

A COMPARISON OF GENERATIVE NETWORK OUPUTS FOLLOWING DIMENSIONALITY REDUCTION



Victoria Shi 992442
NEUR30006: Real and Artificial Networks

Introduction

Artificial Neural Networks have provided many applications for research into aspects of information processing. The advantage of ANNs over biological brains is that components of processing can be pulled apart and investigated in ANNs, near impossible in a brain which processes in milliseconds with no way to pause and dissect it. This paper will focus on two main techniques: dimensionality reduction and generative outputs.

Dimensionality Reduction is a technique that forces the retention of only the minimal, most significant aspects of data. What are the minimum essentials that allow a network to learn?

The latent space can be described as a relative representation of the most important features of the original dataset. Forcing the algorithms to ‘compress’ and remove extraneous information generates an abstract space where each point holds the ‘essence’ of the full concept in the data. It may also show how distinguishable features are, for example more similar features are grouped closer together, for example, all the ‘1’s would sit close together in the latent space, perhaps closer to ‘7’s but further from ‘3’.

Generative Outputs. ANNs are not restricted to merely learning but capable of ‘reconstructing’ what they have learnt. The outputs of their regenerative abilities represent what the network has *retained* from the training data. This retention of the most significant features can be thought of as ‘mapped’ somewhere within the latent space.

The basic versions of three algorithms are summarised below:

	Autoencoders (AE)	Principal Components Analysis (PCA)	Generative Adversarial Networks (GAN)
Basic Structure	Neural Network with hidden layers reducing in number of nodes; then reconstructing	Machine Learning Algorithm that represents multivariate data with a smaller set of variables *Not a neural network	Two Neural Networks, a Discriminator that tells if the input is real or fake, a Generator that learns to create output more similar to real to deceive the Discriminator. Eventually, the Generator can create new examples from random noise
Learning Style	Self-supervised learning (ie. does not compare to target label, but rather trains on loss between original and reconstruction)	Unsupervised	Unsupervised (but partly supervised – learns and adapts via success of deceiving Discriminator)
Latent Space	1-dimension latent space	1-dimension latent space	Multi-dimension latent space

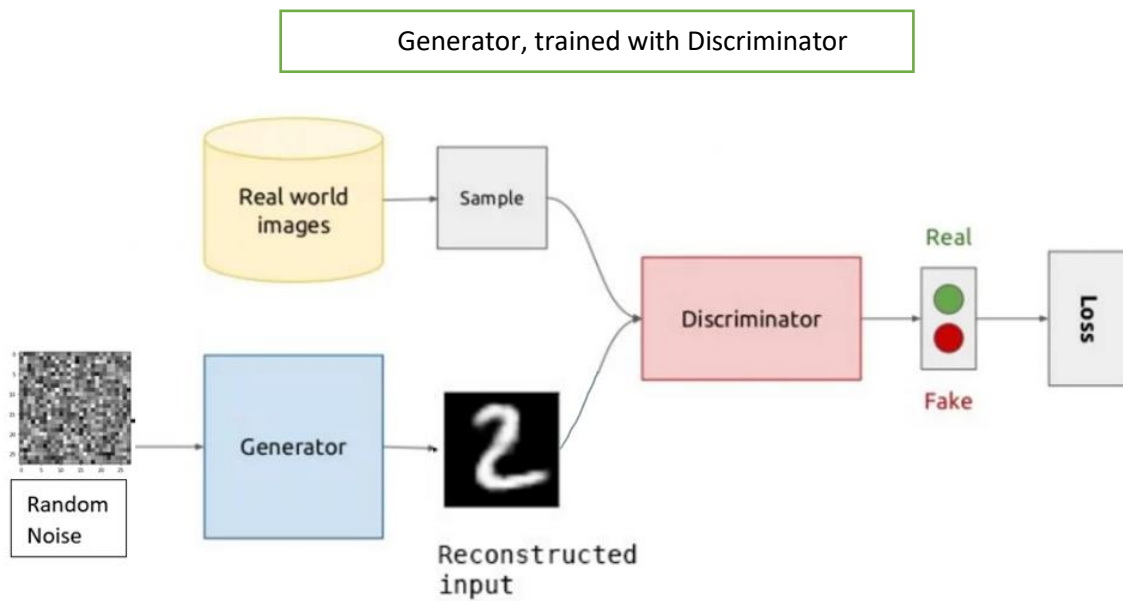
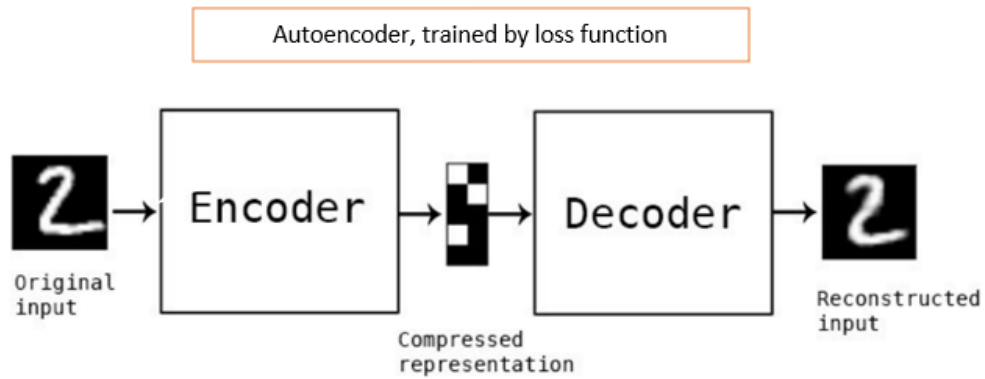
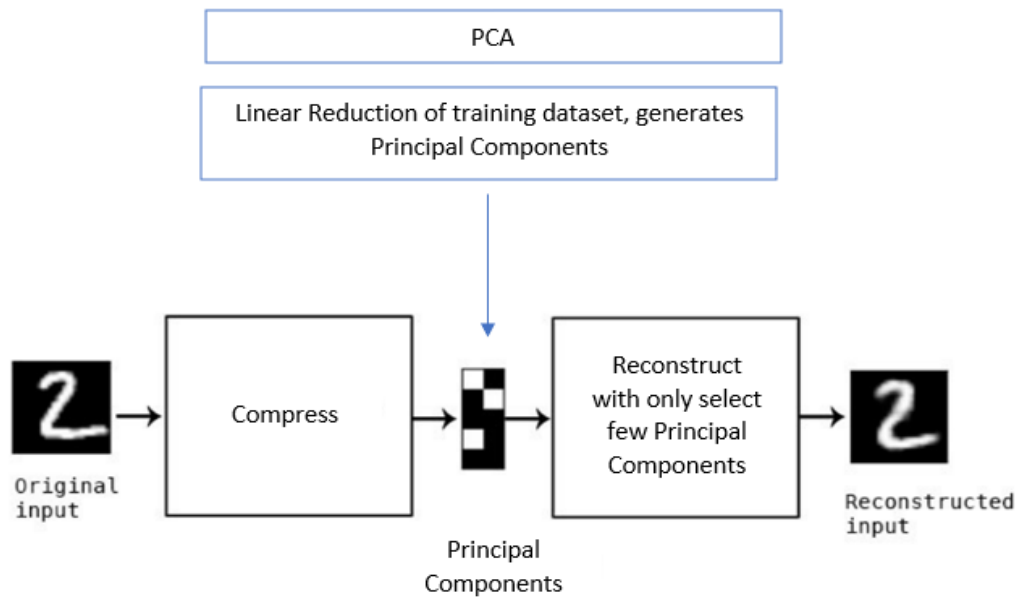


Figure 1: Schematics showing basic structure used in programs

Given the depth of understanding of the biological visual sensory modality (Hutmacher, 2019) and the comparatively simpler implementation of visual input in machine computation, investigation of the machine visual learning process may be a suitable starting point in the quest to understand higher cognitive processing.

To compare the outputs of the three algorithms above, this study will take advantage of the well-established MNIST dataset and relevant infrastructures already developed. All have different objectives in order to compressed data. The GAN learns by the success rate of fooling the discriminator, and slowly builds up a latent space. The PCA and AE methods generally build their latent space through mathematical fitting of training data, extracting the most significant patterns.

This study will qualitatively and quantitatively explore the reconstructed outputs of the algorithms to examine whether common features are present despite the different learning patterns. Such features may shed light on significant elements in a simple visual field that are essential for classification and learning. Outputs at various training stages can also be compared, to see how the machine develops the representation of its understanding.

Method

Summary of Programs used and their CSV outputs, each denoted with a **code** corresponding to respective python and CSV files.

Programs	Outputs	Explanation
M1 PCA	M1.3_pca_data	Output captured with application of PCA to test dataset with 64 principal components
	M1.4_pca_dimensions	Output captured with application of PCA to a test dataset at various dataset [3, 6, 8, 12, 32, 64, 128]
M2 AE 4-layer NN 784 > 128 > 64 > 12 > 3 nodes	M2.3_ae_data_epochs	Output captured at each of the 10 epochs during training
	M2.5_ae_data	Output at the end of training – 10 epochs
M3 GAN 2-layer NN 784 > 200 (normalised) >1	M3.3_gan_data_epochs	Output captured at each of the 10 epochs during training
	M3.5_gan_data	Output at the end of training – 10 epochs
S1 Comparison	S1.2_noise	Collection of random noise for control
	S1.3a Pairwise PCA sum of pixel difference	Compares PCA output at 64 dimensions (M1.3) one-to-one with original
	S1.2b Crosswise PCA sum of pixel difference	Compares PCA output at 64 dimensions (M1.3) one-to-many with original
	S1.3c Pairwise AE sum of pixel difference	Compares AE output at end of training (M2.5) one-to-one with original
	S1.3d Crosswise AE sum of pixel difference	Compares AE output at end of training (M2.5) one-to-many with original
	S1.3e Crosswise between AE and PCA	Compares AE output (M2.5) and PCA output (M1.3)
	S1.4a Noise comparison PCA	Compare PCA outputs (M1.3) to noise (S1.2)
	S1.4b Noise comparison AE	Compare AE outputs (M2.5) to noise (S1.2)

S2 Performance on Vanilla NN	S2.2a PCA performance	Tests how well the Vanilla NN recognises PCA reconstruction (M1.3)
	S2.2b AE performance	Tests how well VNN recognises AE reconstruction (M2.5)
	S2.2c GAN performance	Tests what numbers VNN thinks GAN outputs (M3.5) are – no labels so cannot check performance

The main programs

One program for each of the techniques were used: PCA (M1), AE (M2) and GAN (M3). Code was adapted from existing programs and modified to control input and output. The main dataset used was the MNIST handwritten set, sourced from Rashid's MNIST CSV files. All three codes use this same MNIST input and are loaded into the coded in the same manner, Rashid's method of loading into a pandas pd.dataframe. This allowed their outputs to be saved into CSV files with the same format for easier comparison and testing.

Parameters used to produce the main data for the neural networks (AE and GAN) are summarised below. These parameters were selected based on source code and various articles (Hui 2018, Rashid 2020, CS Toronto). Full specificities can be found in the accompanying python files:

Parameter	AE	GAN
Activation Function	ReLU	Leaky ReLU
Loss Function	Mean Squared Loss	Binary Cross Entropy
Optimiser (eg. Stochastic gradient descent)	Adam	Adam
Layers	Input 784 > 128 > 64 > 12 > 3	Input 784 > 200 (normalised) > 3

Output collection

Two types of CSV files were saved from each of the three programs:

1. 10 reconstructions per various points during learning (M1.4, M2.3, M3.3)
2. 100 samples of reconstructions after training was complete (M1.3, M2.5, M3.5)

Several variables such as number of epochs, number of dimensions, learning rate were tried and tested to produce outcomes for qualitative comparison.

The secondary programs

Two programs were created to conduct analysis on the saved outputs from the main programs.

Comparison (S1): Quantitative comparison of pixel difference between original inputs and reconstructed outputs, with various statistical analyses conducted.

Average sum of pixel-by-pixel differences. 100 reconstructed images each for both PCA and AE (M1.3, M2.5) were compared to the original input data. The control compared originals to noise. This process was also done between the 100 reconstructed outputs of AE and PCA. As GAN images were generated from random noise, they were unable to be included in this comparison.

Pair-wise Comparison: For every reconstructed picture of PCA and AE, there was a corresponding original image before dimension reduction. Each pair were compared by subtracting

every correlated pixel eg. the value at coordinate [0,0], and the differences at every pixel were summed per pair. Pair-wise comparison shows the reproducibility of the algorithms.

If the pictures were exactly the same, this would produce a difference of 0. The maximum value a pixel could differ by is 1.0, so the maximum sum of difference possible is 784.

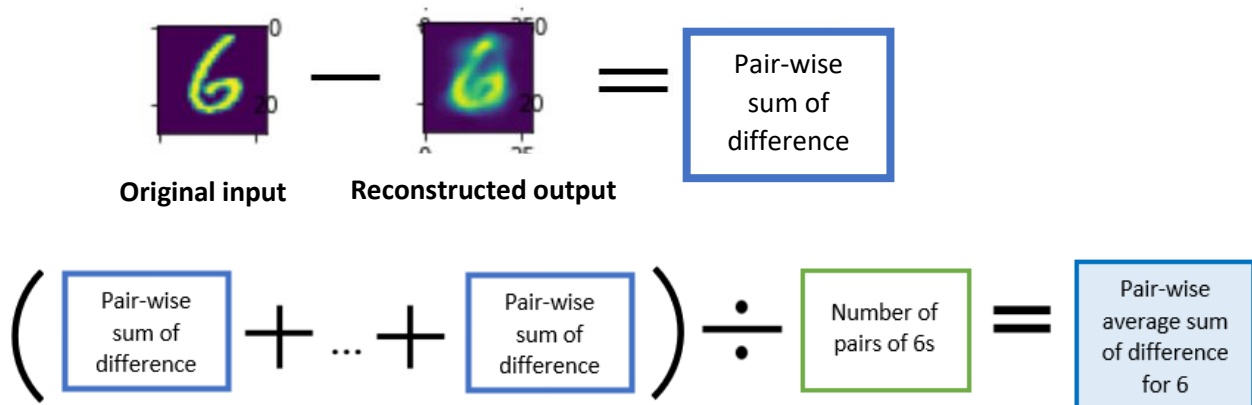


Figure 2: Visual representation of pair-wise comparisons

Cross-wise Comparison: One-to-many comparison between a single original input and *all* of the reconstructions of the same digit. Cross-wise comparison allows the general ‘essence’ of a number be compared to the original.

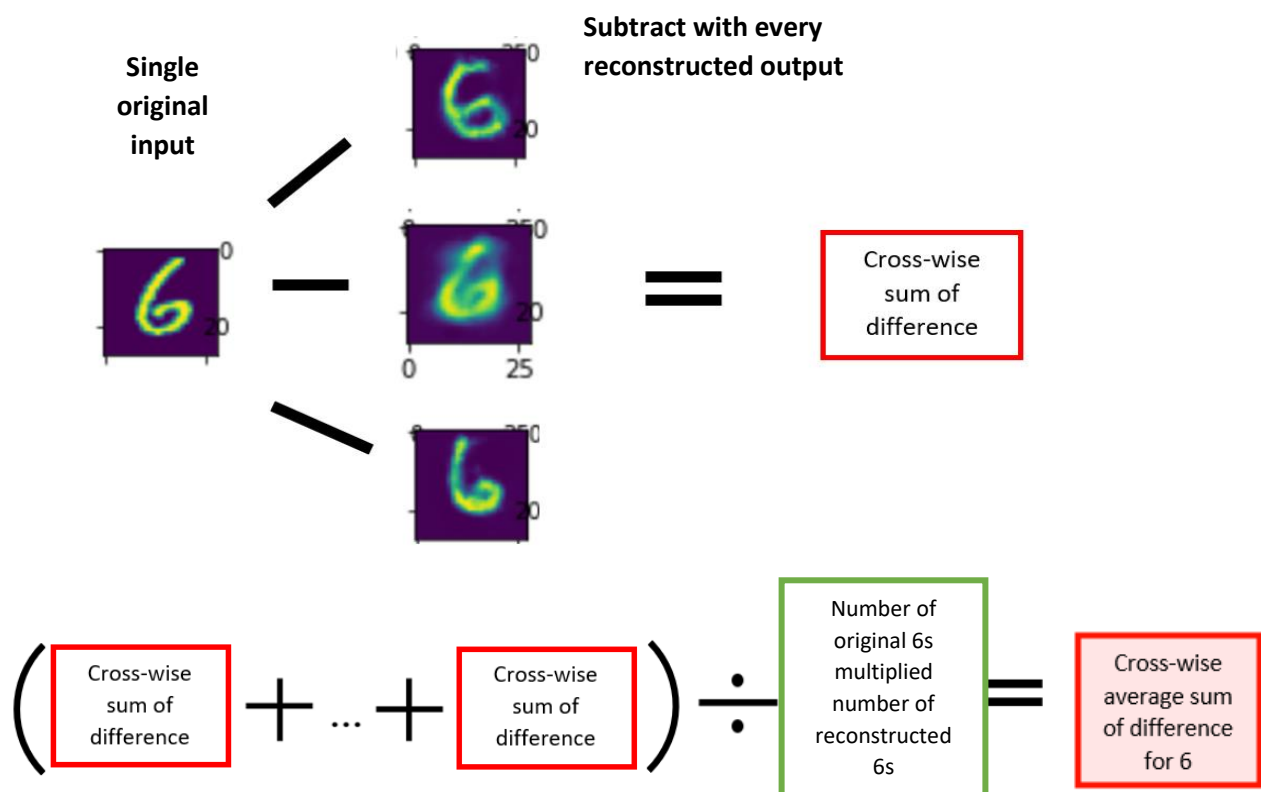
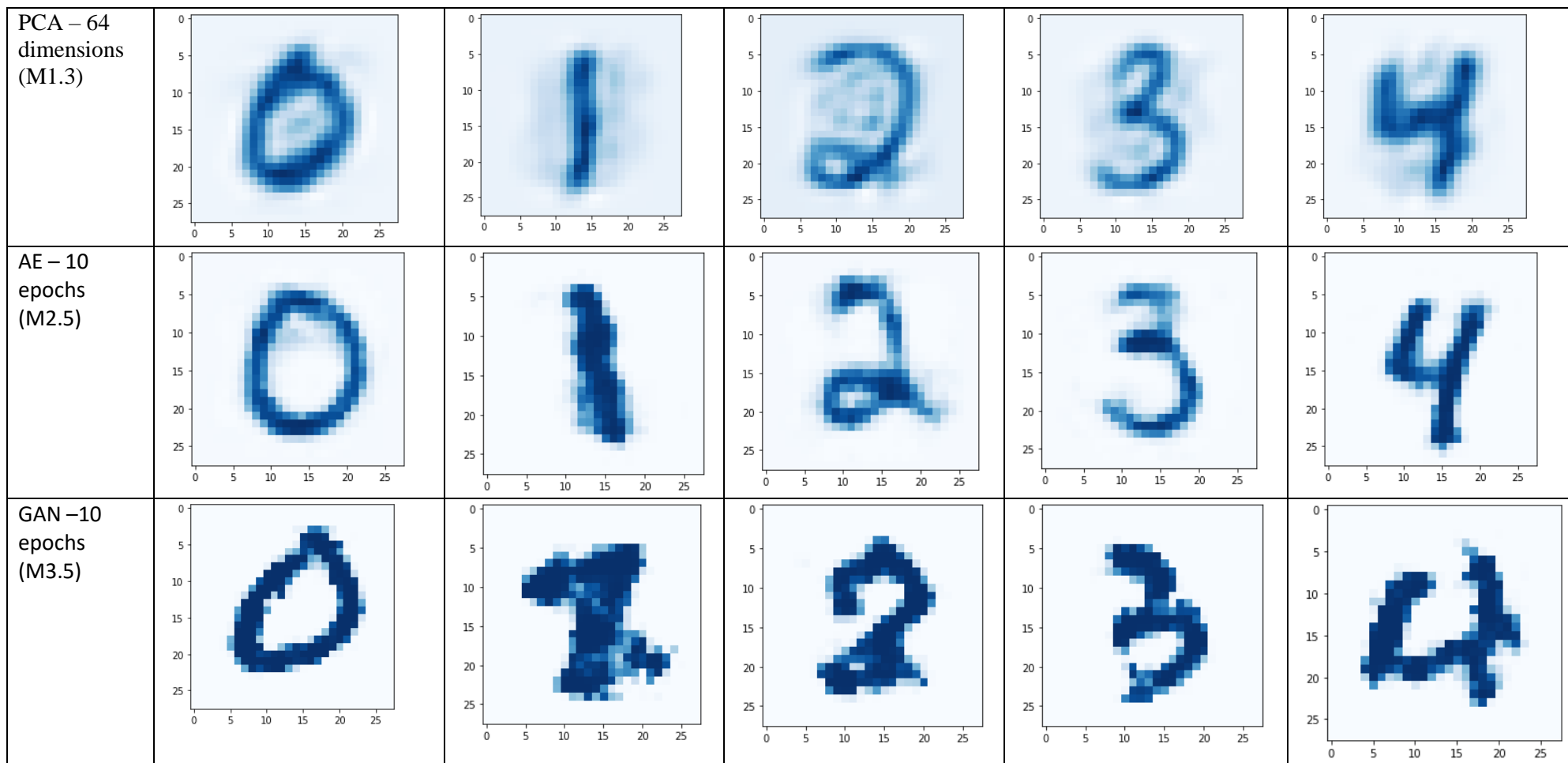


Figure 3: Visual representation of cross-wise comparisons

Finally, the overall average sum of differences *across the 10 digits* were calculated for both types of comparison.

Performance on Vanilla Neural Network (S2): A vanilla ANN (Rashid) was trained with normal MNIST dataset and tested with the outputs of the three codes (M1.3, M2.5, M3.5), to determine whether the reconstructed outputs could be recognised by the vanilla ANN. Confusion matrices could be generated by PCA and AE as they had correct label targets, a bar chart was generated for GAN for the frequency of network predicted values.

Results



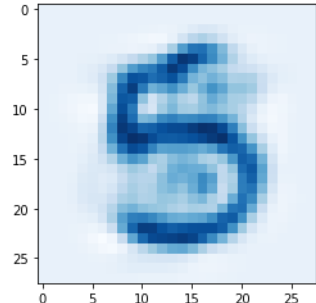
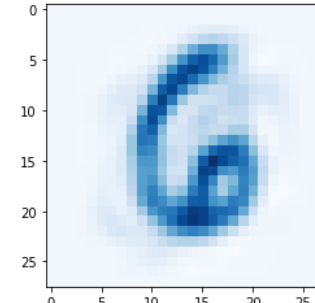
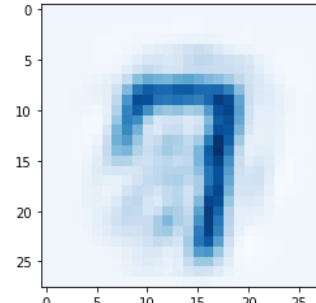
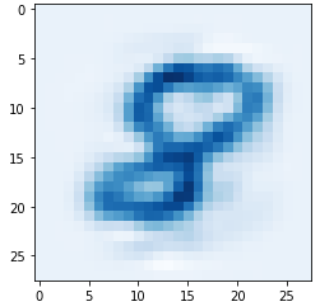
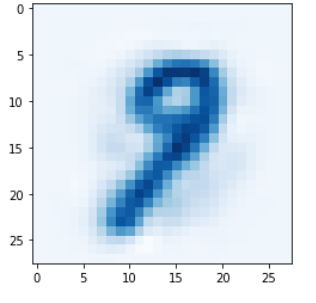
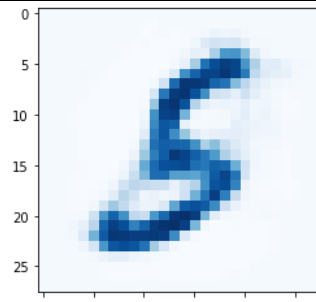
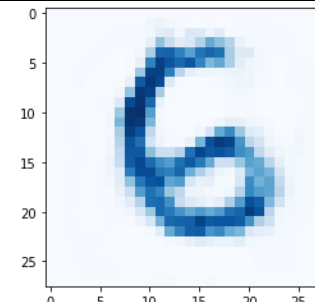
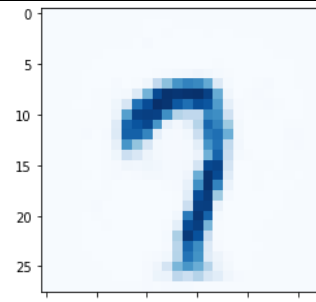
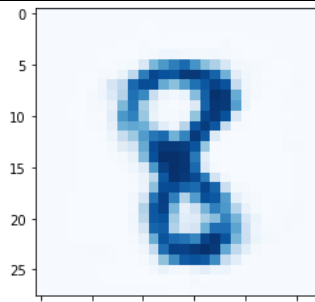
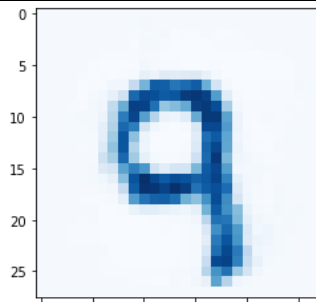
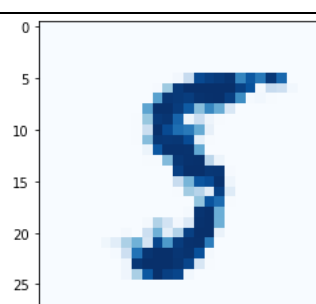
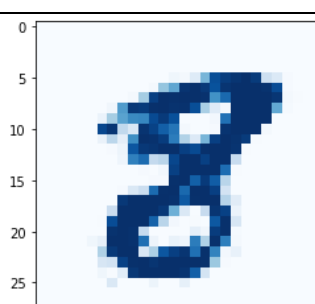
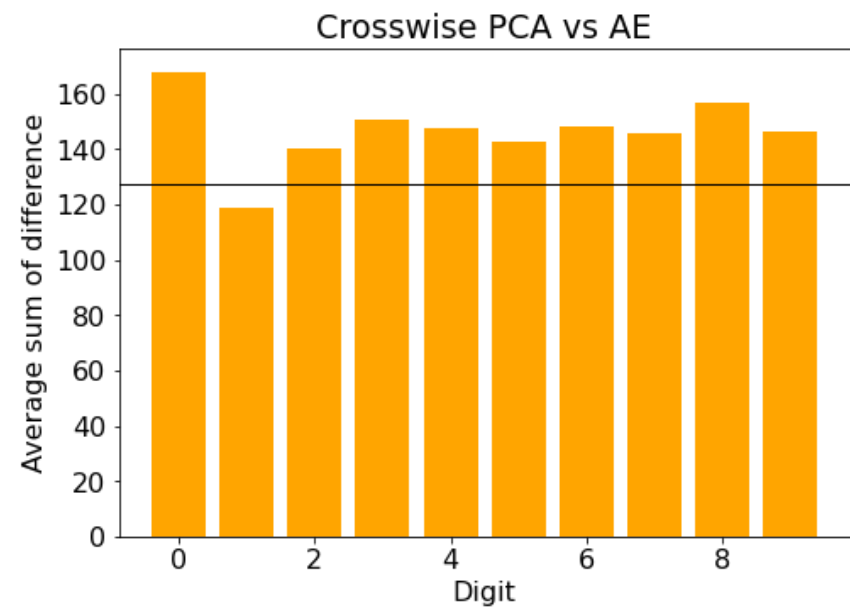
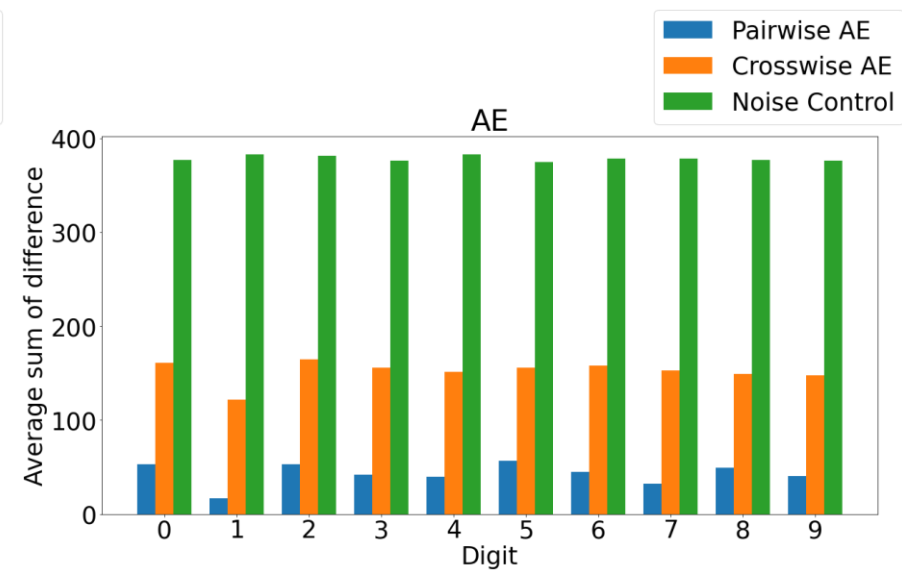
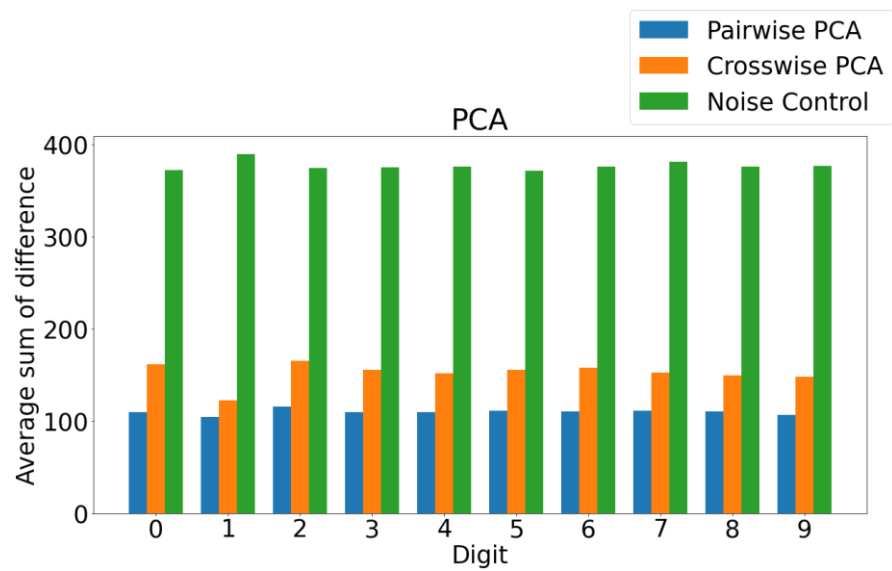
PCA – 64 Dimensions (M1.3)					
AE – 10 epochs (M2.5)					
GAN – 10 epochs (M3.5)					

Table 1: Samples of reconstructed outputs at the end of training by main programs. Full outputs found in the Appendix. GAN did not produce a full range of 10 digits.



Figures 4a, 4b, 4c: Average sum of pixel value difference by digit. Individual graphs are included in the Appendix.

	Overall (% difference)	0	1	2	3	4	5	6	7	8	9
PCA Pairwise (S1.3a)	109.53 (13.97%)	109.27 (13.94%)	104.02 (13.27%)	115.91 (14.78%)	109.68 (13.99%)	109.99 (14.03%)	111.00 (14.16%)	110.72 (14.12%)	111.41 (14.21%)	110.59 (14.11%)	106.65 (13.6%)
PCA Crosswise (S3.1b)	151.92 (19.38%)	161.23 (20.57%)	122.09 (15.57%)	164.93 (21.04%)	155.52 (19.84%)	151.71 (19.35%)	155.63 (19.85%)	158.09 (20.16%)	152.76 (19.48%)	149.40 (19.06%)	147.85 (18.86%)
AE Pairwise (S1.3c)	42.03 (5.36%)	52.76 (6.73%)	17.09 (2.18%)	53.08 (6.77%)	41.65 (5.31%)	39.76 (5.07%)	56.87 (7.25%)	45.16 (5.76%)	32.23 (4.11%)	49.30 (6.29%)	40.43 (5.16%)
AE Crosswise (S1.3d)	94.33 (12.03%)	114.06 (14.55%)	47.59 (6.07%)	116.21 (14.82%)	91.08 (11.62%)	83.30 (10.63%)	118.29 (15.09%)	105.43 (13.45%)	81.39 (10.38%)	90.52 (11.55%)	95.46 (12.18%)
PCA and AE Crosswise (S1.3e)	144.32 (18.41%)	168.10 (21.44%)	118.80 (15.15%)	139.98 (17.85%)	150.94 (19.25%)	147.34 (18.79%)	142.73 (18.2%)	147.95 (18.87%)	145.52 (18.56%)	156.60 (19.98%)	146.22 (18.65%)

Table 2: Average sum of pixel value difference per digit. Percentages were calculated with a maximum sum of difference of 784

Discussion

The Reconstructions

The three algorithms produced interestingly different results. All outputs are 28x28 pixels, yet qualitatively they appear of different clarities. GAN produces a higher contrasting image, while AE and PCA create more consistently recognisable digits.

The blurriness of PCA and AE is likely an artefact of dimensionality reduction, contrast is not as important as the general shape of the numbers. Conversely, GAN generates outputs entirely from random noise inputs, compared to PCA and AE whose inputs are the original real digit. This may suggest that GAN has learnt that the crisp outline is more likely to fool the discriminator, and therefore has retained this feature.

For lower PCA dimensions and early in AE training, there is a common shape. In both, the top left curve and bottom right curve appear to be darker, and it is common through most of the examples at this point in training (see Appendix M1.4, M2.3). GAN in the first epoch has also started to outline several curves, and the top-left, bottom-left pattern is feasible, but not consistently present.

These blobs are likely closely related to the latent space, that is, the abstract space that retains the most important features of the data. The latent space holds information about the entire dataset, and thus at lower dimensions and early training, networks lose nuances of individual numbers in order to faithfully retain the general data for all numbers, which is why it is often present.

By the end of GAN training, not all outputs had number-like appearance, and those that did were often '3', '8', '0' (see Appendix M3.5). Some numbers were entirely absent, such as '6', '7', '9', even the single '1' identified is scarcely recognisable. This suggests that GAN does not learn all the numbers in the dataset, but rather hones in on the numbers that may have consistently fooled the discriminator. It may become averse to the numbers that were easily spotted as fake, leaving them absent from the latent space by the end of training.

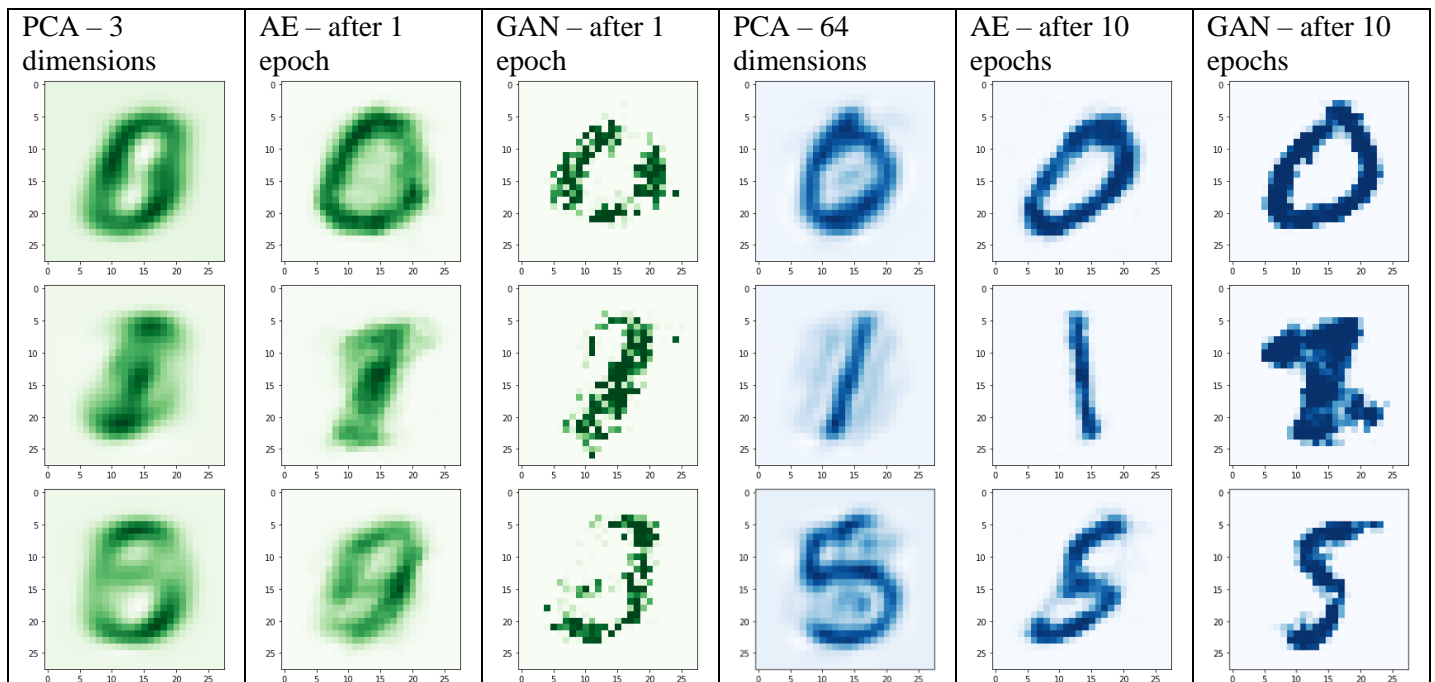


Table 3: Select samples of '0', '1' and '5' at early versus late learning

Quantitative Comparison

It was hoped the reconstructions could be run through the vanilla network to see their performance in replicating a number. It would have been interesting to test outputs at various points in training for AE and GAN or various dimensions in PCA to see the smallest dimension reduction at which the NN could still recognise them. However, the VNN failed to recognise reconstructions as any other number except '5' (see Appendix S2). Once tested on random noise which produced the same outcome, it suggested that the results were more of a limitation of the VNN to discern images than the outputs themselves.

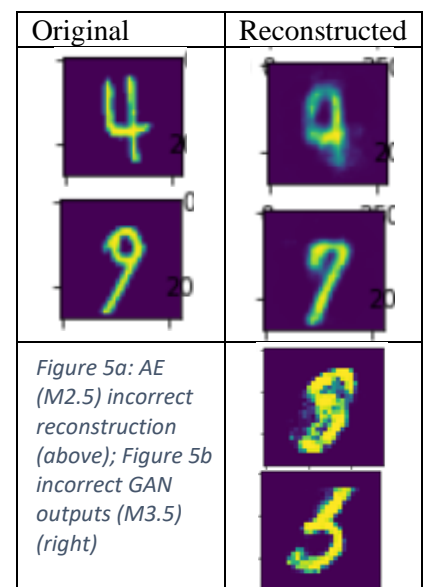
Nevertheless, the quantitative comparison outputs provided several insights. As pairwise comparisons calculated the difference directly between the original and reconstructed of the original following the various dimensionality reductions, this was an indicator of the algorithm's ability to replicate the images. The crosswise comparison took the differences of a single original and compared them to all reconstructed images of the same digit, an indication of how well the algorithm's general idea of a specific digit can replicate it.

AE's results had the lowest overall difference, both in pairwise and crosswise comparisons, suggesting it was more adept at both learning and reconstructing. Furthermore, while differences increased from pairwise to crosswise AE comparisons, it was a relatively consistent scale up across all digits, further suggesting that its more accurate representation of digits in the latent space contributed to its ability to replicate them. In comparison, pairwise PCA comparison showed the least variation between all the numbers, which is not surprising given that its dimensionality reduction is purely mathematical; the samples undergo the same linear transformations. Yet despite this, crosswise PCA comparison began to show preferences for numbers in a similar trend to AE, for example less difference for '1', and greater differences for '0' and '2'. This may hint at certain properties about these numbers causing similar preferences by both machines for the numbers. However, GAN failed to produce '1's, and rather pursued numbers with many curves such as '8' and '3'. It could potentially be that generation of these numbers from noise was easier than generation of a straight line, and so the latent space favoured them.

The Blob & Latent Space

Towards the end of training, especially in AE, the images became clearer, suggesting the networks have more well-defined representations of individual numbers. At this point, networks still reconstruct via the same method – reduce dimensions, then increase – but their weightings are more finely attuned. Both crosswise comparisons suggest that '1's and '7's are the most well-recognised, which is interesting considering they are straight lines. Perhaps they are the most distinct within the round latent space blob, meaning they are easier to discern from others. Conversely, samples in figure 5a demonstrate similar looking numbers may be closer together in the abstract latent space, therefore reconstructed by AE incorrectly, or even form unrecognisable numbers as with GAN (figure 5b).

The blob being present in all of the earlier training examples suggests that the machines look for the *most common* curves drawn across all examples. It also suggests that features of a specific number are *relative* to the entire dataset, that all information was being learned and processed in parallel. A machine may first learn by creating the latent space capturing all of the data (represented by the blob) and moving towards different points in the latent space to achieve variation within the set.



This may potentially link to humans' innate ability to categorise; the concept of a number may be our 'latent space', a foundational set upon which slight tweaks generate different subsets within the overall set, and are inherently related to one another. After learning, recall of the number may be accessed first through the set of all numbers, then gaining the nuances later. It is rare that we learn about the number '1' without learning about the number '2' and that they serve the same purpose. Children may forget the symbol or its phonetic name but if they have learnt the existence of ordinality, it is possible they search within their numerical 'lexicon' to retrieve it.

In humans, does the recognition path traverse a latent space or is it bypassed altogether? Neuroscience studies have observed selective regions in the ventral visual stream activate for letter and word recognition (visual word form area) and a separate region activates for numerical form recognition (numerical form area), in the inferior temporal gyrus (Yeo et al., 2017). However, most of these studies have not explored the differential activations for different symbolic numbers, the region itself too small to provide fine imaging signals. A review by Merkley et al. (2014) outlined results showing pre-literate children show activations in the VFWA in response to both letters and numbers, and suggested the separation of regions for symbols is experience-dependent. Might this be analogous to an early 'latent space' for all symbols, and following learning it becomes two separate latent spaces? Or two groups within one latent space that are mapped far apart? It could also be plausible that the subspaces within the NFA latent space separate even further for selective activation of specific numbers. How well these ideas of latent spaces and machine learning can explain human learning remains a large field of potential study.

One marked difference between human and machine learning is the integration of novelty into latent spaces. Pimentel et al. (2014) describe the growth versus stop problem in generative networks with regards to deciding whether to integrate new units into the model structure or to stop. The GAN in this paper training on a simple dataset already demonstrates this effect. While more complex generative models would be stronger at differentiating between different units, in higher dimensional spaces, the nuances between a new unit and a varied existing unit would plausibly be even more difficult to train, especially for concepts that are not discrete like numbers here. Higher cognitive functions may even use categorisations recursively, which may be difficult to model.

Additionally, these three techniques, although different, are similar considering the range of machine learning programs invented. Not to mention, they are simple in comparison and trained with a simple database. Therefore, the idea that the fundamental features of digits is an amalgamated representation of all digits may have only occurred as characteristic of these simple generative techniques.

Moreover, GANs and AEs utilise set parameters that shape the process of computation and learning. Although intuitively, standardisation of these parameters across codes may produce a more 'controlled' setting, parameters that worked for one code did not produce feasible outcomes for another. The nature of each neural network may be such that a combination of parameters more optimised for one may produce poor learning in the other. Pimentel et al. (2014) suggest that generative methods "require the optimisation of a pre-defined number of parameters that define the structure of the model, and may also be very sensitive to these model parameters".

This raises an interesting question about the 'optimisation' of parameters to produce better outcomes. Will complex models eventually reach a point of connectivity where parameters do not affect it or are self-optimised, or will brain models demand even more precise mathematical modelling and parameters before brain-like processing appears?

Will computation alone be sufficient for brain models?

Conclusion and Future Implications

The ‘blob’ in reconstructed outputs suggest that the fundamental features of an individual number is the basis of all numbers as a whole. Nuances upon the basic foundation create the variations within the set of numbers. As learning continues, networks become better at recreating them, suggesting the more refined a latent space becomes the better the outputs. Numbers that are more distinct to others are potentially easier to learn. The fact that three different techniques produced similarities may infer that this amalgamated representation occurs for learning of any dataset however, this may be an attribute of the style of these relatively similar techniques. It would be interesting adapt programs to reproduce outputs without original input, ie. by telling it to draw a ‘5’, to test how well its representation replicates the digit. Ultimately, while the representation of all numbers seems to underlie the machine’s learning of individual numbers, this is still a long way to understanding how a human may acquire the set of all numbers.

References

- Hui, J. (2018) GAN – Ways to improve GAN performance. Retrieved from <https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b>
- Hutmacher, F. (2019). Why is there so much more research on vision than on any other sensory modality? *Frontiers in Psychology*, 10, Article 2246. <https://doi.org/10.3389/fpsyg.2019.02246>
- Merkley, R., Wilkey, E. D., & Matejko, A. A. (2016). Exploring the origins and development of the visual number form area: a functionally specialized and domain-specific region for the processing of number symbols?. *Journal of Neuroscience*, 36(17), 4659-4661.
- Pimentel, M. A., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, 99, 215-249.
- Rashid, T. (2020) Python Notebooks Accompanying the Book *You’re your Own GAN*. Retrieved from <https://github.com/makeyourownneuralnetwork/gan>
- Santens, S., Roggeman, C., Fias, W., & Verguts, T. (2010). Number processing pathways in human parietal cortex. *Cerebral cortex (New York, N.Y. : 1991)*, 20(1), 77–88. <https://doi.org/10.1093/cercor/bhp080>
- University of Toronto Computer Science <https://www.cs.toronto.edu/~lczhang/360/lec/w05/autoencoder.html> via Python Engineer (2021). Retrieved from <https://www.python-engineer.com/posts/pytorch-autoencoder/>
- Yeo, D. J., Wilkey, E. D., & Price, G. R. (2017). The search for the number form area: A functional neuroimaging meta-analysis. *Neuroscience & Biobehavioral Reviews*, 78, 145-160.

Appendix

Main Programs Outputs

Figure M1.3 PCA data – outputs applied to test data set with 64 components

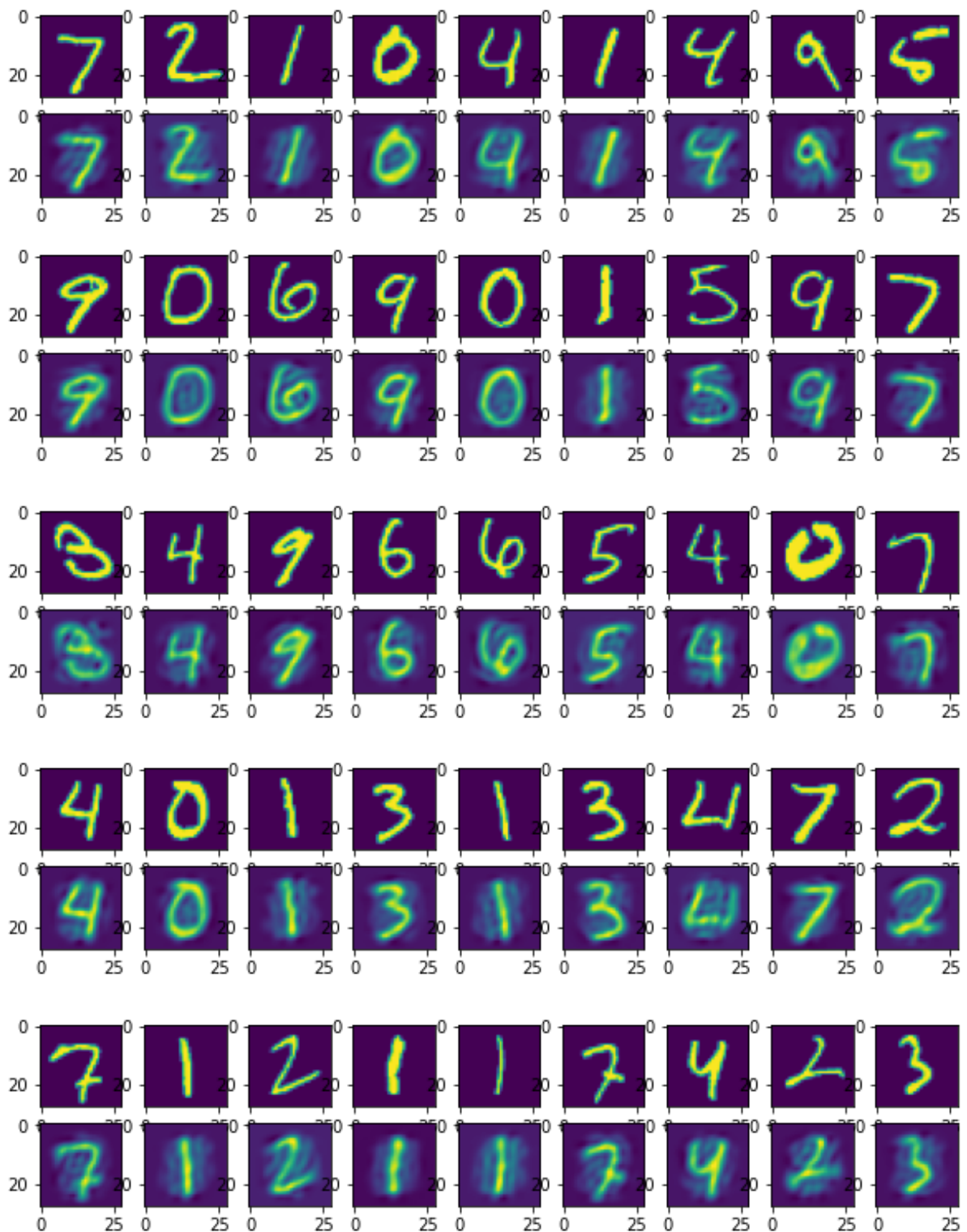
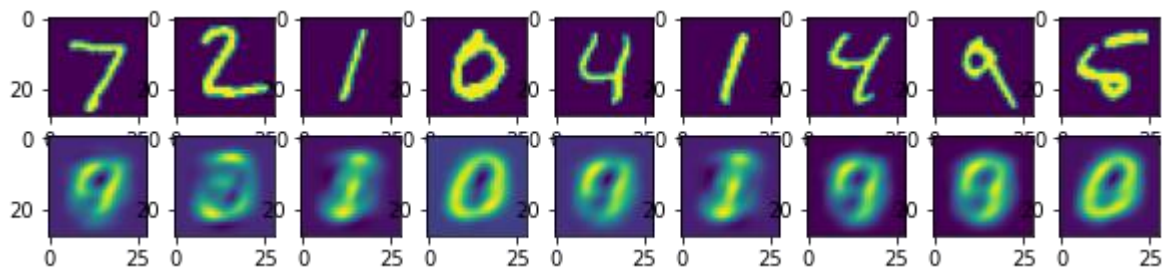
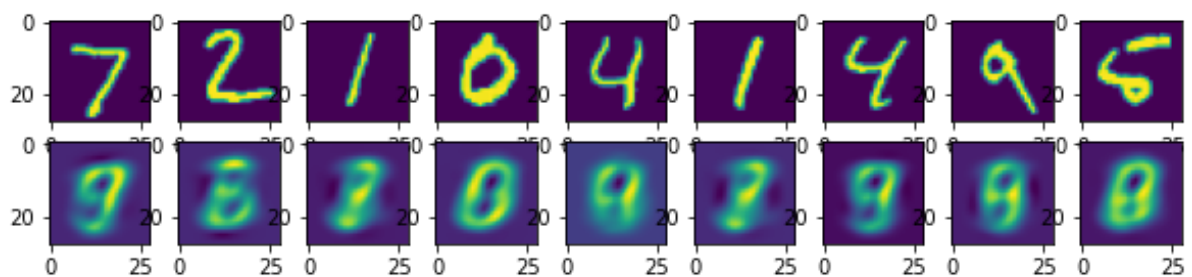


Figure M1.4 PCA dimensions data – outputs captured with different number of principal components

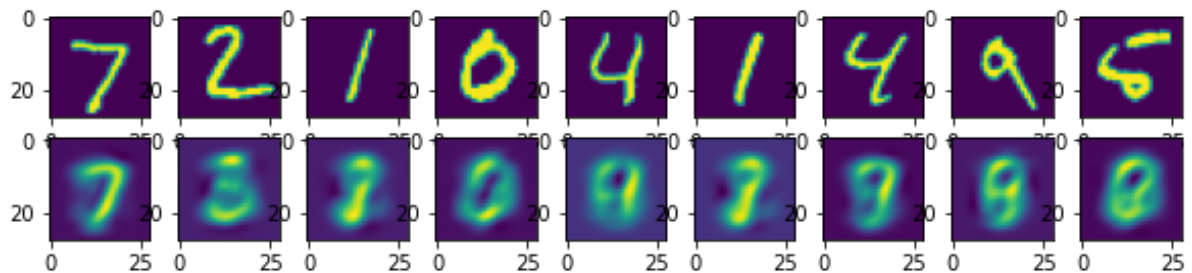
3 Principal Components



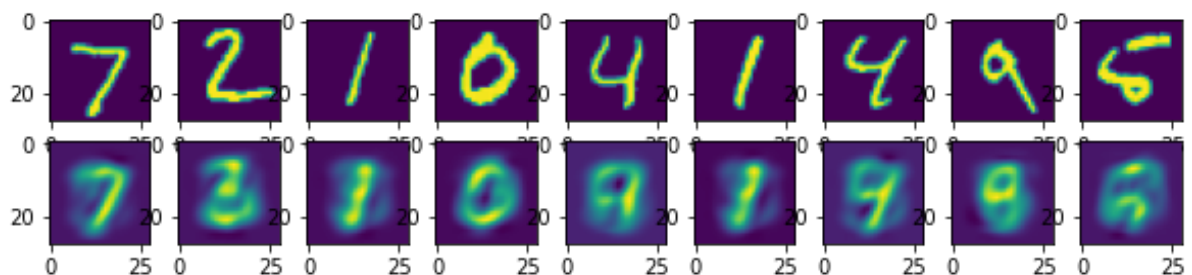
6 Principal Components



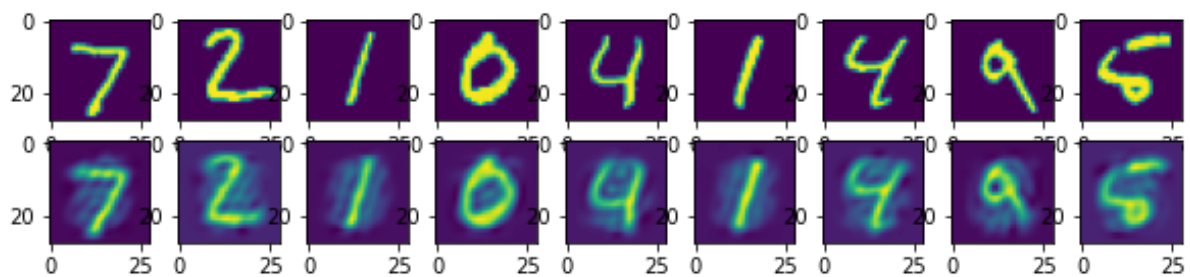
8 Principal Components



12 Principal Components



64 Principal Components



128 Principal Components

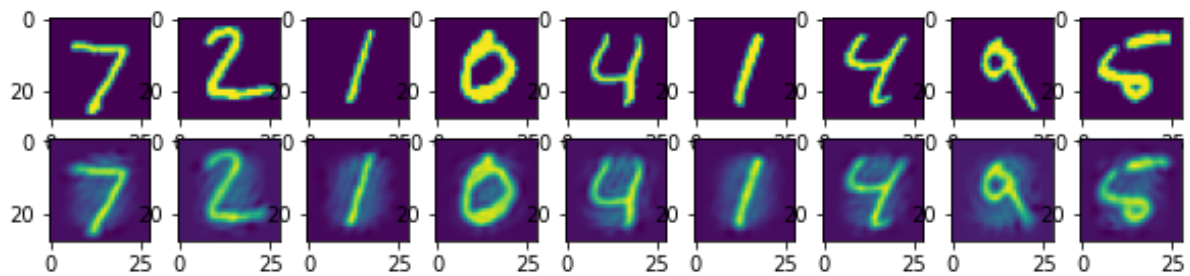
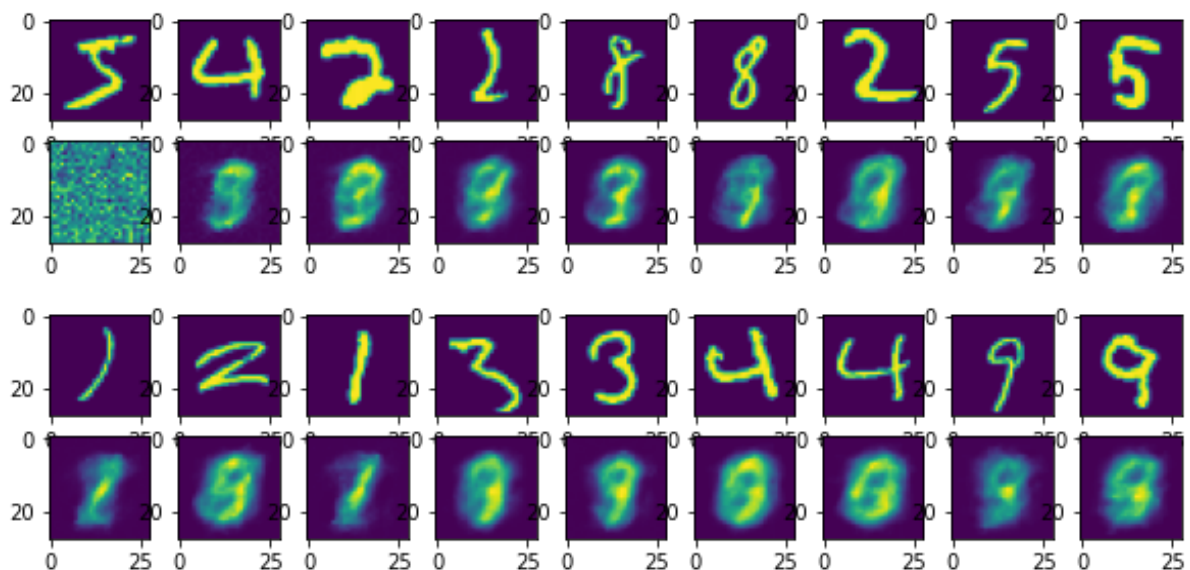
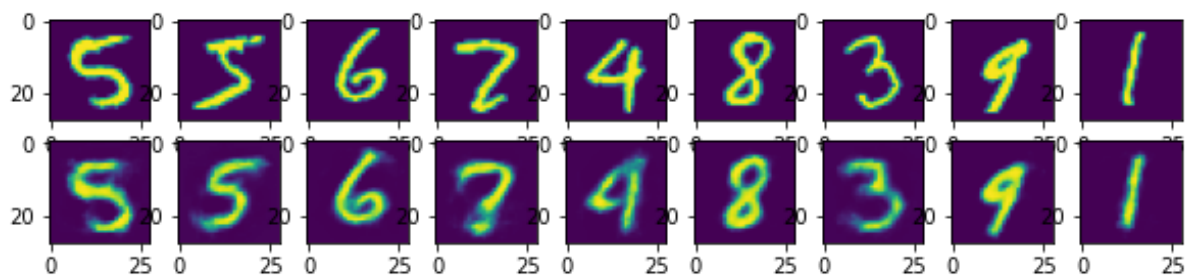


Figure M2.3 AE output captured at various points during training

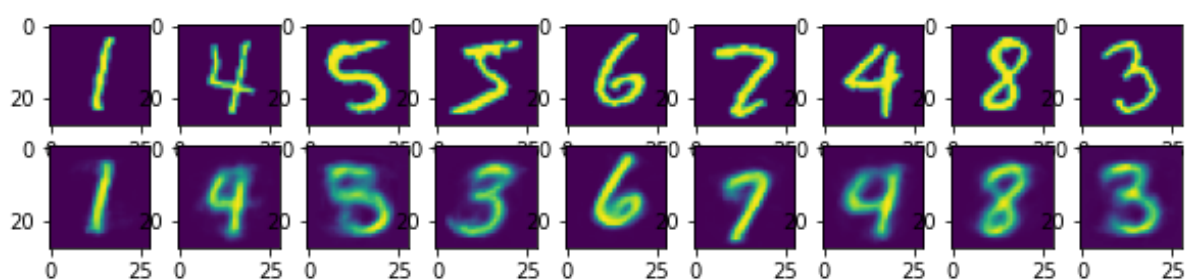
Epoch 1



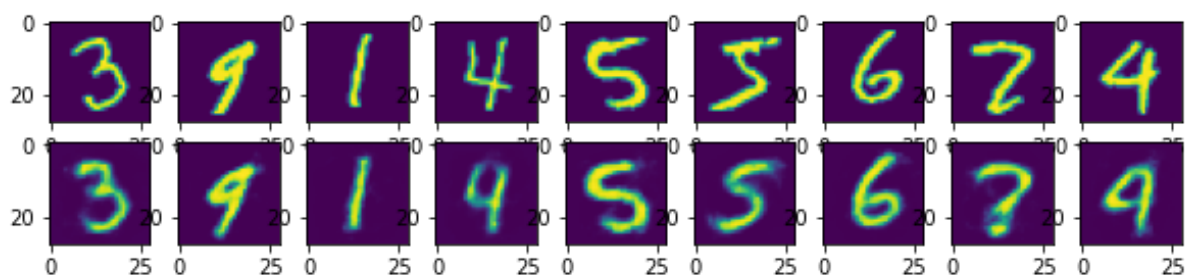
Epoch 2



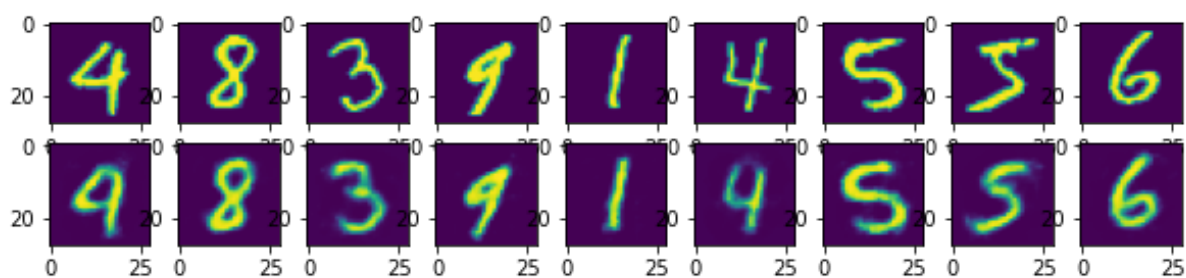
Epoch 4



Epoch 6



Epoch 8



Epoch 10

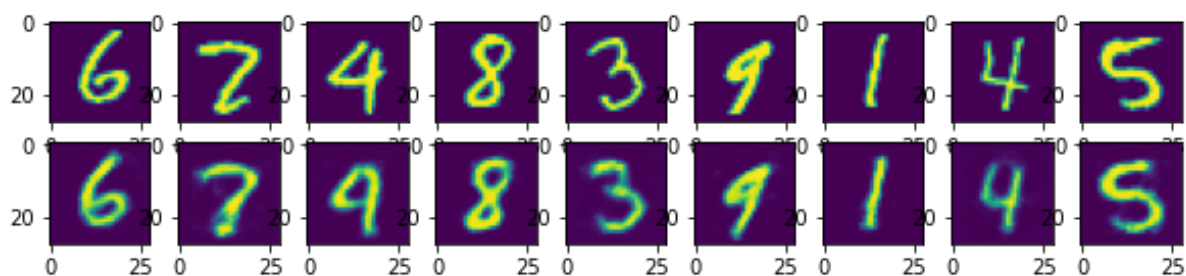


Figure M2.5 AE output captured at the end of 10 epochs

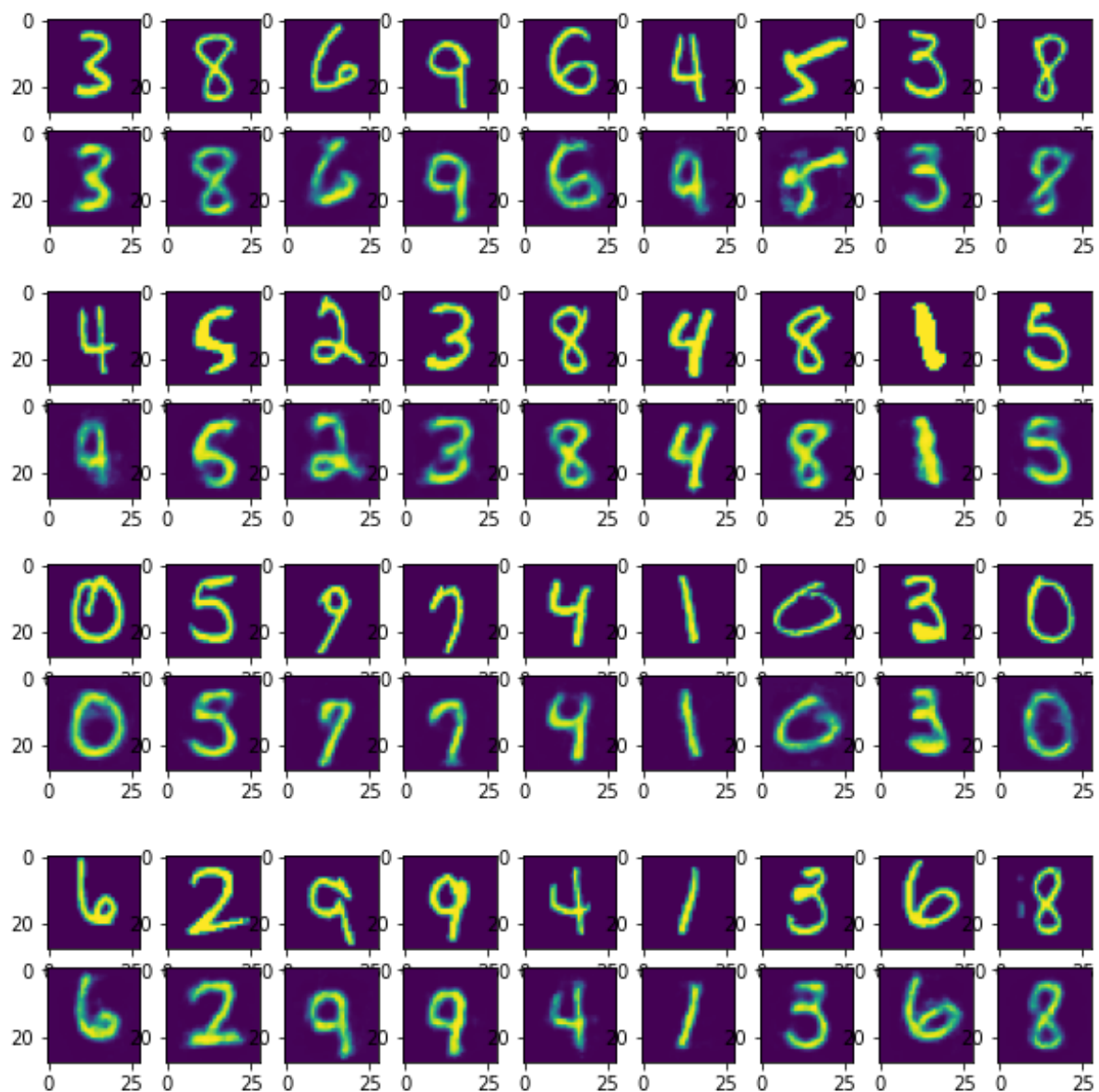
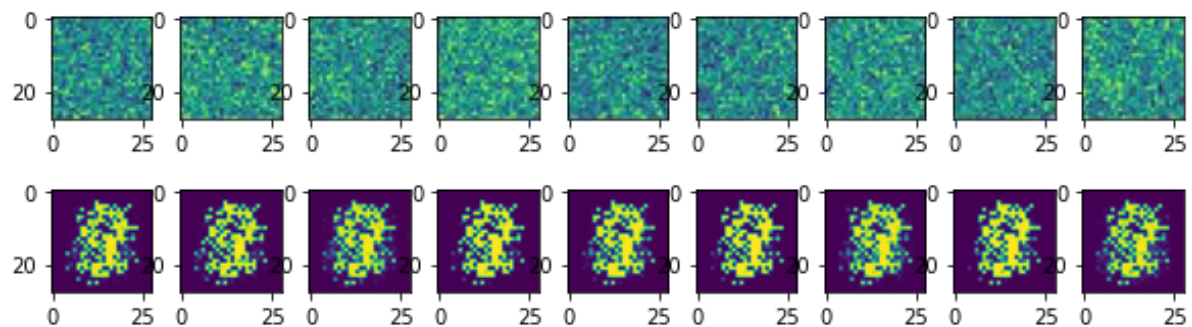
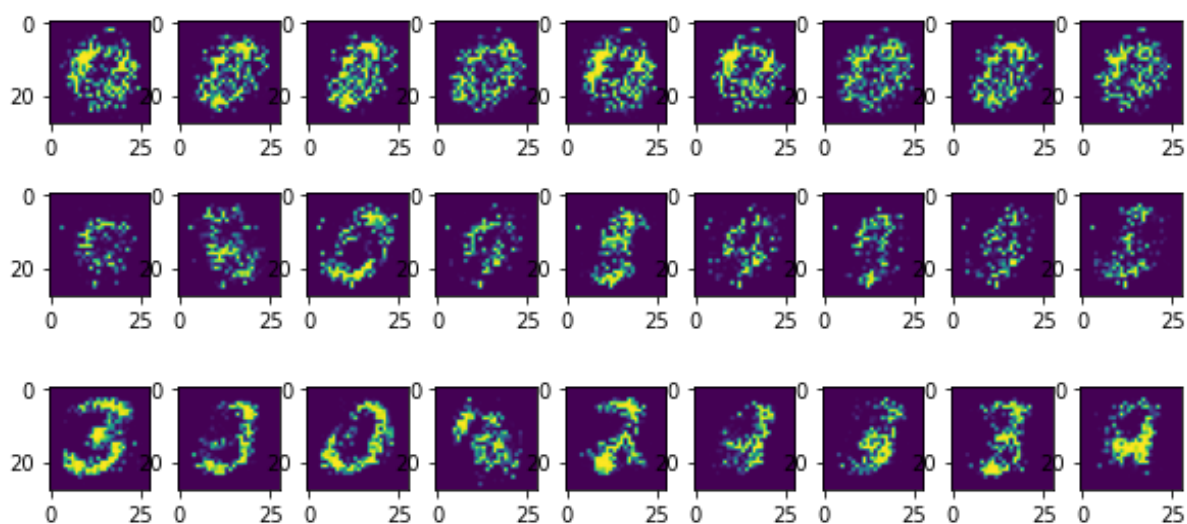


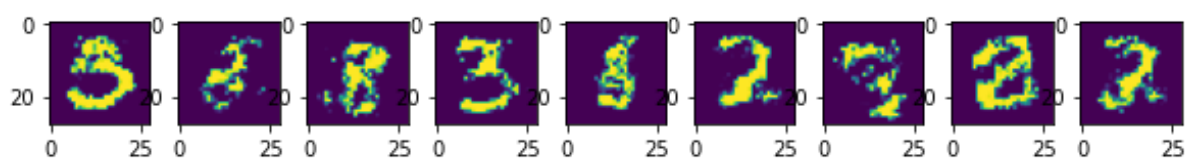
Figure M3.3 GAN outputs captured at various points during training

Epoch 1

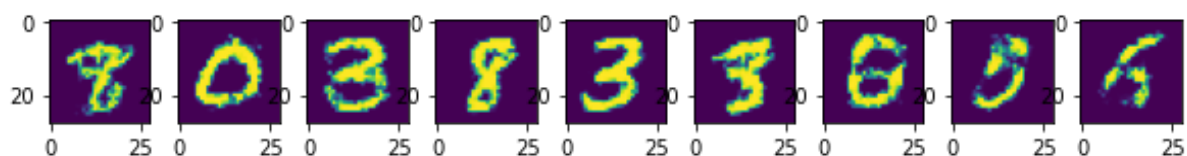




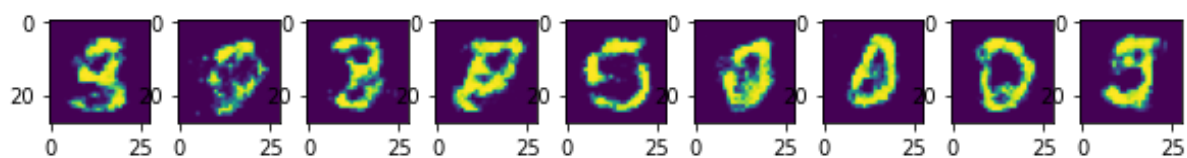
Epoch 2



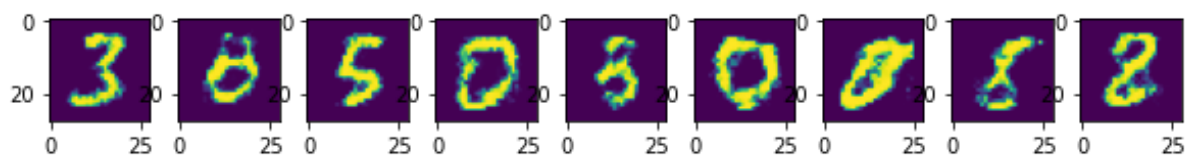
Epoch 4



Epoch 6



Epoch 8



Epoch 10

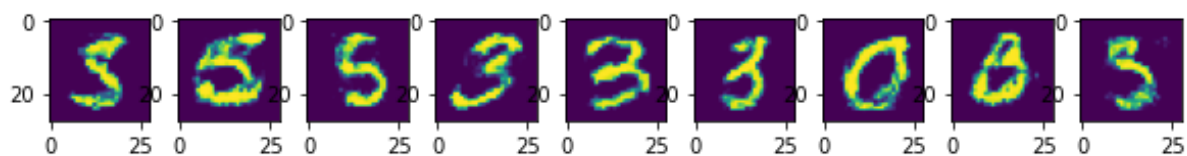


Figure M3.5 GAN output captured at the end of 10 epochs



Secondary Programs Outputs

Figure S1.3a Pairwise PCA average sum of pixel difference by digit

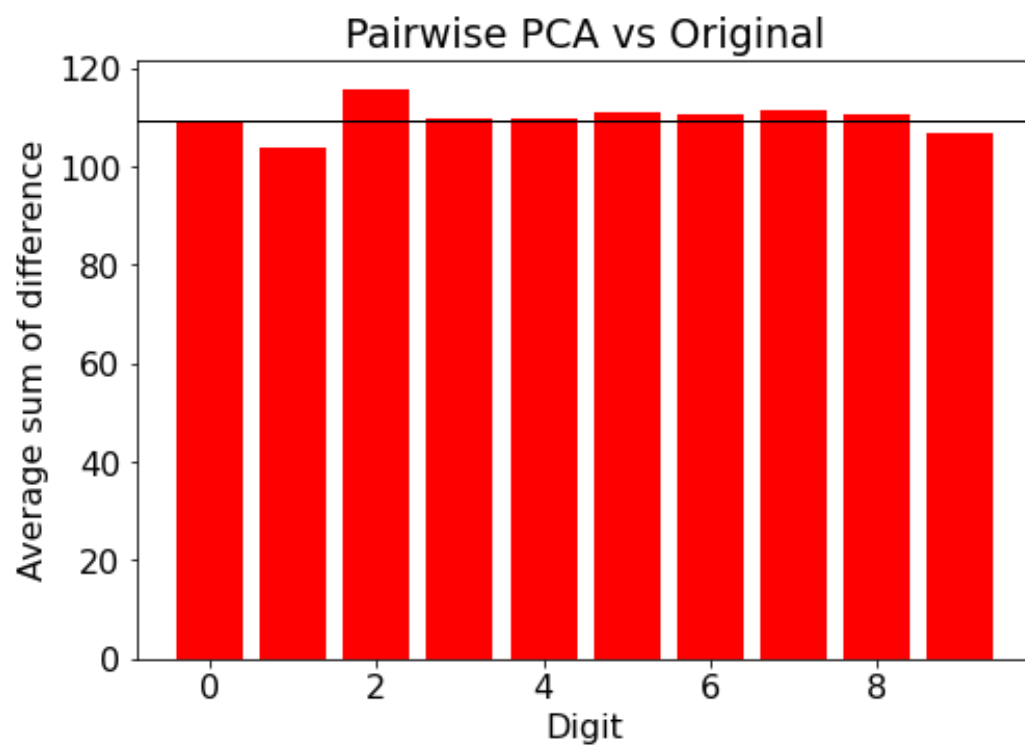


Figure S1.3b Crosswise PCA average sum of pixel difference by digit

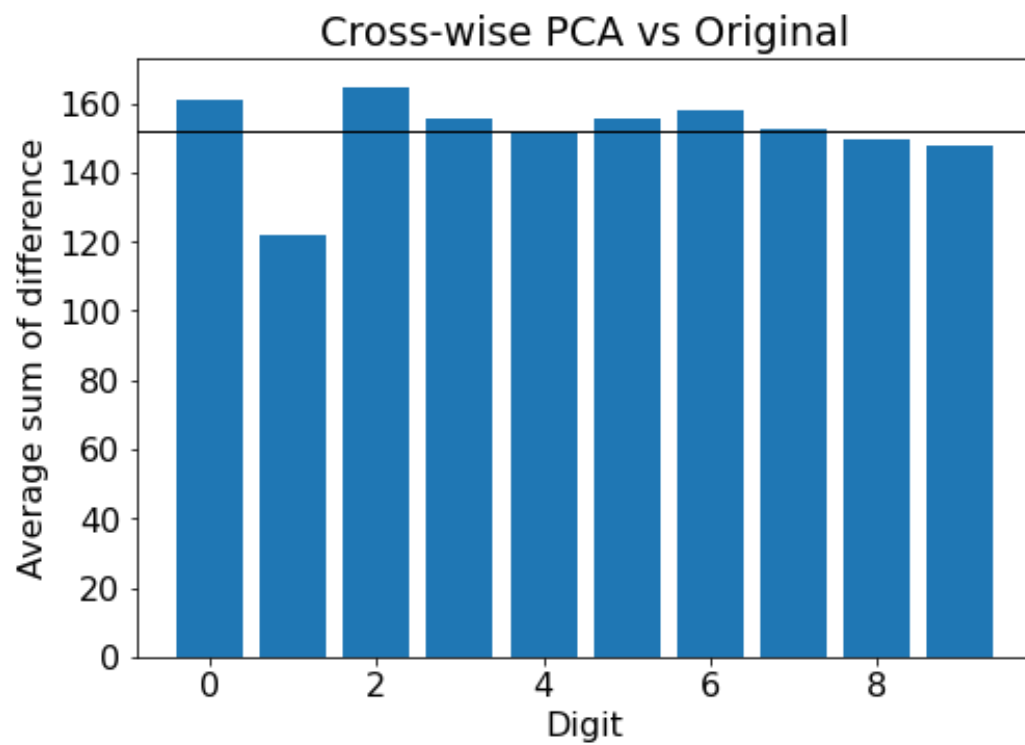


Figure S1.3c Pairwise AE average sum of pixel difference by digit

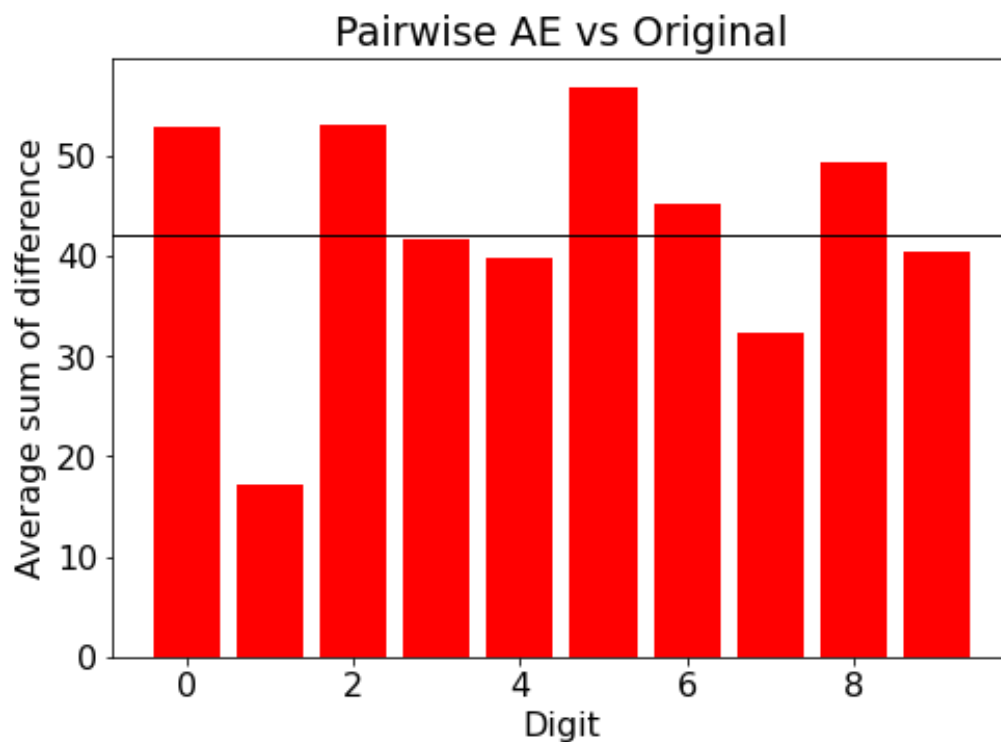
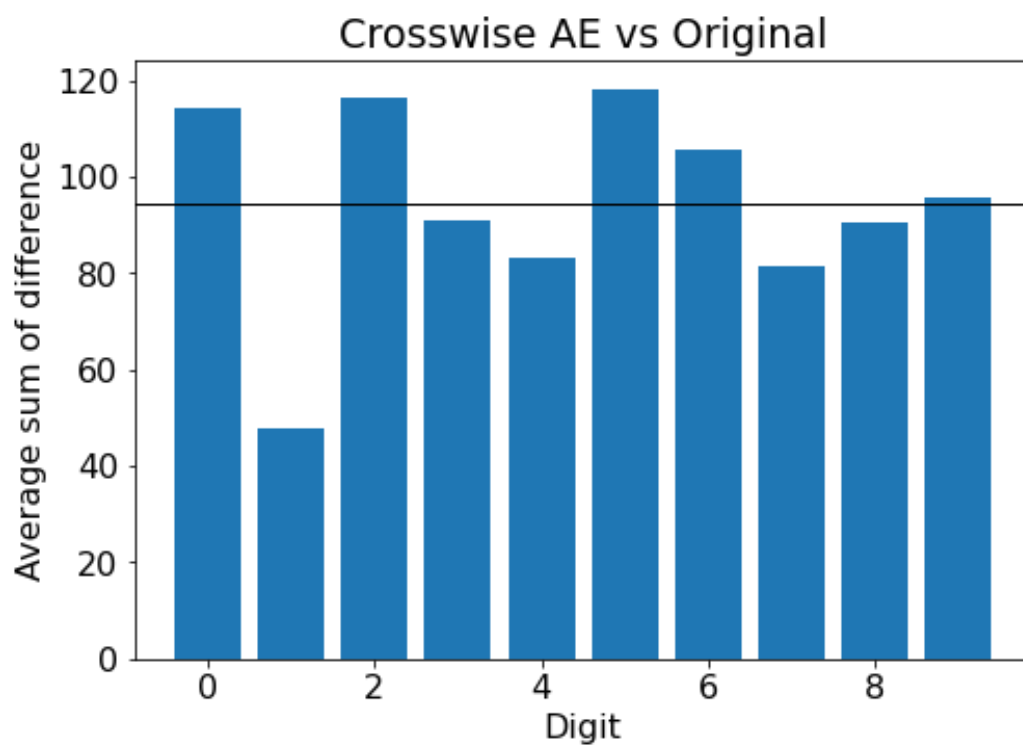
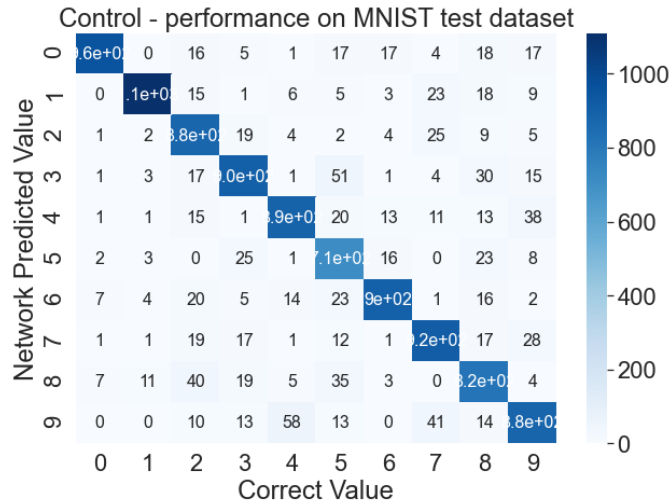


Figure S1.3d Crosswise AE average sum of pixel difference by digit

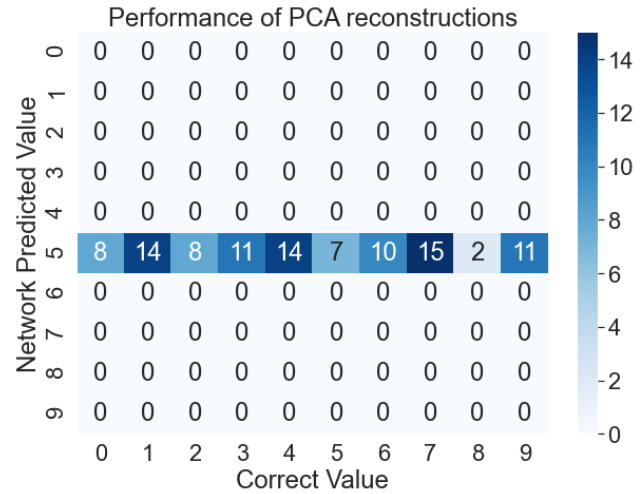


S2 Performance on Vanilla Neural Network

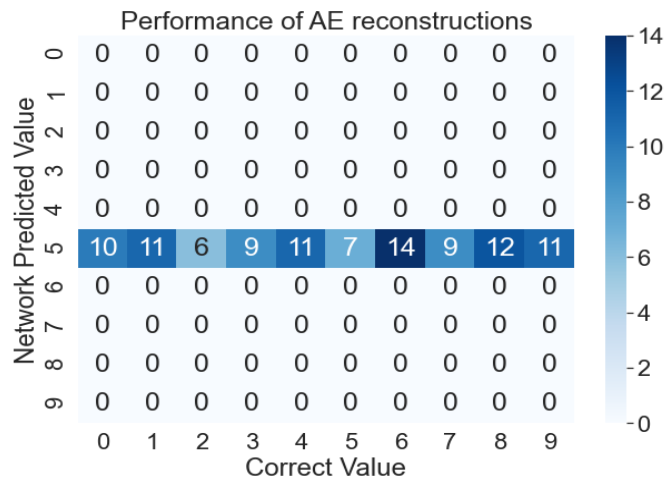
S2.1



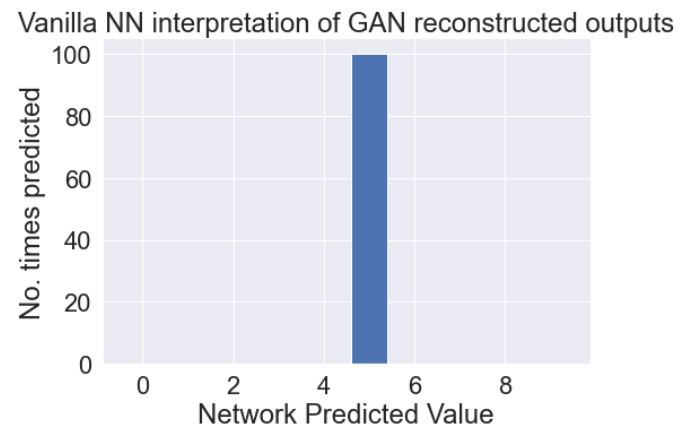
S2.2a



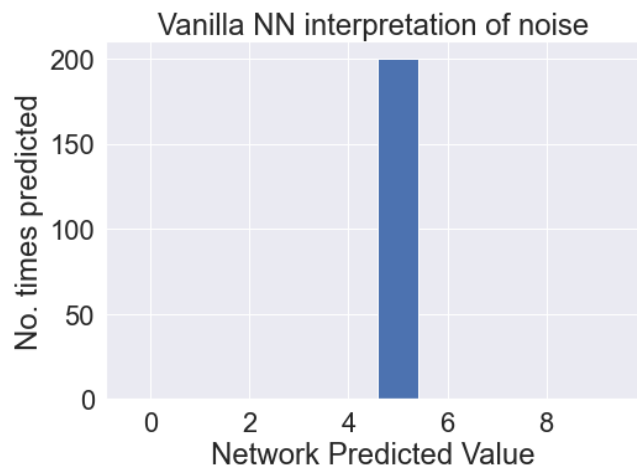
S2.2b



S2.2c



S2.2d



Figures S2.1, 2.2a, 2.2b, 2.2c, 2.2d: Performance of final outputs (M1.4, M2.5, M3.5) compared to two controls; MNIST test dataset and noise.