Investigating the Sensitivity of Deep Anomaly Detection Methods to Hyperparameter Changes

Zakaria Patel, Anthony Rinaldi, Vahid Zehtab
{zpatel, arinaldi, zehtab}@cs.toronto.edu
Department of Computer Science, University of Toronto

Abstract

Machine learning models are frequently used for anomaly detection, and repurposing generative deep-learning models is a popular approach due to the unsupervised nature of this task. In this report, we explore how different architectural hyperparameters and training objectives of autoencoders, a type of reconstruction-based generative model commonly used for anomaly detection, can impact their effectiveness in detecting anomalies. Our code is made public at this Github Repository.

1 Introduction

Anomaly detection is the problem of identifying *out-of-distribution* (OOD) or unseen data. There are many classes of anomaly detection problems, but in this project, we are interested in *novelty detection*, also known as the one-class classification problem [9].

Many machine learning algorithms hinge on the assumption that the evaluation data is of the same distribution as the training data. However, in novelty detection, the goal is entirely different as we aim to learn the distribution and/or characteristics of the training data in a way that allows us to detect whether the evaluation sample comes from the same distribution or is an anomaly. One can train autoencoders on the training distribution to reconstruct normal samples and then discriminate anomalies based on their (expectedly) poor quality of reconstruction.

We expect that modifying autoencoder-based architectures would result in a non-trivial pattern when looking at the performance of the downstream anomaly detection task. We hypothesize that increasing the model complexity/capacity or adding on additional training signals will sometimes trade-off generalization but, at the same time, result in latent representations that better capture the information of in-distribution samples, resulting in more suitable models for detecting anomalies. In this report, we seek to investigate such patterns by searching through the space of different hyper-parameter settings and potential training objectives for autoencoders.

2 Background and Related Works

Approaches for tackling anomaly detection can be split into two categories: **(1) discriminative** and **(2) generative**. Discriminative approaches try to directly model $P(Y = \text{in-distribution}|X)$, where Y represents whether the sample with feature representations X, is normal or anomaly. Generative approaches, on the other hand, model the joint density $P(Y, X)$, and come up with a way of using the joint probability as a discriminative measure between normal and anomalous data.

Recently, deep learning solutions have been popular in detecting anomalies, especially in high-dimensional regimes such as images. Our observation is that all of these works incorporate a compression algorithm via architectural bottlenecks or through specific objective functions [9].

Some of the recent notable works include work done in [3], where the authors combine a very popular deep discriminative approach [6] into autoencoders. Or in [10] where the authors come up with

a compressed reconstructive-based algorithm in the feature space of pre-trained neural networks. Countless other variations of the same compressive generative models are also noticeable throughout the literature [11, 12, 7, 1, 8].

We focus on using autoencoders where an anomaly or normal data are distinguished by observing whether their reconstruction error is above or below some threshold. Autoencoders present a simple and flexible medium for exploring the problem of anomaly detection, allowing us to emphasize the problem scenario rather than the implementation details of state-of-the-art methods. We also limit our analysis to the unsupervised setting where only normal data is available for training. However, we later provide suggestions for the case where limited labeled data is available.

3 Methods

3.1 Model Architecture and Training Objective

The deep generative model we consider for this report is the autoencoder. This type of model takes in an input and sequentially condenses it down (encoder) into a bottleneck (latent) layer; from there, it sequentially increases its size again (decoder) to the original input size. More on the autoencoder can be seen in subsection A.1. We consider training MLP and Convolution based autoencoders using a plain MSE loss (Equation 1) in the pixel-space and a VAE loss (Equation 2) with a diagonal standard Gaussian as the prior distribution for the latent representation using the reparameterization trick:

$$\textbf{AE objective: } \min_{\theta,\phi} \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{P}_\text{T}} \left[L_\text{AE}\left(x, \tilde{x}_{\theta,\phi}\right)\right] \approx \min_{\theta,\phi} \frac{1}{n}\sum_{i=1}^{n} \left[\mathbf{x}^{(i)} - \underbrace{f_\theta(g_\phi(\mathbf{x}^{(i)}))}_{\tilde{x}_{\theta,\phi}}\right]^2 \tag{1}$$

$$\textbf{VAE objective: } \min_{\theta,\phi} \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{P}_\text{T}} \left[L_\text{VAE}\left(x\right)\right] =$$

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{P}_\text{T}} L_\text{AE}(x, \underbrace{f_\theta(g_\phi(x)_{\{1\cdots\lfloor\frac{\mathrm{d}}{2}\rfloor-1\}} + g_\phi(x)_{\{\lfloor\frac{\mathrm{d}}{2}\rfloor\cdots\mathrm{d}\}} \odot \boldsymbol{\epsilon})}_{\text{Reparameterization trick.}}) + \lambda \text{KLD}\left(q_\phi(z|x)||p_\text{prior}(z)\right) \tag{2}$$

Here g_ϕ, f_θ denote the encoder and the decoder parameterized by ϕ and θ, respectively. d denotes the latent size, and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}_\mathbf{d})$. $q_\phi(z|x)$ signifies the probabilistic encoder for creating the latent representation conditioned on input x, which could be written similarly to $g_\phi(x)_{\{1\cdots\lfloor\frac{\mathrm{d}}{2}\rfloor-1\}}$ and $g_\phi(x)_{\{\lfloor\frac{\mathrm{d}}{2}\rfloor\cdots\mathrm{d}\}}$ denoting the first half (mean) and the second half (variance) of the latent representation associated with input x.

Overall Model Architecture We study the ability of feedforward, convolutional, and variational autoencoders for anomaly detection. These models follow the standard autoencoder structure, but we explore variations in the encoder and decoder widths and depths (number of layers). The encoder width is the widest layer of the encoder, and the decoder width is the widest layer of the decoder. In the case of the convolutional autoencoder, the "width" refers to the maximum number of channels. Please refer to subsection A.2 for more details on the architectural setup.

Computational Block Structures Each model is comprised of blocks, which apply a linear or convolution transform and additional layers to help with training. The additional layers include ReLU/LeakyReLU activations, batch normalization, and dropout. The final encoder and decoder layers do not contain any activation or normalization. An example block can be seen in subsection A.3

3.2 Inference and Evaluation

Once a fitted model has been obtained, the next task is a classification of inputs. Predictions are made based on the average reconstruction MSE (loss) across all pixels of an image compared to some threshold. Now all that is left is to determine the threshold. In the case where no supervised data is available, we suggest taking the maximum reconstruction error across the training set as the threshold (i.e., threshold $= \max_{x_i}[\mathcal{L}(x_i)]$). Where x_i are the training examples and $\mathcal{L}(x)$ is the mean reconstruction error across all pixels of an image. We found this approach to work well under a few experimental settings. When some supervised data is available, we find that selecting the threshold that results in the upper left point on the ROC curve gives better performance. This point has the highest product of true positive rate and specificity out of any possible threshold. We suggest using this approach when limited labeled data is available, as the classification is superior.

We use AUROC to evaluate our models. This metric gives an idea of the separation between the scores of different classes. A better model will have a greater degree of separation between the scores assigned to normal inputs and the scores assigned to anomalous inputs, resulting in a better AUROC score.

4 Experiments

Dataset and Problem Setup To conduct our study, we will use the MNIST [5] dataset, which is a collection of 70,000 handwritten digits (0-9). To use this data for anomaly detection, we consider the case of *leave-one-in* training: one class of numbers (one digit) is trained on, and all other digits are only used in testing/inference. All other digits are considered anomalies, and we desire our model to only identify the single trained-on digit as normal. To push our experiments to the maximum difficulty, we selected the digit 8 as the class trained on. This digit is notoriously difficult to work with for anomaly detection [9] since it encompasses almost all other digits, so it is difficult for a model to distinguish between it and others. We performed some quick sensitivity analysis on other digits, such as 1 and 5, and noticed that it was too easy for the model to perform anomaly detection (AUROC > 0.95), so all experiments mentioned in this paper are based on digit 8.

Training We consider the completely unsupervised case where one only has access to the normal data and not to the anomalous inputs. This is applicable to the real world where it is unknown what malicious inputs may be passed to a model, and all the practitioner has access to is the correct inputs that a model should be receiving. As a consequence, there is no validation set, and no tuning of hyper-parameters is possible. We also provide suggestions for the setting where limited labeled data is available.

Since no parameter tuning is possible in this setting, we train until the learning curves plateau, which occurs for all models before our max epochs of 600. We use the Adam optimizer [4] with a learning rate of 1e-3. The learning rate was experimented with, and the chosen value resulted in the fastest, smooth convergence (see subsection A.4 for training curves).

5 Results and Analysis

The training and testing performance of each considered experiment can be found in subsection A.5. Training losses are reported for the training set of normal digits, while testing loss is reported across all digits (90% anomaly) and is, therefore, representative of the loss on anomalous digits.

Depth and Width As hypothesized earlier in the report, the width of the encoder and decoder have a significant impact on the performance of anomaly detection. As we use networks of larger width, the model can overfit to the training data more easily, allowing it to perform well only on the normal class (see Figure 1).

The effect of the depth of the network is not as clear. For the MLP autoencoder, increasing the depth of the encoder and decoder, both improve the performance of the model, although increasing the encoder depth has less of an effect when we are already using a deep encoder. For the VAE MLP, this is not the case, and increasing the depth of either the decoder or encoder marginally decreases performance. This is likely because regularization discourages the model from overfitting to the normal class, therefore promoting generalization. For convolutional autoencoders, increasing the depth always increases the performance. In this case, the decoder depth has a more significant impact than that of the decoder for the vanilla MLP. We hypothesize that this is because if the decoder is larger than a particular threshold, it can reproduce more intricate reconstructions, which could potentially generalize to the OOD samples.

In the VAE CNN, we observe yet the same trend; however, increasing the encoder depth here has less of an impact (and sometimes a negative impact) when the decoder depth is small. This can be explained by the encoder condensing information too much for the decoder to upsample in only a few layers. When the decoder is deep, it is fine for the encoder to compress information more. This means that in the CNN autoencoder case, the decoder is the more important factor, and having a deep decoder can substantially improve the performance of the model, regardless of the encoder's depth.

Architecture The results show that the feedforward autoencoder in general, performs better than its convolutional counterparts. This is the opposite of our expected results since convolutional networks

tend to work better with image data. However, this is precisely what makes convolutional networks undesirable for anomaly detection. As they are able to reconstruct anomalous digits well, their reconstruction losses are not as high as the MLP network (as seen by its lower test loss compared to MLP experiments). We also notice that the convolutional networks have magnitudes more parameters but do not overfit to the normal class as we would want a model to do for anomaly detection. Although the MLP has fewer parameters in general, it is able to overfit to the normal input quite well and produces very noisy reconstruction for anomalous classes (subsection A.6).

Variational The variational MLP and convolutional counterparts are not able to achieve as low of a test reconstruction error since their objective is mixed between reconstruction and the KL-divergence. We would expect that this results in better performance since the test reconstruction loss is inversely related to the AUROC score. However, we see lower AUROC for the variational autoencoders and explain this through regularization. VAE loss is essentially the reconstruction loss with regularization on the latent variable, preventing the model from overfitting the training data, which we have pointed out previously is needed for an anomaly detection autoencoder. Therefore, VAEs do not perform as well compared to their vanilla autoencoder counterparts.

Test Loss vs. AUROC The above findings inspired us to explore the relationship between test reconstruction loss and AUROC (subsection A.7). We found a general negative trend between the two. As a model is able to reconstruct the test data well, it cannot separate examples between normal and anomalous. For convolutional models, we observe that if they are fully trained, they can reconstruct images too well (even anomalous ones), so we compare the AUROC to the *trainedness* of the model. The test loss is a proxy for the *trainedness* of the model, and we observe that after the initial stages of high test loss due to random model initialization follows the best AUROC scores. A partially trained model (as determined by test loss) is thus more suitable than a model that is fully trained on the task.

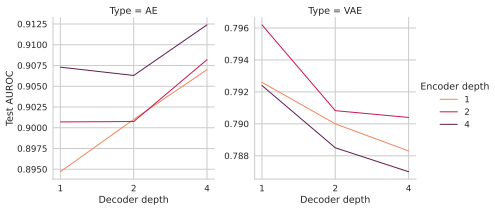
Dropout While we initially included dropout to prevent overfitting, we observed that setting it to 0 led to better performance. We hypothesize that overfitting, while detrimental to the task of reconstruction, is beneficial for anomaly detection to some degree. Allowing overfitting reduces the ability of the model to generalize to other digits, which we expect the model to have worse reconstruction on. We observed that with dropout, the model was able to reconstruct even anomalous inputs relatively well, which limits the classification power of the model. On the other hand, removing dropout led the network to produce extremely noisy outputs for anomalous samples, therefore making them easier to detect as anomalies (subsection A.6).

Reconstruction Images We previously noted that the convolution networks can easily reconstruct images it has not seen before (see subsection A.8). However, the MLP networks cannot do such, and it mainly creates noise. The MLP tends to reconstruct one into eights, possibly because one can be seen as a very narrow eight. We also saw in the reconstruction images that anomalous inputs were reconstructed into noise during earlier timesteps, supporting our claim that stopping the model before it is fully trained would be beneficial for anomaly detection.

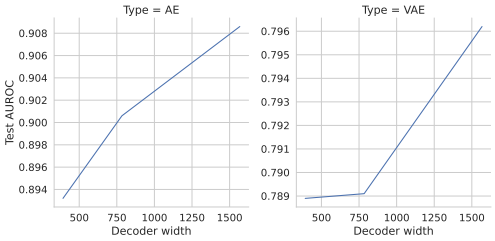
6 Conclusion

Anomaly detection for the unsupervised setting is quite powerful using autoencoders. Exploring the network architecture settings, we find that MLPs benefit from scaling the depth and width of the network, while CONVs benefit mainly from scaling the decoder depth. Adding the variational KL-divergence objective does not benefit the model performance in any way since regularization is not beneficial to autoencoder anomaly detection. Using MLPs will tend to result in better performance since they can more easily overfit to the data, which is somewhat needed for this task. Finally, we note that training a model to convergence is not ideal for this task, and more research will need to go into identifying the optimal stopping point in training.

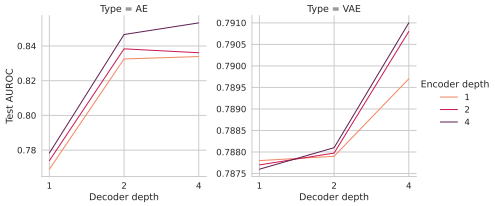
Future work Some possible extensions for future work include analyzing more difficult image datasets such as CIFAR-10, varying other model parameters such as the latent dimension, looking into how the interaction between width and depth scaling affects model performance, and the optimal early stopping point for a model to be used in anomaly detection.



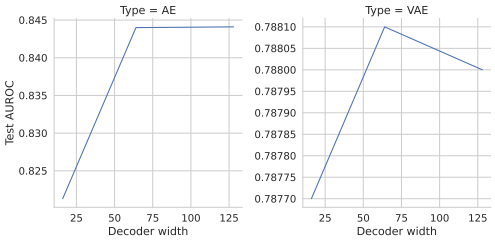
(a) MLP depth comparison



(b) MLP width comparison



(c) Conv depth comparison



(d) Conv width comparison

Figure 1: Model AUROC comparison for all experimental settings

References

[1] Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Latent space autoregression for novelty detection. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] S. Ben Driss, M. Soua, R. Kachouri, and M. Akil. A comparison study between MLP and convolutional neural network models for character recognition. In Nasser Kehtarnavaz and Matthias F. Carlsohn, editors, *Real-Time Image and Video Processing 2017*, volume 10223, page 1022306. International Society for Optics and Photonics, SPIE, 2017.

[3] Hadi Hojjati and Narges Armanfard. Dasvdd: Deep autoencoding support vector data descriptor for anomaly detection, 2021.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Yann LeCun and Corinna Cortes.

[6] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 10–15 Jul 2018.

[7] Mohammadreza Salehi, Atrin Arya, Barbod Pajoum, Mohammad Otoofi, Amirreza Shaeiri, Mohammad Hossein Rohban, and Hamid R Rabiee. Arae: Adversarially robust training of autoencoders improves novelty detection. *Neural Networks*, 144:726–736, 2021.

[8] Mohammadreza Salehi, Ainaz Eftekhar, Niousha Sadjadi, Mohammad Hossein Rohban, and Hamid R Rabiee. Puzzle-ae: Novelty detection in images through solving puzzles. *arXiv preprint arXiv:2008.12959*, 2020.

[9] Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv preprint arXiv:2110.14051*, 2021.

[10] Mohammadreza Salehi, Niousha Sadjadi, Soroosh Baselizadeh, Mohammad H Rohban, and Hamid R Rabiee. Multiresolution knowledge distillation for anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14902–14912, 2021.

[11] Et al. Schlegl, Thomas. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *Information Processing in Medical Imaging*, pages 146–157, Cham, 2017. Springer International Publishing.

[12] Chong Zhou and Randy C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 665–674, New York, NY, USA, 2017. Association for Computing Machinery.

A Appendix

A.1 Autoencoders

The architecture of an autoencoder comprises of two primary components: an encoder and a decoder. In the case of a vanilla autoencoder, both of these are feedforward networks. The encoder maps the input data to a lower-dimensional latent representation, while the decoder maps the latent representation back to the original input data. The encoder and decoder are trained together using some kind of reconstruction loss, such as mean squared error, to ensure that the reconstructed output is as similar to the input as possible. The MSE reconstruction loss is:

$$\mathcal{L}(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^{n} ||x_i - \hat{x_i}||^2 \tag{3}$$

where x is the input data, $\hat{x}$ is the reconstructed output, and n is the number of samples in the dataset. Minimizing this loss requires that the network learns a good compressed representation of the input, capturing the most essential features for reconstruction.

Convolutional autoencoders are an extension of vanilla autoencoders specifically designed for image processing. They use convolutional layers in the encoder and decoder to capture spatial relationships in the input data, making them a natural fit for image tasks. Their architecture is similar to that of a vanilla autoencoder, replacing the dense layers with convolutional layers. The loss function for a convolutional autoencoder is identical to a regular autoencoder.

Variational autoencoders (VAEs) are yet another type of autoencoder that not only learn compressed representations of data, but also a probability distribution over the compressed representations. Unlike vanilla autoencoders and convolutional autoencoders, VAEs are trained to generate new data that is similar to the original data, rather than just reconstructing it. This is done with the introduction of the Kullback-Leibler (KL) divergence loss.

The encoder of a VAE maps the input data to a mean and a standard deviation vector, which are used to define a multi-variate Gaussian distribution over the latent space. The decoder then maps samples from the latent space back to the original data. The objective of a VAE is to minimize the reconstruction error, as well as the KL divergence between the learned distribution over the latent space and a prior distribution, which is typically a standard normal distribution. The loss function for a VAE can be written as:

$$L = -\mathbb{E}_{z \sim q(z|x)}[\log p(x|z)] + \lambda \mathrm{KL}[q(z|x)||p(z)] \tag{4}$$

where x is the input data, z is a sample from the latent space, $p(x|z)$ is the probability of generating the input data given the latent variable z, $q(z|x)$ is the distribution over the latent space given the input data, $p(z)$ is the prior distribution over the latent space, and λ is the strength of the regularization applied by the KL-divergence in the loss.

A.2 Autoencoder Architecture Setup

Multi-Layer Perceptron We study the ability of a feedforward autoencoder for anomaly detection. This model follows the standard autoencoder principle, but we explore variations in the encoder and decoder widths and depths, respectively. The encoder width refers to the widest layer of the encoder (at the beginning of the encoder), and the decoder width refers to the widest layer of the decoder. In a standard autoencoder, these widths are equal and limited by the input image dimensions, but we overcome this limitation by adding two projection layers: the first at the start of the encoder to project the input dimensions to the desired encoder width, and the second at the end of the decoder to project from the decoder width to the output dimension. The depth refers to the number of layers in either the encoder or decoder.

Convolutional We also explore the ability of convolutional neural networks for the model architecture, which have been shown to work better for image data [2]. The model receives an image as an input and applies convolutions in succession to reduce the spatial dimension of the image and increase the channel dimension of the image up to the encoder width. After the last convolution of the encoder, the image is flattened into its latent representation. Passing the latent representation to the decoder, the vector is reshaped into an image (3-dimensional tensor) and upsampled using transpose convolutions to reduce the channel dimension and increase the spatial dimension back to the original image shape.

Variational For the variational MLP/CONV autoencoder we utilize the same architecture, with the only difference being in the latent layer where we split the latent representation into mean (μ) and variance (σ) vectors. These vectors are used to sample from a multi-variate Gaussian distribution which is then passed to the decoder (using a reparameterization trick for training).

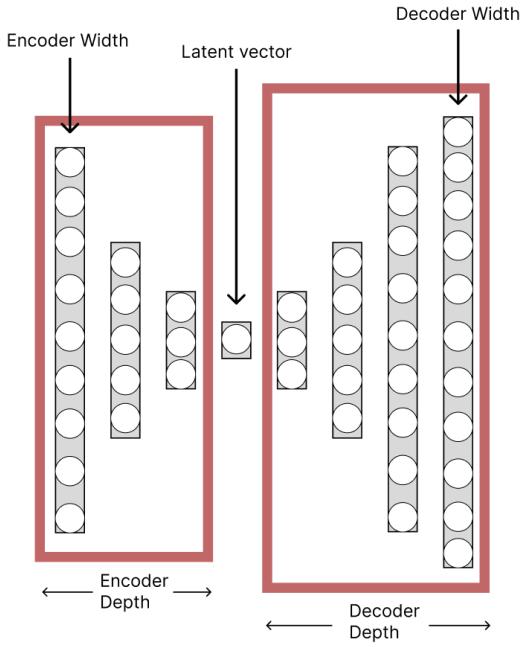


Figure 2: Example architecture of the feedforward autoencoder

A.3 Block Structure

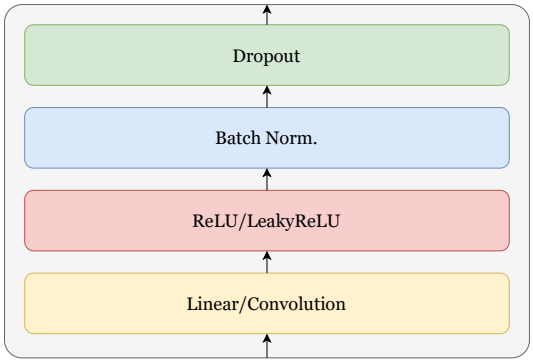


Figure 3: Block structure used in each layer of network

A.4 Training Loss Curves

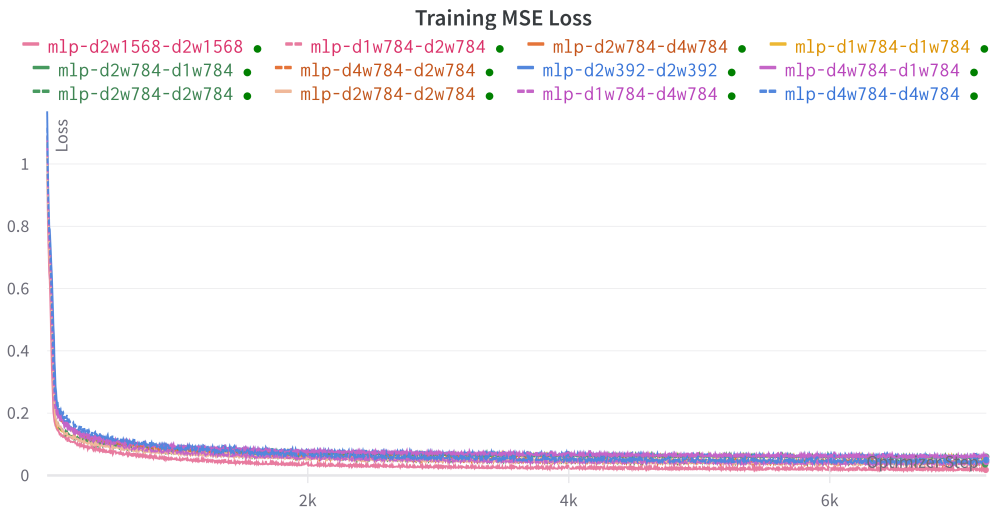


Figure 4: MSE training loss curves for all MLP (non-VAE) experiments

A.5 Detailed Experimental Results

The tables in this section outline all the results of the experiments for each model architecture. Some columns are omitted from the VAE tables, as they are redundant (VAEs follow the same architecture as their vanilla counterparts).

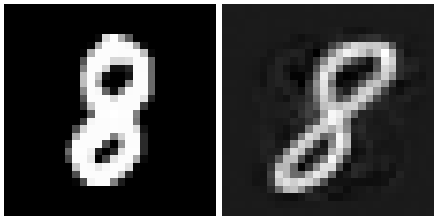
Convolutional Autoencoder								
Encoder		Decoder		Latent Dimension	Parameters	Train Loss	Test Loss	AUROC
Depth	Width	Depth	Width					
2	16	2	16	32	51.1K	0.05666	0.1614	0.8213
2	64	2	64	32	204K	0.03437	0.1775	0.844
2	128	2	128	32	408K	0.02233	0.2041	0.8441
1	64	1	64	32	815K	0.07445	0.4293	0.769
1	64	2	64	32	505K	0.03134	0.2201	8325
1	64	4	64	32	410K	0.03951	0.2136	0.8339
2	64	1	64	32	514K	0.07431	0.2475	0.7739
2	64	2	64	32	204k	0.03437	0.1775	0.8440
2	64	4	64	32	109K	0.03862	0.1701	0.8361
4	64	1	64	32	402K	0.07876	0.1704	0.7783
4	64	2	64	32	112K	0.03891	0.1655	0.8466
4	64	4	64	32	17.1K	0.04197	0.1616	0.8533

Convolutional VAE							
Encoder		Decoder		Train Loss	Test Loss	Test MSE	AUROC
Depth	Width	Depth	Width				
2	16	2	16	0.6116	0.7937	0.7928	0.7877
2	64	2	64	0.6128	0.8007	0.7932	0.7881
2	128	2	128	0.6178	0.8153	0.7933	0.788
1	64	1	64	0.6826	0.9242	0.7931	0.7878
1	64	2	64	0.6682	0.9422	0.7929	0.7879
1	64	4	64	0.666	0.9215	0.7923	0.7897
2	64	1	64	0.6128	0.8	0.7932	0.7877
2	64	2	64	0.6128	0.8007	0.7932	0.7881
2	64	4	64	0.613	0.8	0.7938	0.7908
4	64	1	64	0.6171	0.794	0.7931	0.7876
4	64	2	64	0.6167	0.7939	0.7931	0.7881
4	64	4	64	0.6116	0.7942	0.7932	0.791

Feedforward Autoencoder								
Encoder		Decoder		Latent Dimension	Parameters	Train Loss	Test Loss	AUROC
Depth	Width	Depth	Width					
2	392	2	392	32	2.4K	0.05679	0.2677	0.8932
2	784	2	784	32	4.7K	0.03536	0.3037	0.9006
2	1568	2	1568	32	9.4K	0.01627	0.3156	0.9086
1	784	1	784	32	3.1K	0.05525	0.2685	0.8947
1	784	2	784	32	3.9K	0.03618	0.3069	0.901
1	784	4	784	32	4.5K	0.04243	0.3426	0.907
2	784	1	784	32	3.9K	0.05484	0.2779	0.9007
2	784	2	784	32	4.7K	0.03536	0.3037	0.9006
2	784	4	784	32	5.3K	0.03354	0.3319	0.9082
4	784	1	784	32	4.5K	0.05615	0.2881	0.9073
4	784	2	784	32	5.3K	0.03751	0.3064	0.9063
4	784	4	784	32	5.9K	0.03875	0.3585	0.9124

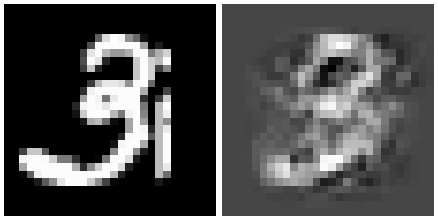
Convolutional VAE							
Encoder		Decoder		Train Loss	Test Loss	Test MSE	AUROC
Depth	Width	Depth	Width				
2	16	2	16	0.6129	0.7947	0.7941	0.7889
2	64	2	64	0.6134	0.7987	0.7941	0.7891
2	128	2	128	0.6149	0.8046	0.7974	0.7962
1	64	1	64	0.6125	0.9442	0.7953	0.7926
1	64	2	64	0.6128	1.787	0.7941	0.79
1	64	4	64	0.6147	0.955	0.7952	0.7883
2	64	1	64	0.6119	0.8016	0.7954	0.7962
2	64	2	64	0.6134	0.7987	0.7941	0.7891
2	64	4	64	0.6157	0.8004	0.7965	0.7904
4	64	1	64	0.6117	0.7955	0.7953	0.7924
4	64	2	64	0.6129	0.7938	0.7935	0.7885
4	64	4	64	0.616	0.7955	0.7953	0.787

A.6 MLP Digit reconstruction



(a) Input image (b) Reconstruction

Figure 5: Best reconstruction achieved on test data using the best MLP autoencoder architecture



(a) Input image (b) Reconstruction

Figure 6: Worst reconstruction achieved on test data using the best MLP autoencoder architecture

A.7 Test Loss vs. AUROC

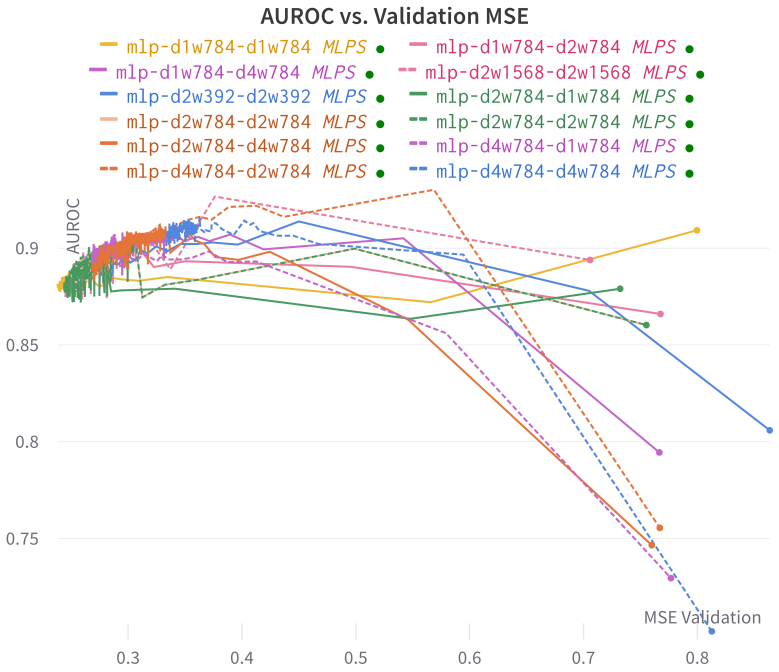
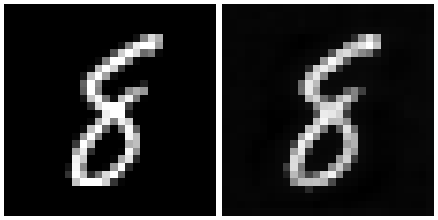


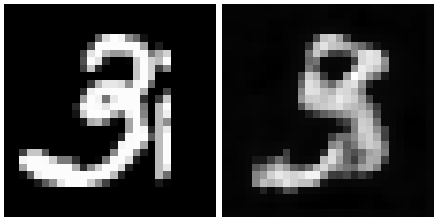
Figure 7: Test set MSE versus AUROC for all MLP experiments

A.8 CONV Digit reconstruction



(a) Input image (b) Reconstruction

Figure 8: Best reconstruction achieved on test data using the best CONV autoencoder architecture



(a) Input image (b) Reconstruction

Figure 9: Worst reconstruction achieved on test data using the best CONV autoencoder architecture