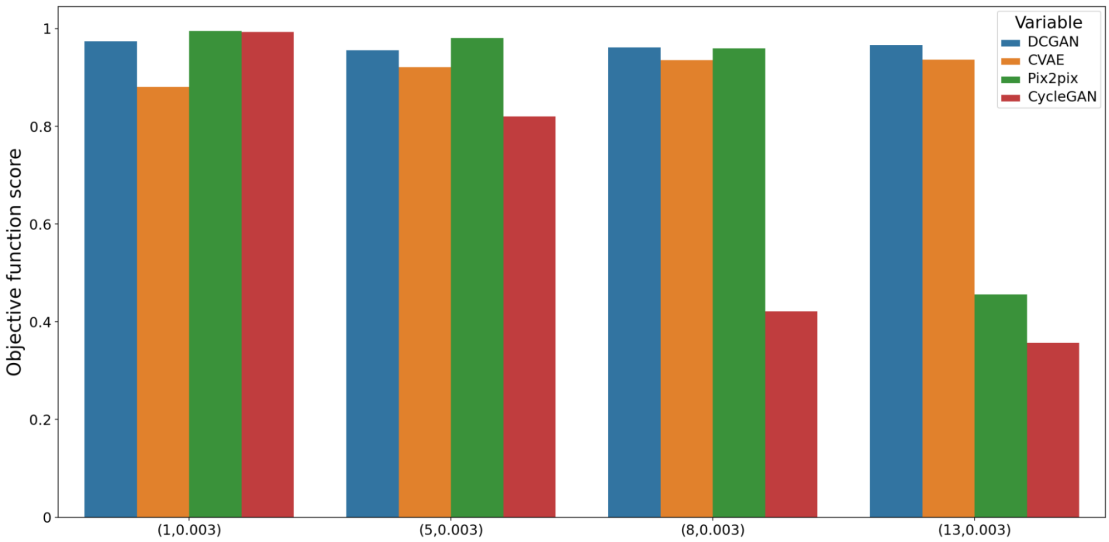

Figure 34. Comparison of studied models based on the objective function score under different signal conditions. Objective function consists of measure of object realness (based on the feedback from a well-trained discriminator of DCGAN network) and loss, based on the amount of dots crossed.

4.4 Baseline classification accuracy on the dataset

One way to see how easy it is to discern the original shape from a constellation image is to look at classifier accuracy under different signal/noise ratios (Figure 35). Having trained an ordinary classifier (CNN) on MNIST dataset, we obtain an accuracy of 0.96 on the test dataset. Next, we validate the same classifier on other test datasets with different signal/noise ratios (high signal means dots, consisting of original shape, are located close to each other, and low signal means they are more far apart). The results are reported through figure 36.

We observe a trend of classifier accuracy deterioration as task becomes more complex. Failing classifier gives a hint that in environments with low signal (where distance between dots of the original contour is high) iterative inference models are going to outperform the direct ones, as classification of the object on the picture plays an important role in the inner working of these models.

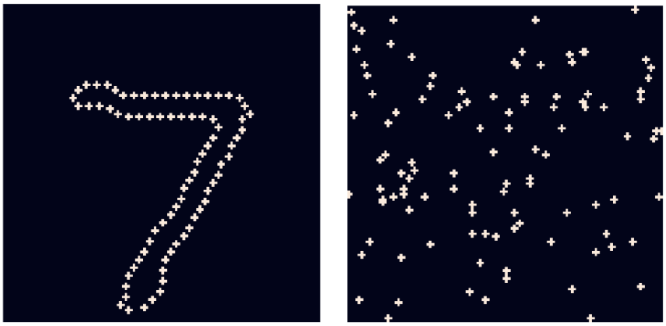

Figure 35. Constellation images with different signal/noise ratios. Left image is in high signal/no noise ratio, which corresponds to a (1, 0) condition of constellation images. Right image is in low signal/high noise ratio, which corresponds to a (15, 0.006) condition of constellation images

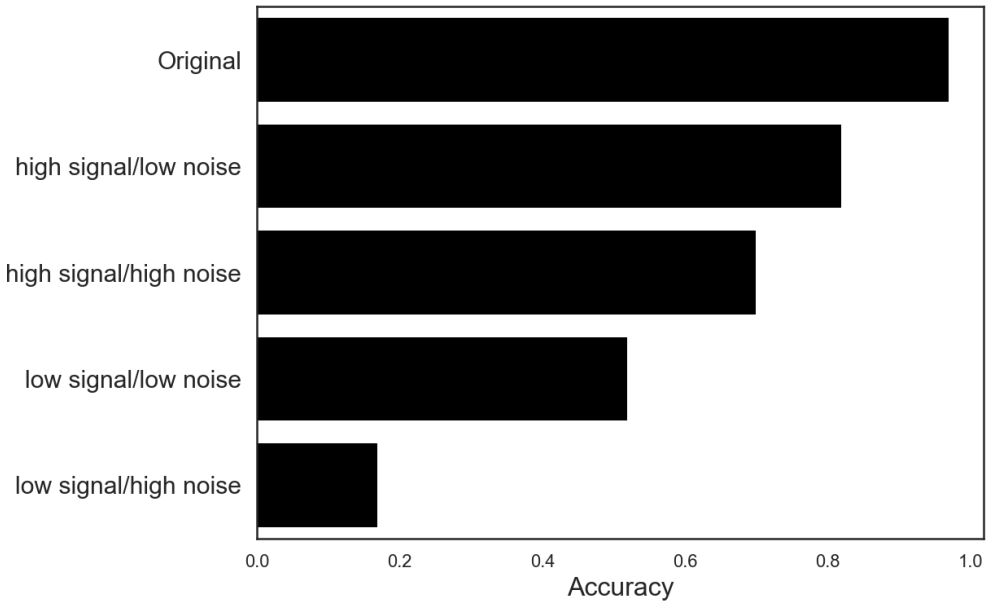

Figure 36. CNN classification accuracy under different signal/noise ratios. High signal/low noise condition corresponds to (d, noise level) = (1, 0.001). High signal/high noise condition corresponds to (1, 0.006), low signal/low noise corresponds to (15,0.001), and low signal/high noise corresponds to (15, 0.006).

5 Discussion

We have tried out two families of models to solve the constellation images problem posed by Khajuria and colleagues [KTLA22]. Pix2pix and CycleGAN have been trained on direct translation task from images of constellation modality into original images. We demonstrated under which conditions both of them work, and when they start to fail. The condition of **d** = 13 has been observed to be a point where both of them are no longer able to perform a given task.

Convolutional Variational Autoencoder and DCGAN have been trained and then used for iterative solution involving CMA-ME. They show robustness against high **d** values.

In the literature the error gap between amortized and iterative inference predictions has been called 'amortization gap' [vK20]. We report amortization gap as difference of objective function scores between amortized and iterative solutions. We observe an amortization gap of 0.55 on the constellation images of (13, 0.003) modality.

5.1 Iterative inference superiority

We observe from Figure 34 in the results section how optimized sampling from DCGAN latent space is a superior solution to the posed problem amongst all those that we have studied. Performing the same sampling with VAE works slightly worse. One reason can be that it is biased towards generating exactly the same images it *has seen* in the training. DCGAN's training happens in a setting where the generator doesn't directly see the underlying distribution, and learns only by receiving feedback loss from the discriminator network. This difference in training lets it produce images that closely resemble the underlying distribution, but are more flexible. Thus, while for many images DCGAN and VAE are both working well, there are some images that are particularly hard for VAE.

Pay attention at the images from Figure 24. While, for example, for the 2nd and 4th image a very good solution that is fitting the dots nicely has been found, in the 3rd and 6th image the found solution is not fitting dots that well. This may be referred to as just randomness of initial guess through which the solution has iterated to this. While, on the other hand, as we have encountered such inexactness of VAE generated images multiple times in training, we theorize that this behavior can be better attributed to the fact that this exact solution is not present in its latent space.

To conclude, we observe how VAE works well in the conditions where multiple solutions can be found for the same image and all be good (conditions, where noise is higher and distance between dots of original outline is high), but works worse in what could be considered an easier condition. Interestingly, on the contrary direct inference solutions we have studied work well in easier conditions, but struggle in high noise - high **d** conditions.

5.2 Direct inference models rely on classification

The dots of original shape in high noise/low signal images are located so far away that it is not trivial even for a human to discern the original shape (Figure 35). This is reaffirmed also by the classifier's accuracy failure, as it drops from 0.96 on the original test set to 0.18 on the constellation dataset in the hardest case (Figure 36).

We hypothesize that the trend of failing classifier and the trend of failing direct inference models may be closely related. At a high level, what direct inference models do can be imagined as classification of the observed shape (without explicitly understanding which class it is, of course) and then applying a network-tuned transformation, conditioned on the 'class' it has identified.

Consider the transformation applied to turn the facade drawing into a realistic image on Figure 37. Again, on the high level, we can observe that some distinct colors could be classified into different categories (brown can be classified as door, while light green as balcony) and then transformation is applied curated on specific class.

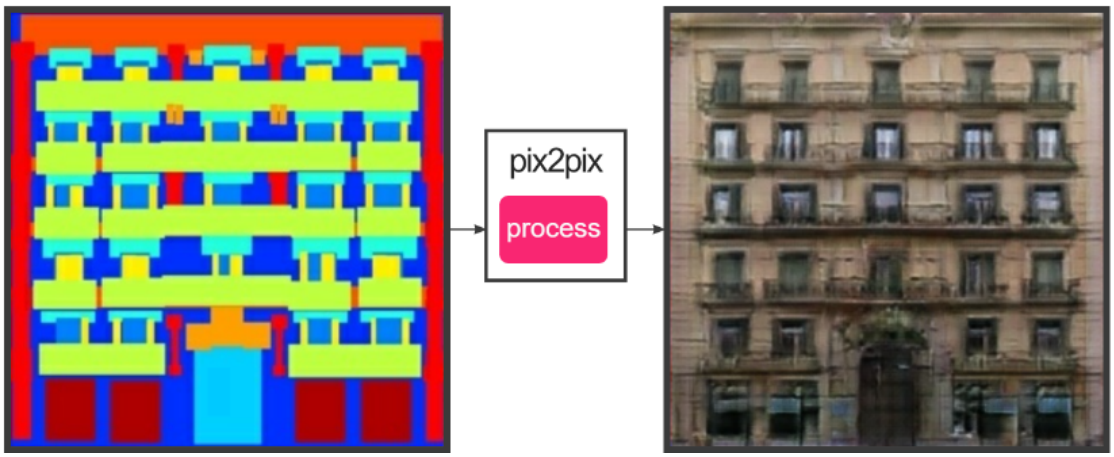

Figure 37. Semantic facade to real image translation using Pix2pix. Image generated on [Hes17].

Hence, it makes sense that direct inference models tend to fail on almost the same amount of signal/noise as a classifier. *Classification* of the parts of the input image plays a huge role in the whole dynamic of image-to-image translation, so when this classification fails (as, for example, for right image on Figure 35) the model generates unrealistic low-quality samples.

This breaking point is also the point at which we wanted to test whether machine vision algorithms are going to continue to 'denoise' the environment, trying to find what was *originally* decoded, or will try to 'daydream' about the solution, generating different shapes that also fit the dots. After the breaking point it becomes impossible to simply understand what was originally there; creative solutions are becoming the necessity in

order to produce a satisfactory result. If a dragon was encoded, but the generated shape is closer to a giraffe and satisfied the requirements, this is still considered to be a good solution.

This is exactly what is observed in the case of shapes, generated by walking through the latent space of the generator. At the time of iterating through solutions different behaviors are walked through, and this gives novelty and variability among generated images, while still satisfying the main goal even on complex images. This has also to do with the fact that iterative solutions involving generative models and optimization algorithms do not rely on classification of the original image, but just 'imagining' shapes (they can be different, and at the start they may not even fit the image that well. See Figure 27 from DCGAN results, for example) and then gradually trying to fit them into dots by shifting or imagining other, more suited shapes.

5.3 Generated images have wavy contours

The problem setup encourages shapes to have rippled contours to fit more dots, and some models are susceptible to this behavior more than others (For example, see Figure 32). This never happens with VAE images, to some extent happens with CycleGAN and pix2pix images as d increases, and happens almost always with DCGAN images.

To understand why this would happen we look at our problem setup. Each image that a shape has to be drawn on consists of a number of dots that belonged to the original outline, and a number of dots that were randomly inserted and serve as noise.

With direct inference models we have observed that with increased complexity as **d** is increasing they fail to recognize the shape (perform classification of object). As recognition has failed, they are less confident in how the outcome should look like, and tend to generate more wavy and inexact images guided by the feedback of noisy dots on the image.

For iterative inference models the situation is trickier. Throughout iterations the DCGAN model is generating guesses of the drawn shapes, and they tend to get better as the objective score is increasing. Consider Figure 38. At some point, the model has to make a new guess out of the previous solution. To simplify, it faces the following choice - should it pursue the path of rotating and increasing area of the shape, resembling 0, or alternate it into another shape, that better fits dots, or cheat by adding small lines and curves along the existing shape to better fit dots.

As it happens in real life, all choice options have different cost values. Cheating is always easier - the new solution has to change just a little by adding two small lines to the contour, already crossing more dots. On the contrary, the first two possible directions for change, although arguably superior, are harder to reach for the model as they both require a bigger leap from the existing solution.

It happens even when quite a good solution (let's say, first choice out of three in Figure 38) has already been found. Optimization algorithm pushes to increase objective

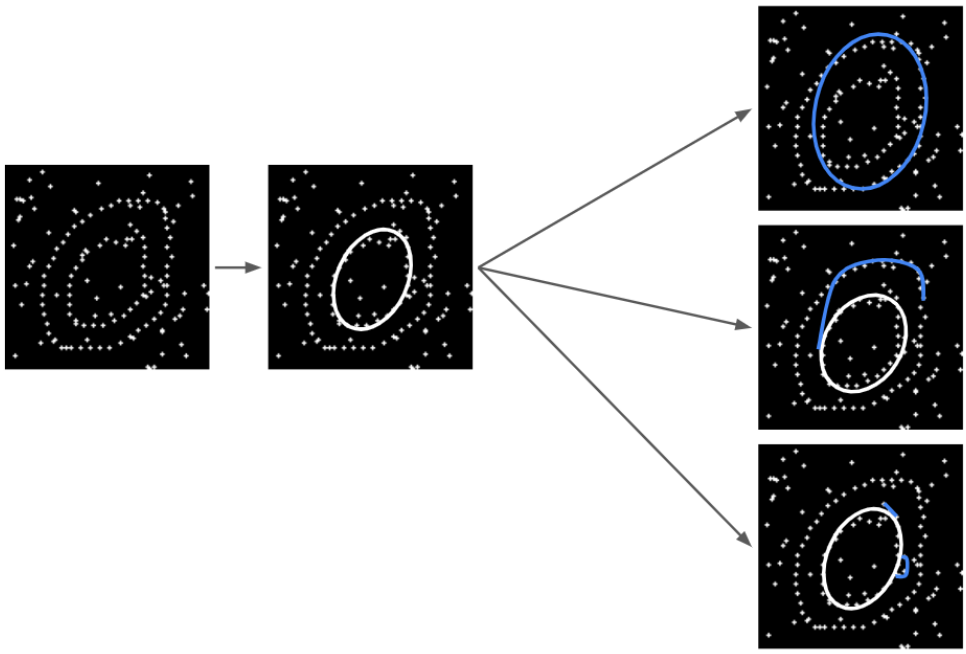

Figure 38. Situation where iterative inference model has to make a choice for the next improvement. By the blue color we indicate possible change directions for the image.

function even more, and being unable to resist it, even the best solution starts to cheat and add small tentacles out of contour in order to increase its objective score.

But why is VAE free of this bias? Well-trained autoencoder learns to generate copies of the training dataset as its loss function is based on the likelihood. This can be a drawback as by this formulation it can't really produce novel samples, but is convenient in a sense that it can't cheat - images with small artifacts here and there are just not present in its latent space. On the other hand, GAN's whole training dynamics with discriminator and generator is based on the concept of trying to cheat, or fool each other, thus the generator is able to generate quite different images with any number of artifacts on the contours.

5.4 Iterative inference issues

Iterative inference solutions seem to be superior for the given task, but this superiority comes with its own drawbacks. The main issues to be concerned about are:

1) It is generally more computationally expensive than amortized inference models as optimization algorithm has to look at and optimize the solution for a single image at a time for at least 200-300 iterations ($\approx$ 8-10 minutes), while amortized inference models can be applied to batches of images and produce the result in under 1 minute.

2) Having a generator of GAN that was properly trained. Unstable training (discriminator dominating over generator) was usually observed, and while finding right learning rate/batch size/loss schedule has helped to mitigate this issue, still training a generator that would generate realistic samples, while spanning **all** classes initially present in the dataset was sometimes an issue.

3) Samples generated by VAE are lacking the flexibility needed. This is, unfortunately, not quite an issue but an overall inherent drawback of this family of models. For some images for which VAE already has an encoded representation it generates perfect solutions, while small rotations or shifts are causing it to struggle to fit dots. Again, if we compare it to GAN - GAN's training dynamics pushes it to learn to generate similar, but not the same solutions compared to the training dataset, which makes the latent space of GAN inherently more rich to be drawing samples from than the latent space of VAE.

Nevertheless, it is much easier to properly train a VAE due to it having much fewer parameters and 'easier-to-obtain' loss. The typical autoencoder that we trained has about 160 000 trainable parameters, while the typical generative adversarial network had about 3.8 million trainable parameters.

5.5 Future work

There are multiple directions in which further improvement could be done. More sophisticated GAN architectures, such as StyleGAN [KLA18] could improve the quality of generated images in an iterative solution. StyleGAN no longer uses a noise vector as input for the first layer, but instead introduces noise injection at all layers in the neural network. This, along with other minor architecture improvements, contributes to increased quality of images, and may potentially help against 'cheating' solutions of GAN. Furthermore, VAE architectures that contain adversarial mechanics in training [BLRW17], could improve overall flexibility of VAE images, thus improving its potential in solving constellation images too.

Another possible improvement direction may lie in the way of sketching not the whole solution at once as was explored here, but by generating a series of strokes on the constellation image that lead to a shape. [HHZ19] has explored a Reinforcement Learning based approach to generate a sketch by a series of strokes. They use output of Wasserstein GAN's discriminator output as a reward to guide the sketching process of the RL agent. In [ZFW$^+$18] a different reward system has been proposed, based on the similarity of the current state of the image with the reference image. We envision that it is possible to condition a reward system based not only on image realness / image similarity, but also based on following dots of the constellation.

Current state of the 'Constellations' dataset, released by Khajuria et al, may not contain sufficient amount of images per category to train most of the deep learning models. In that sense, it may be suitable to turn attention to few-shot models, such as [RCKH20], where training is optimized to be done over limited data. Another way could

be pre-training model on overlapping categories from other, bigger datasets (such as Quick, Draw!) and do inference on the ‘Constellations’ dataset.

6 Conclusion

Modern computer vision models play an important role in understanding human vision. Successful implementations of artificial neural networks in solving specific vision tasks provide hypotheses which may then be utilized to better understand the human brain [HKSB17], [YD16].

In the current thesis we investigated two distinct models of inference: amortized and iterative inference, in an attempt to solve novel image generation task proposed by Khajuria et al. Amortized inference models were implemented by using architectures of image-to-image solutions - Pix2pix and CycleGAN. Iterative inference framework was implemented by search for a solution using optimization algorithm, CMA-ME in the latent space of generative models - Variational Autoencoder and Generative Adversarial Network. Iterative solution showed robustness against images with low amount of signal, where the amortized inference paradigm has failed to fulfill the same task. This failure may be attributed to inability to classify objects on the image with low signal of original outline. Taken together, this thesis demonstrates the potential of iterative inference and suggests that future work is required to understand how exactly amortized and iterative inference are used in the human brain.

7 Acknowledgements

In this section I would like to thank everyone who supported me through my study at the University of Tartu. I would like to thank my supervisors prof. Jaan Aru, Tarun Khajuria and Taavi Luik for valuable comments and support while I was working on thesis. I would like to thank Dmitro Fishman and Mikhail Papkov for their guidance and patience through my work at BIIT lab at the University of Tartu, which has certainly positively impacted my technical capabilities. I would like to also thank dr. Meelis Kull and prof. Raul Vicente for giving me a chance to share my knowledge with others through teaching on practical classes in the fields of Data Science and Computational Neuroscience. My study in UT was funded through Dora Plus Scholarship.

References

[ACB] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan.

[Ane22] Eugenia Anello. Denoising autoencoder in pytorch on mnist dataset, Mar 2022.

[BB21] J. Babcock and R. Bali. *Generative AI with Python and TensorFlow 2: Create images, text, and music with VAEs, GANs, LSTMs, Transformer models*. Packt Publishing, 2021.

[BLRW17] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks, 2017.

[CCVM20] Konstantinos I. Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. Quality-diversity optimization: a novel branch of stochastic optimization. *CoRR*, abs/2012.04322, 2020.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[DZR12] James J. DiCarlo, Davide Zoccolan, and Nicole C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

[EHA12] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):44:1–44:10, 2012.

[FTNH19] Matthew C. Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K. Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. *CoRR*, abs/1912.02400, 2019.

[GPAM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[Han16] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016.

[HDK+19] Martin N. Hebart, Adam H. Dickter, Alexis Kidder, Wan Y. Kwok, Anna Corriveau, Caitlin Van Wicklin, and Chris I. Baker. THINGS: A database of 1, 854 object concepts and more than 26, 000 naturalistic object images. *PLOS ONE*, 14(10):e0223792, October 2019.

[HE17] David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017.

[Hes17] Christopher Hesse. Interactive image translation with pix2pix-tensorflow, 2017.

[HHZ19] Zhewei Huang, Wen Heng, and Shuchang Zhou. Learning to paint with model-based deep reinforcement learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[HKSB17] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, July 2017.

[IZZE16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[KLA18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2018.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[KS20] Gabriel Kreiman and Thomas Serre. Beyond the feedforward sweep: feedback computations in the visual cortex. *Annals of the New York Academy of Sciences*, 1464(1):222–241, 2020.

[KTLA22] Tarun Khajuria, Kadi Tulver, Taavi Luik, and Jaan Aru. Constellations: A novel dataset for studying iterative inference in humans and ai. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[LB19] J. Langr and V. Bok. *GANs in Action: Deep learning with Generative Adversarial Networks*. Manning, 2019.

[LC10] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[LR00] Victor A.F. Lamme and Pieter R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neurosciences*, 23(11):571–579, 2000.

[LSO22] M. Lupión, J. F. Sanjuan, and P. M. Ortigosa. Using a multi-GPU node to accelerate the training of pix2pix neural networks. *The Journal of Supercomputing*, February 2022.

[MC15] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *CoRR*, abs/1504.04909, 2015.

[MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[PKJ20] Mangal Prakash, Alexander Krull, and Florian Jug. Fully unsupervised diversity denoising with convolutional variational autoencoders, 2020.

[QDW] Quality-diversity optimisation algorithms. `https://quality-diversity.github.io/`. Accessed: 2022-05-15.

[RCKH20] Esther Robb, Wen-Sheng Chu, Abhishek Kumar, and Jia-Bin Huang. Few-shot adaptation of generative adversarial networks, 2020.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[TŠ13] Radim Tyleček and Radim Šára. Spatial pattern templates for recognition of objects with regular structure. In Joachim Weickert, Matthias Hein, and Bernt Schiele, editors, *Pattern Recognition*, pages 364–374, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[vK20] Ruben S van Bergen and Nikolaus Kriegeskorte. Going in circles is the way forward: the role of recurrence in visual inference. *Current Opinion in Neurobiology*, 65:176–193, 2020. Whole-brain interactions between neural circuits.

[YD16] Daniel L K Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19(3):356–365, February 2016.

[ZFW+18] Tao Zhou, Chen Fang, Zhaowen Wang, Jimei Yang, Byungmoon Kim, Zhili Chen, Jonathan Brandt, and Demetri Terzopoulos. Learning to sketch with deep q networks and demonstrated strokes, 2018.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

Appendix

I. Access to Code

The code used to obtain the results can be found in this GitHub repository given below:
`https://github.com/faridhasanov96/constellations_master_thesis`

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Farid Hasanov**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Iterative versus amortized inference solutions to the constellation problem,
   (title of thesis)

   supervised by Jaan Aru, Tarun Khajuria and Taavi Luik.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Farid Hasanov
17/05/2022