Replicating Glow and Investigating Its Performance on OOD-detection

Simon Westberg **Tony Rönnqvist** **Fredrik Danielsson**

{swestber, tonyr, fdaniels}@kth.se

Abstract

In this paper, we implement the flow-based deep generative model Glow, which incorporates an invertible 1x1 convolution in its flow as a generalization of previously used permutation operations. We train the model on the MNIST data set and show that the 1x1 convolution achieves a lower negative log-likelihood than both a shuffling operation and a reversing operation. Additionally, we show that the trained model can generate new, realistic images, and we find that linear interpolation in latent space produces realistic images with smooth transitions. Finally, we investigate the models performance on Out-of-Distribution (OOD) detection using typicality tests and find that the model can detect OOD samples from Fashion MNIST with almost 100% accuracy for all tested batch-sizes, while for EMNIST Letters the model reaches an OOD accuracy of 75.73% when tested on batch-sizes of one and an OOD accuracy of 100% when tested on batch-sizes of 10. Our implementation of Glow is available at `https://github.com/simonwestberg/DD2412-Glow`.

1 Introduction

In recent years, deep learning has shown extraordinary results in the discriminative branch of machine learning, e.g. reaching human-like accuracy within many classification tasks where the goal is to estimate the probability $p(y|\mathbf{x})$ given some input data $\mathbf{x}$ and label y. However, discriminative modeling has some inherent drawbacks, such as its general inability to estimate the probability distribution of the input data, $p(\mathbf{x})$. To solve some of these drawbacks, one can instead turn to generative modeling, another branch of machine learning, which, given a training set $\mathcal{D}$, tries to accomplish the following fundamental goals: 1) perform density estimation given a new sample $\mathbf{x}$, i.e. estimating $p(\mathbf{x})$, 2) generate new samples $\mathbf{x}$ following the same distribution as the data in $\mathcal{D}$, and 3) learn a latent space representation of the input data. Generative modeling has many real-world applications, such as data compression [1], synthesizing speech and music [2], and data completion, e.g. semantic image inpainting [3]. Moreover, generative models can also be used for discriminative tasks, e.g. as a way to perform unsupervised pre-training of models later used for classification [4].

There are many different families of deep generative models (DGMs), the most common being flow-based models, variational autoencoders (VAEs), autoregressive (AR) models, and generative adversarial networks (GANs). In this paper, we implement Glow [5], a flow-based DGM where each step of flow is based on an activation normalization layer, an invertible 1x1 convolution operation, and an affine coupling layer. The main advantages of flow-based models are that they allow exact calculations of both latent representations and log-likelihood, as opposed to e.g. GANs and VAEs, and that they allow efficient synthesis, as opposed to AR models [5]. We train the model on the MNIST [6] data set and compare the performance of the model when replacing the 1x1 convolution with either a shuffling operation or a reversing operation. To qualitatively assess the models generative capability, we perform linear interpolation in the latent space between test images of the same class and also generate new images from the model and visually review their quality. Lastly, we use

Project report for DD2412

the trained model for Out-of-Distribution (OOD) detection using a method based on typicality [7], assessing its ability to detect OOD-samples from Fashion MNIST [8] and EMNIST Letters [9].

In our experiments, we found that the 1x1 convolution achieved the lowest negative log-likelihood out of the three compared operations. When performing linear interpolation in latent space, all the generated intermediate samples were realistic and the transition between each consecutive sample in the interpolation was smooth. We also found that most of the newly generated images from the model were realistic. In the OOD-detection experiments, we found that the model, used in combination with typicality tests, were able to classify samples from Fashion MNIST as OOD with almost 100% accuracy for all tested batch-sizes. For EMNIST Letters, the models ability to detect OOD-samples depended heavily on the used batch-size, reaching an accuracy of 75.73% when tested on batches containing single samples and an accuracy of 100% when tested on batches containing 10 samples.

2 Related work

Glow [5] builds heavily upon the work done in NICE [10] and Real NVP [11]. The flow-based approach to generative modeling was first introduced in NICE, as well as the affine coupling layer. Real NVP then expanded upon these ideas and introduced the multi-scale architecture that is reused in Glow. The main contribution of Glow is the introduction of the invertible 1x1 convolution used for permutation of the channel dimensions, as opposed to the reversing of channels that is used in Real NVP.

For OOD-detection we use a method based on the notion of typical sets [7]. A similar typicality-based method was proposed in [12], however, this method can not be used with deep models. Other recent work on OOD-detection with the use of deep generative models include [13], where the authors propose a method based on likelihood-ratios, training an additional DGM on perturbed input data and using this model to decrease the influence of irrelevant background pixels in the data when performing OOD-detection.

3 Methods

3.1 Flow-based generative models

Glow [5] is a so-called flow-based DGM, a deep generative model based on *normalizing flows*. Given a relatively simple probability distribution over the latent variables, $\mathbf{z} \sim p_z(\mathbf{z})$, e.g. a multivariate Gaussian with zero mean and unit variance, the idea is to apply a sequence of M invertible functions $\{\mathbf{g}_i\}_{i=1}^{M}$ on $\mathbf{z}$, gradually transforming the probability distribution into a more complex distribution over the input data. Letting $\mathbf{g} := \mathbf{g}_M \circ ... \circ \mathbf{g}_1$ the generative process can thus be stated as $\mathbf{x} = \mathbf{g}(\mathbf{z})$ where $\mathbf{z} \sim p_z(\mathbf{z})$. Since all $\mathbf{g}_i$ are invertible, we can define $\mathbf{f} := \mathbf{g}^{-1}$, where $\mathbf{f} = \mathbf{f}_M \circ ... \circ \mathbf{f}_1 = \mathbf{g}_1^{-1} \circ ... \circ \mathbf{g}_M^{-1}$, and thus $\mathbf{z} = \mathbf{g}^{-1}(\mathbf{x}) = \mathbf{f}(\mathbf{x})$. In each transformation $\mathbf{f}_i$, the change in probability density is given by the absolute value of the determinant of the Jacobian of the transformation, so that the log-probability of $\mathbf{x}$ can be written as [5]

$$\log p_x(\mathbf{x}) = \log p_z(\mathbf{z}) + \sum_{i=1}^{M} \log |\det d\mathbf{h}_i / d\mathbf{h}_{i-1}|, \tag{1}$$

where $\mathbf{h}_0 := \mathbf{x}$, $\mathbf{h}_M := \mathbf{z}$, and $\mathbf{h}_i := (\mathbf{f}_i \circ ... \circ \mathbf{f}_1)(\mathbf{x})$ for $i = 1, ..., M-1$. To ensure that Eq. 1 is efficient to compute, flow-based models use transformations $\mathbf{f}_i$ whose Jacobian is a triangular matrix, since the determinant of a triangular matrix is just the product of its diagonal elements.

Given a training set $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, the training objective of flow-based models is then to minimize the average negative log-likelihood (NLL) of the training data, given Eq. 1. Since Eq. 1 gives a continuous probability distribution and we are training on discrete data (the input images have discrete pixel values), one has to *dequantize* [14] the data by adding uniform noise to the pixel values. Given pixel values in the range $[0, 255]$ which are normalized to lie in $[0, 1]$, the objective to minimize is [5]

$$\mathcal{L}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} -\log p_x(\mathbf{x}_i + u) - D \cdot \log(1/256), \tag{2}$$

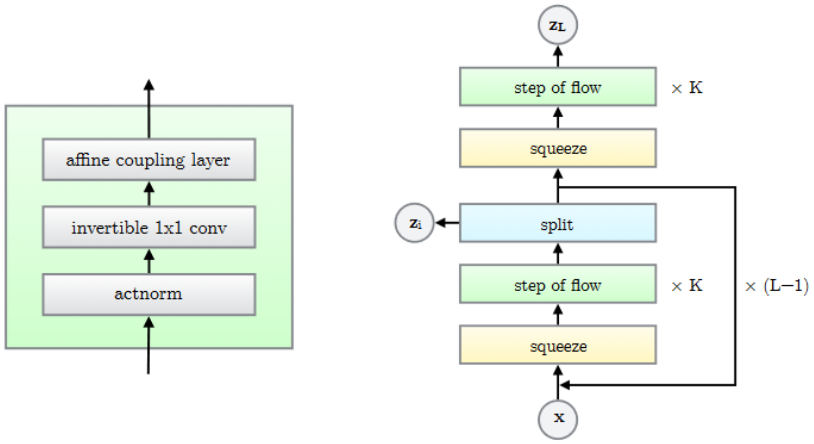


Figure 1: The definition of a single flow-step and the architecture of Glow. Image from [5].

where $u \sim \text{Uniform}(0, 1/256)$ and D is the dimensionality of the input data.

3.2 Glow

The architecture of Glow is given in Fig. 1. Given an input tensor $\mathbf{x}$ of shape $H \times W \times C$ to each layer, where H, W, and C are the height, width, and number of channels respectively, the transformations in Glow are defined as follows [5].

Squeeze The squeeze layer rearranges non-overlapping $2 \times 2 \times C$ blocks into $1 \times 1 \times 4C$ blocks, producing an output tensor of shape $H/2 \times W/2 \times 4C$. The inverse operation rearranges non-overlapping $1 \times 1 \times 4C$ blocks into $2 \times 2 \times C$ blocks.

Activation normalization The actnorm layer uses C scale parameters $\mathbf{s} = (s_1, ..., s_C)$ and C bias parameters $\mathbf{b} = (b_1, ..., b_C)$ and outputs $\mathbf{y} = \mathbf{x} \odot \mathbf{s} + \mathbf{b}$, where $\mathbf{s}$ and $\mathbf{b}$ are applied element-wise for each spatial location. The inverse operation instead calculates $\mathbf{x} = (\mathbf{y} - \mathbf{b})/\mathbf{s}$. This operation has a log-determinant of $H \cdot W \cdot \sum_{i=1}^{C} \log |s_i|$. The scale and bias parameters are initialized using an initial input mini-batch so that the output of the layer has zero mean and unit variance per channel.

Invertible 1x1 convolution This layer uses a $C \times C$ weight matrix W, which is initialized as a random orthogonal matrix, and outputs $\mathbf{y}_{i,j} = W\mathbf{x}_{i,j}$ for each spatial location i, j. The inverse operation instead outputs $\mathbf{x}_{i,j} = W^{-1}\mathbf{y}_{i,j}$. The log-determinant is given by $H \cdot W \cdot \log |\det(W)|$. This layer replaces the operation in Real NVP that reverses the order of the channels, and can be seen as a generalization of a permutation operation [5].

Affine coupling layer The affine coupling layer consists of a split along the channel dimension of the input tensor $\mathbf{x}$, producing two tensors $\mathbf{x}_a$ and $\mathbf{x}_b$ of equal shape. A non-linear mapping is then applied to $\mathbf{x}_b$ by sending $\mathbf{x}_b$ through a shallow neural network $\text{NN}()$, producing an output $(\log \mathbf{s}, \mathbf{t})$. $\mathbf{x}_a$ is then shifted and translated to produce the tensor $\mathbf{y}_a = \exp(\log \mathbf{s}) \odot \mathbf{x}_a + \mathbf{t}$. Finally, the output of the layer is given as the concatenation of $\mathbf{y}_a$ and $\mathbf{x}_b$ along the channel dimension. Since $\mathbf{x}_b$ is passed through the layer unaltered, the inverse operation is easily calculated as $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\exp(\log \mathbf{s})$, where $(\log \mathbf{s}, \mathbf{t}) = \text{NN}(\mathbf{y}_b)$, and then concatenating $\mathbf{x}_a$ and $\mathbf{y}_b$ along the channel dimension. The log-determinant of the layer is given by $\sum_i \log |s_i|$.

Split The split layer splits the input in half along the channel dimension. The first half is sent forward in the network, while the second half is modeled as a latent variable $\mathbf{z}_i$. The inverse operation instead concatenates the inputs along the channel dimension.

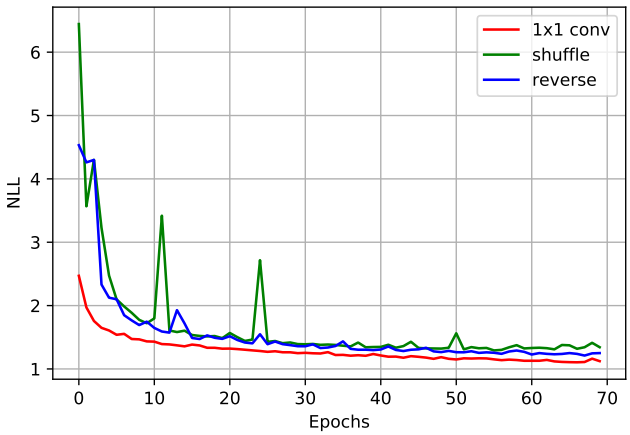


Figure 2: Average negative log-likelihood in bits/dimension of the MNIST test set as a function of number of epochs for the different permutation operations.

3.3 Additional implementation details

When inspecting the publicly available code for Glow, we noticed a few differences in implementation as compared to what was explained in the paper. In the affine coupling layer, a sigmoid-function was applied to $\log \mathbf{s}$ instead of an exponential-function. We believe this was done in order to stabilize the training. They also used a Gaussian with a learned mean and variance for the latent distribution, instead of a fixed Gaussian with zero mean and unit variance. Given an input tensor to the split layer, the tensor is first split in half along the channel dimension to produce two tensors $\mathbf{x}$ and $\mathbf{z}_i$ of identical shape. $\mathbf{x}$ is then passed through a convolutional layer and then scaled with a scale parameter per channel. The output is then used to model the mean and variance of a Gaussian used for the latent variable $\mathbf{z}_i$.

3.4 Out-of-Distribution detection

Given a deep generative model trained on a data set $\mathcal{D}$ with model likelihood $p_x(\mathbf{x})$, and a new, unseen batch of data $\widetilde{\mathcal{D}}$, the goal of Out-of-Distribution (OOD) detection is to determine whether $\widetilde{\mathcal{D}}$ follows the same distribution as the data in $\mathcal{D}$. To achieve this, one could use $p_x(\mathbf{x})$ to evaluate the likelihood of $\widetilde{\mathcal{D}}$, but this has been shown to yield subpar results for DGMs [7, 13]. Another method for OOD-detection is to use a so-called *typicality* test [7], which tries to determine if $\widetilde{\mathcal{D}}$ belongs to the *typical set* of $p_x(\mathbf{x})$, and not just to the regions of high probability density, meaning that the samples in $\widetilde{\mathcal{D}}$ "*have an information content sufficiently close to that of the expected information*" of $p_x(\mathbf{x})$ [7]. The expected information of $p_x(\mathbf{x})$ is estimated using the average NLL of the training data, $NLL_{\mathcal{D}}$, and the typicality test declare $\widetilde{\mathcal{D}}$ as OOD if

$$|NLL_{\widetilde{\mathcal{D}}} - NLL_{\mathcal{D}}| > \epsilon_{\alpha}^{M}, \tag{3}$$

for a threshold ϵ_{α}^{M}, where α is the confidence level (e.g. 0.99) and $M = |\widetilde{\mathcal{D}}|$. To determine this threshold, one uses bootstrap resampling to sample K M-sized data sets from a validation set $\mathcal{D}_{val}$ (following the same distribution as the data in $\mathcal{D}$) and evaluate Eq. 3 for each of these data sets, resulting in K values $\{\epsilon_1, ..., \epsilon_K\}$. Using these values, ϵ_{α}^{M} is determined so that a fraction α of the bootstrapped data sets would have been classified as In-Distribution, given Eq. 3.

4 Experiments and findings

4.1 Data and preprocessing

The data sets used in the experiments were MNIST [6], Fashion MNIST [8] and EMNIST Letters [9], which contain gray-scale images of hand-written numbers, clothes, and hand-written letters, respectively. MNIST was used to train the models, while Fashion MNIST and EMNIST Letters were used for OOD-detection. Since each squeeze layer halves the spatial dimensions of the input, we increased the dimensions of the images from 28×28 to 32×32 by adding zero padding to the borders, thereby allowing more layers being used in the model. The pixel values of the images were normalized to lie between $[-0.5, 0.5]$ by dividing each pixel value by 255 and then subtracting 0.5. Uniform noise was also added to the pixel values, as explained in Section 3.1. To decrease the computational complexity when training the model on MNIST, we extracted a balanced training set containing 40000 images, i.e. 4000 samples per class, from the MNIST training set. For the OOD-detection experiments, we extracted 5000 images from each test set and used another 5000 images from the MNIST test set as validation for the bootstrap sampling described in Section 3.4.

4.2 Experiments

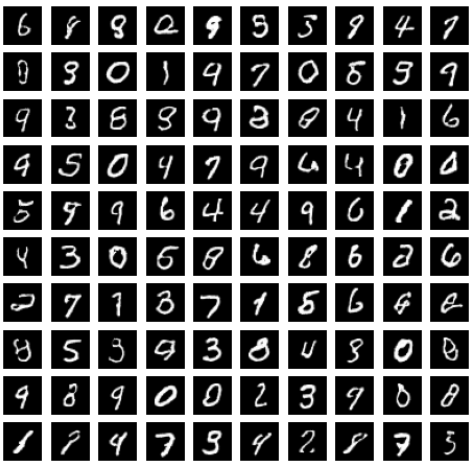
We trained a model on MNIST using $L = 3$ levels and $K = 32$ flow-steps per level (see Fig. 1). As in [5], we used a 3-layer convolutional neural network for each non-linear mapping $\mathrm{NN}()$ in the affine coupling layers, where layer one and three had a kernel size of 3 and layer two a kernel size of 1. For the two hidden layers we used 512 filters and ReLU activation. For the last layer we used the same number of filters as the number of channels of the layer input and no activation function. As in [5], the weights of the last convolutional layer was initialized with zeros. We trained the model using the Adam optimizer [15] with a learning rate of $1e-4$ for 70 epochs, using a batch size of 128.

As in [5], we compare the performance of the model when trained using the invertible 1x1 convolution described in Section 3.2 to the performance of identical models where the 1x1 convolution is replaced with either an operation that reverses the channel dimensions (as in Real NVP) or an operation that uses a fixed, random shuffling of the channel dimensions. In Fig. 2, we can see the result of this comparison. Much like in [5] we clearly see that the implemented 1x1 convolutional achieves a lower NLL than the shuffle and reverse operations, demonstrating the power of the learnable 1x1 convolutional permutation, a key result by Kingma et al. The two other operations, shuffle and reverse, achieves similar NLL, contrary to the result in Glow where shuffle outperformed reverse. This may simply be due to the fact that we only trained the models once, while in Glow they trained the models three times using different random seeds and then averaging the results for each model. Additionally, we trained the models on MNIST, a simpler data set than CIFAR-10 [16] which was used in Glow, and thus the gains obtained when replacing the reverse operation with a shuffling operation may be limited. Nevertheless, the important finding is that the 1x1 convolution indeed outperforms the other operations.

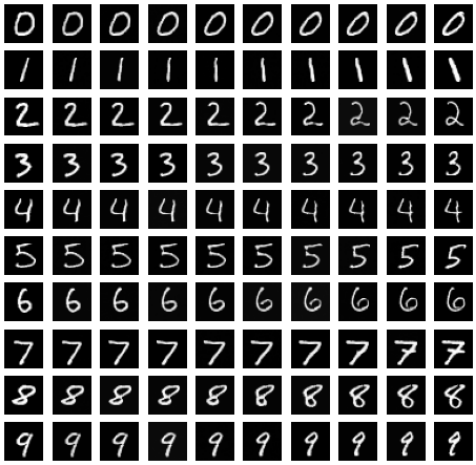
We used the model trained with the 1x1 convolution to generate new images, by sampling from the latent distribution and then running the model backwards. Similarly as in [5], we sampled new images with a reduced temperature, by multiplying the standard deviation of the latent distribution with 0.7, since we found that this resulted in slightly more realistic images. As can be seen in Fig. 3a, most of the sampled images look realistic, although there are some unrealistic samples such as the generated numbers in row 6, column 9, and in row 10, column 2.

To further assess the models generative capabilities, we also performed linear interpolation in latent space between two test images for each class. Given two images $\mathbf{x}_1$ and $\mathbf{x}_2$ of the same class, the interpolation was done by passing the images through the network to get their latent representations $\mathbf{z}_1$ and $\mathbf{z}_2$. The latent vectors were then interpolated by calculating $(1-\alpha) \cdot \mathbf{z}_1 + \alpha \cdot \mathbf{z}_2$ for different values of α, and then sent back through the network backwards to get the resulting interpolated images. The results can be seen in Fig. 3b. We can see that all intermediate, interpolated samples look like real numbers and that all transitions between consecutive samples are smooth.

To assess the models ability to distinguish between In-Distribution (ID) samples and Out-of-Distribution samples, we performed typicality tests on each test set, using $M \in \{1, 2, 10\}$, $K = 50$, and $\alpha = 0.99$. As explained in Section 3.4, the value of M determines both the size of the bootstrap sampled validation sets and the batch-size of input data at test time. The typicality tests thus determine



(a) New samples generated by the model, with temperature $= 0.7$



(b) Latent space linear interpolation. For each row, the left and right-most images are real test images, while the images in between are interpolated images.

Figure 3: Generated samples (a) and latent space interpolation (b).

Table 1: OOD results using typicality tests with $K = 50$ and $\alpha = 0.99$

	Out-of-Distribution ratio for different values of M		
Data sets	M = 1	M = 2	M = 10
MNIST	$(1.04 \pm 0.54)\%$	$(1.2 \pm 0.68)\%$	$(2.4 \pm 1.41)\%$
EMNIST Letters	$(75.73 \pm 1.74)\%$	$(91.80 \pm 2.04)\%$	$(100 \pm 0.00)\%$
Fashion MNIST	$(99.21 \pm 0.04)\%$	$(100 \pm 0.00)\%$	$(100 \pm 0.00)\%$

if entire batches of M samples are OOD or not. We ran the tests twice for each test set and calculated the mean accuracy and standard deviation over both runs. As we can see in Table 1, the model is able to classify samples from Fashion MNIST as OOD with 100% accuracy for batch-sizes $M = 2, 10$ and with 99.21% accuracy for $M = 1$, indicating that the model coupled with a typicality test is highly accurate for OOD-detection of Fashion MNIST. For EMNIST Letters, the models ability to detect OOD-samples differ greatly depending on the value of M, reaching 100% accuracy for $M = 10$, but only 75.73% for $M = 1$. These differences can most likely be explained by the similarities between certain digits and certain letters. For example, the model probably have difficulties determining if a single sample of a letter "O" is OOD, since it has similarities with the digit zero. Therefore, when $M = 1$, the model will misclassify a larger fraction of letters as In-Distribution, since it will be presented with many instances of, e.g., a single letter "O". For MNIST, we can see in the table that almost all batches are classified as In-Distribution for all values of M.

5 Challenges

As briefly discussed in Section 3.3, there were a few important details missing from the paper [5]. Without the publicly available code, we think that reproducing the *exact* results, purely based on the paper, would be nearly impossible since a few of the details seemed to be vital for their results, especially the use of a learnt mean and variance for the Gaussian latent distribution. How to preprocess the data was also a bit unclear, since the authors of Glow claim to follow the same preprocesing as in Real NVP [11], but as far as we can tell from looking at their code, they do not use the logit-function to preprocess the pixel values as explained in Real NVP.

One of the main challenges in this project was the computing power necessary to reproduce some of the results. Some of the more impressive results from Glow were done on the CelebA-HQ data set. Unfortunately, experiments on that data set were completely infeasible for us due to the amount of

GPUs and time required to train a model. Even for the relatively shallow networks we trained on MNIST, the time needed to properly train the models was quite large, and thus we did not have time to properly optimize the hyperparameters and depth of the model.

6 Conclusions

The results obtained in our experiments suggest that the 1x1 convolution achieves the best NLL out of the three permutation operations tested. However, due to time constraints we were only able to train the models once for each operation, as opposed to three times in Glow. To achieve more statistically certain results, additional testing with different random seeds should be performed.

Our model trained with the 1x1 convolution displayed satisfactory generative capabilities. The linear interpolation in latent space produced realistic images and smooth transitions. Furthermore, most of the newly generated samples were realistic. To further analyze the models generative capabilities on MNIST and perhaps improve the qualitative results, training could be performed on the entire training set (60000 images) and not only on a reduced training set of 40000 images. Additionally, one could do proper validation experiments to tune the learning rate and experiment with deeper and wider models, by increasing the level L of the model, the depth K of each flow, and the amount of hidden channels in the affine coupling layers.

Our OOD experiments seem to indicate that Glow, coupled with typicality tests, can detect OOD samples with very high accuracy when the samples are clearly visually distinguishable from the training data, as in the case of Fashion MNIST. When some of the OOD-samples resemble those of the training data, such as for EMNIST Letters, the model can accurately classify the samples as OOD when using a batch-size of 10, while it experiences difficulties when using a batch-size of 1, only reaching an accuracy of 75.73%. Something that would have been interesting to try would be to instead train the model on Fashion MNIST and then do OOD detection on MNIST, to see if the OOD results are consistent in both directions. Another extension of our experiments could be to perform OOD detection on specific classes of EMNIST Letters, to see if our assumption that the model struggles more with certain letters that resemble the digits in MNIST is true.

7 Ethical considerations, societal impact, and alignment with SDG targets

As with most developments in machine learning, the use of deep generative models raises some ethical considerations. Deep generative models can, e.g., be used to create fake, realistic media which could be used to misrepresent people. When this is applied to politicians, this could e.g. affect elections [17]. DGMs can also be used in bots to generate fake speech, e.g. as in GoogleDuplex [18] which poses as a human with the use of WaveNet [2] to carry out tasks by making phone calls. Since the listener is not made aware of the fact that they are talking with an AI they can consequently be fooled, which can be considered unethical from a consent and deception perspective [19].

As discussed earlier, one of the use-cases for generative models is Out-of-Distribution detection, which can be beneficial in many different areas where one need to be able to determine when input data differs significantly from the data that the model was trained on, in order to create safer systems. For example, as discussed in [13], OOD-detection can be useful for medical diagnosis' based on bacterial identification, since test data often contains genomic sequences that was not part of the training data. For such an application, generative models can help push the advances concerning Goal 3 of the UN SDG target [20]: to "ensure healthy lives and promote well-being for all at all ages".

8 Self assessment

We think that our project deserves an A for the following reasons. Glow, being a deep generative model, was quite hard and time consuming to implement and we needed to consult the publicly available code several times to clarify several implementation details. When consulting the code for Glow, we also discovered differences in the way they implemented the model, as opposed to what was described in the paper, and we needed to take these things into consideration when building our model. Additionally, even with the "shallow" networks we trained, the experiments were really computationally heavy, and it took several days just to perform the NLL-comparisons shown in Fig. 2. Furthermore, we measure the sensitivity towards design choices by comparing the

different permutations. Lastly, we also extend the paper by evaluating the models performance on OOD-detection. Given these experiments, we successfully implemented our project proposal.

References

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[2] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

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

[4] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[5] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31*, pages 10215–10224. Curran Associates, Inc., 2018.

[6] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[7] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using typicality. *arXiv preprint arXiv:1906.02994*, 2019.

[8] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[9] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[10] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[11] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[12] Elyas Sabeti and Anders Høst-Madsen. Data discovery and anomaly detection using atypicality for real-valued data. *Entropy*, 21(3):219, 2019.

[13] Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, pages 14707–14718, 2019.

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

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL `http://arxiv.org/abs/1412.6980`.

[16] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

[17] Nicholas Diakopoulos and Deborah Johnson. Anticipating and addressing the ethical implications of deepfakes in the context of elections. *New Media & Society*. URL `https://doi.org/10.1177/1461444820925811`.

[18] Yaniv Leviathan and Yossi Matias. Google duplex: An ai system for accomplishing real-world tasks over the phone?, 05 2018. URL `https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html`.

[19] Sandeep Chowdhury. *How do ethics influence Google Duplex's acceptance and the incoming wave of Proactive Voice Assistants?* PhD thesis, 08 2019.

[20] United Nations. The sustainable development goals report 2018. 2018. URL `https://unstats.un.org/sdgs/report/2018(visitedon08/14/2018)`.