Analyzing Latent Spaces in Autoencoders of Varied Bottleneck Size

Andre Ye

April 2021

Contents

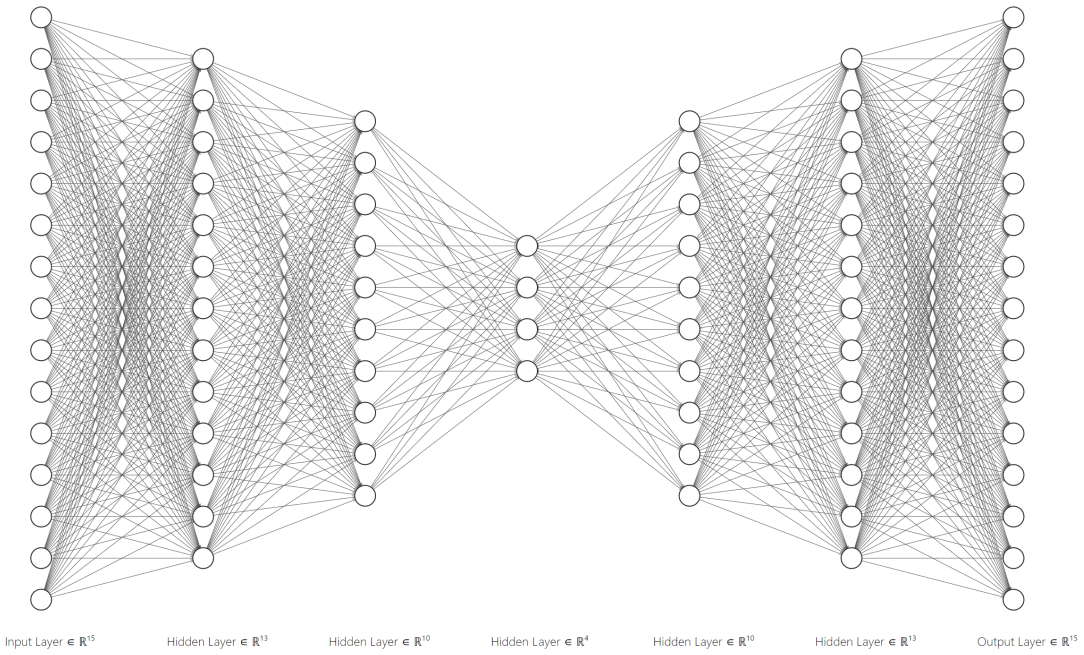

Figure 1.1: A simple autoencoder structure with 15-dimensional input and output and a 4-dimensional bottleneck.

1 Introduction

Neural networks are powerful tools for learning complex forms of data in the modern age. The entire field of deep learning has surged in popularity. However, in the words of a wise person on the Internet, "neural networks are matrix multiplications no one understands." Indeed, much of this is true; despite the large predictive power of neural networks, the actual internal processes of how neural networks learn are difficult to understand. As such, it is important not only to analyze the outputs of the model – which have traditionally been the dominant focus of research – but intermediate layers. One can analyze the neural network's intermediate representations of data to more broadly investigate the learning phenomenon and representation properties of neural networks.

Autoencoders are neural networks structures in which the input is identical to the output. However, the network architecture is built with a "bottleneck" in the middle; that is, in the feed-forward operation, the input is compressed, or encoded, into a smaller representation. This representation is then decoded into the output, which in the case of optimal performance is identical to the input. Because the bottleneck is smaller than the input size, the network must find an efficient representation for all inputs such that it can decode the representation.

- *Encoder*. Represents the first half of the autoencoder that encodes the input in an efficient representation. Restated, it maps the input to a coordinate in the latent space of the bottleneck.
- *Bottleneck*. The bottleneck is n neurons wide; n thus indicates the dimensionality of the latent space.
- *Decoder*. Represents the second half of the autoencoder that decodes the efficient representation into the output (which should be identical to the input).

Because autoencoders are unsupervised in that it can be used on a dataset without labels, they are versatile in being able to find the structure of any dataset. Given the task of exploring the representation capabilities of neural networks, the autoencoder's bottleneck as the intermediate network representation of data is a concrete method of accomplishing the abstract goal of viewing the "thinking processes" of neural networks.

This paper will utilize statistical tools to analyze the latent space of an autoencoder for different dimensionalities of the latent space. That is, as the width of the bottleneck – and hence the "difficulty" of the efficient representation, in that smaller latent spaces are more difficult to store information in – varies, how do encoded representations of the dataset differ? Section 2: Attaining Data will provide a more detailed and specific formulation of this paper's inquiry.

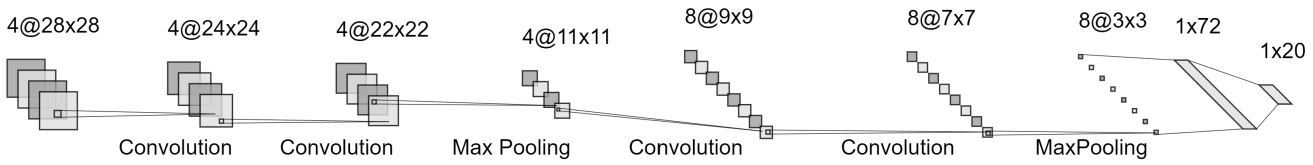

Figure 2.1: The structure of the encoder for the simple autoencoder used for this paper, assuming a bottleneck size of 20 neurons. The decoder is symmetrical to the encoder, but convolutions are replaced with transposed convolutions and max pooling layers are replaced with upsampling layers.

2 Attaining Data

2.1 Process

This paper will use the MNIST dataset, which consists of 60,000 28 by 28 pixel grayscale images of handwritten digits, from 0 to 9. As such, there are 10 classes, each with 6,000 images. Since its initial release in 1999, the MNIST dataset has been a standard benchmark for new models and methods. As such, it serves as a good dataset to analyze the latent space of autoencoders.

To collect the data, the following steps were taken for the bottleneck sizes: 1, 2, 3, 4, 5, 10, 20, 50, 100, and 200 neurons.

1. Create an autoencoder structure with input 28 by 28 by 1 and output 28 by 28 by 1, with a bottleneck of n neurons. Since the bottleneck is one-dimensional whereas convolutions are two-dimensional, the bottleneck will be flanked on either side by a flatten and reshape layer. The encoder architecture is diagrammed in Figure 2.1.

2. Train an autoencoder to reproduce the input image on training data. The labels are not used. Use binary crossentropy loss and train for a maximum of 200 epochs. If the loss does not improve for 3 consecutive epochs, stop training.

3. Detach the encoder from the autoencoder. Get predictions from the encoder on the test data. Since the encoder takes an input and maps it to the latent space, the result is an array of latent-space coordinates for each input in the test data.

4. The encoder is run on the test data and not the training data to evaluate the autoencoder's generalized mapping, rather than the specific learned ones, which may have been overfit to and are hence not representative of the model's intermediate representation. The latent-space coordinates of the test data for each autoencoder are stored.

5. Run a t-Stochastic Neighbors Embedding dimensionality reduction algorithm on the latent-space coordinates and reduce the latent space for each autoencoder into two dimensions.

Although the process to collect the latent space data disregards labels, labels will be included in the analysis of the acquired data; each coordinates in the latent space is associated with a label indicating what digit the original sample input was. As such, we can analyze how clusters of coordinates in the latent space for different classes change.

2.2 Results

The models took 5 hours in total to train on total on GPU. Visual demonstrations of the representation capabilities are displayed in Figures 2.2-2.11. The top row of images represents the input to the autoencoder, and the bottom row represents the autoencoder's output – their attempt at recreating the input from a representation size of n dimensions. The detailed results for training loss (binary crossentropy), training mean average error, and epochs needed for training are displayed in the table on page 5; the loss is visualized in Figures 2.12 and 2.13.

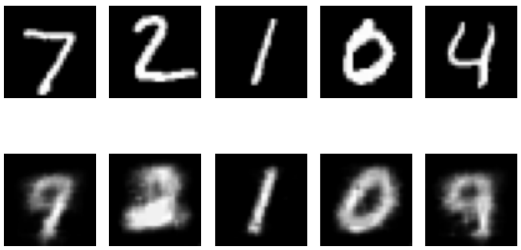

Figure 2.2: Bottleneck size of 1 neuron.

Figure 2.3: Bottleneck size of 2 neurons.

Figure 2.4: Bottleneck size of 3 neurons.

Figure 2.5: Bottleneck size of 4 neurons.

Figure 2.6: Bottleneck size of 5 neurons.

Figure 2.7: Bottleneck size of 10 neurons.

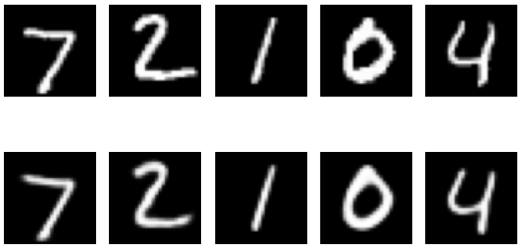

Figure 2.8: Bottleneck size of 20 neurons.

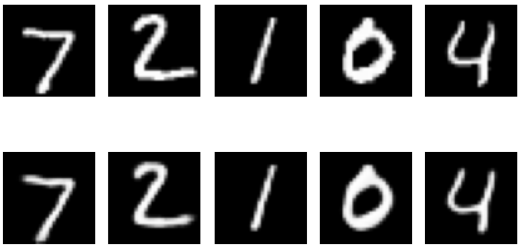

Figure 2.9: Bottleneck size of 50 neurons.

Figure 2.10: Bottleneck size of 100 neurons.

Figure 2.11: Bottleneck size of 200 neurons.

An analysis of the visual representations of autoencoder capabilities is fruitful. The autoencoder with one neuron yields an output resembling the digit "9" when given the digits "7" and "4", which demonstrates an interesting similarity in structure that the model took advantage of.

Yet, the autoencoder is able to reproduce the inputs "1" and "0" from a representation size of one neuron well, capturing the slight slant of the "1". Interestingly, many features of the "0" in the input, like the knob on the top and its relatively straight orientation, have been discarded in favor of a general "0" shape. This suggests that the model has learned to perform some level of clustering by digit without actually receiving labels for that digit.

The autoencoder with two neurons predicts an all black output; analyzing its short training history, it is likely that it was initialized poorly or something went wrong in one of its initial steps.

Bottleneck sizes of three through 200 yield progressively better reconstructions of the inputs. We see two simultaneous trends emerging: confidence and detail. As the bottleneck size increases, the autoencoder becomes more confident in its reconstructions. Autoencoders with bottleneck sizes of three, four, five, and even ten neurons suffer from varying degrees of lack of confidence, representing by gray fuzziness. In autoencoders with larger autoencoder bottleneck sizes, the outlines and filling of the reconstructed shapes are more defined, indicating that the model is more confident in its reconstruction and representation capabilities. Meanwhile, as bottleneck size grows, the reconstruction can increasingly capture more detail. For instance, one can observe the transitioning appearance of a clear knob like the one in the input for the digit zero from 10-neuron to 50-neuron bottlenecks. Moreover, the autoencoder no longer (at least visibly) "mushes together" digits of similar structures, as observed with the digits "7", "4", and "9" in the autoencoder with one neuron. This suggests something intriguing about a change in the representation space of the autoencoder: does a high-dimensional latent space still map inputs belonging to the digit classes "4" and "9", for example, to similarly close regions? How is the autoencoder with a large quantity of bottleneck neurons able to efficiently represent details – does it still learn the ability to perform a clustering by digits, or is this not required given the size of the latent space?

Bottleneck Size	Loss	Training MAE	Epochs Needed
1	0.2139	0.1172	70
2	0.3873	0.2414	6
3	0.1708	0.0875	116
4	0.1564	0.0779	185
5	0.1478	0.0718	159
10	0.1234	0.0545	192
20	0.0875	0.0297	200
50	0.0766	0.0219	172
100	0.0711	0.0173	200
200	0.0680	0.0147	200

2.3 Specific Formulation of Inquiry

After attaining the data, we can now formulate a more specific path of inquiry. In this paper, we will attempt to find similarities and differences across the representation capabilities of autoencoders of bottleneck sizes 1, 2, 3, 4, 5, 10, 20, 50, 100, and 200 neurons in compressing 28 by 28 pixel gray-scale images of handwritten digits. The visual and numerical demonstrations of autoencoder performance suggest two groups of questions:

- How spread out are the points in the latent spaces of autoencoders across bottleneck size? How spread out are the latent representations for each digit class for each autoencoder?
- How far apart are the centroids for clusters of latent representations for each digit class for each autoencoder? As the number of neurons in the bottleneck increases, does the autoencoder get "better" at separating clusters of different classes?

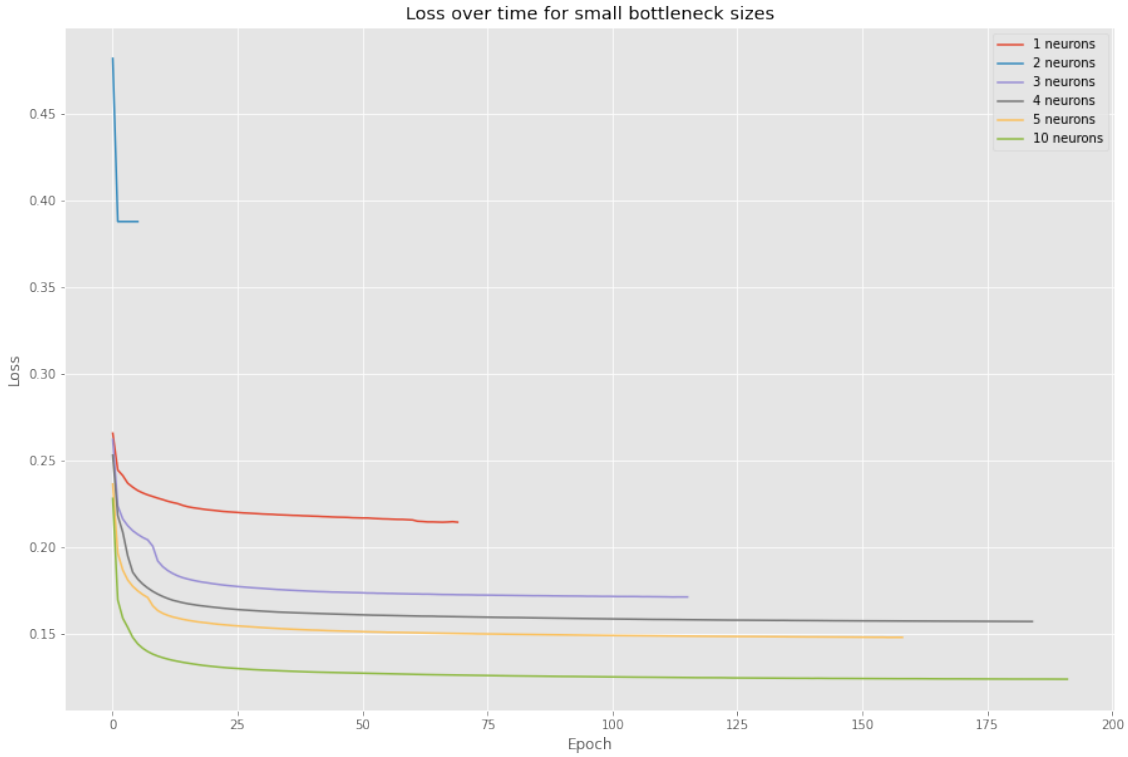

Figure 2.12: Loss over epochs for smaller bottleneck sizes.

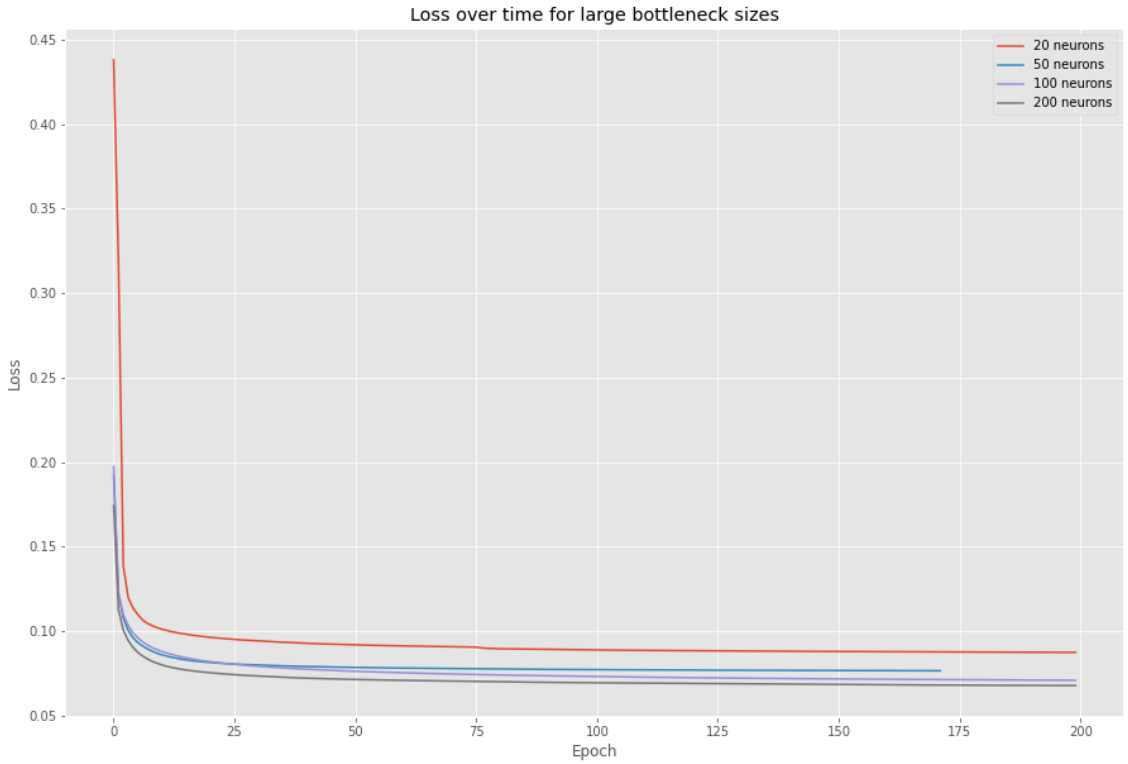

Figure 2.13: Loss over epochs for larger bottleneck sizes.

3 Visual Analysis of the Latent Space

Before engaging in statistical analysis of the latent space, we should begin by performing a visual analysis in two dimensions. Given that the latent spaces are of different dimensionality, it can be difficult to roughly confirm if the statistics we derive seem to be reasonable or not. By visually inspecting the latent space of different autoencoders, we can roughly approximate relationships and expect to find more solidified versions of them in statistical analysis.

Using the t-Stochastic Neighbors Embedding (t-SNE) algorithm, which preserves local structure over global structure to map complex manifolds into lower dimensions effectively, we can generate a two-dimensional representation of the latent space of each autoencoder. t-SNE is unsupervised, meaning it does not attempt to project data into a lower space to maximize distances between classes in a way that Linear Discriminant Analysis does. Therefore in figures 3.1 to 3.6 in which bottleneck sizes of 3 to 50 neurons are displayed, although individual points are highlighted by which class they belong to, the t-SNE algorithm did not consider labels. Hence, any separation observed by class is a representation of the autoencoder, not a feature of the t-SNE visualization process. Note, however, that t-SNE is a stochastic process, meaning that the orientation and specific layout of entities may not be the same; while analysis should not consider the specific orientation of the diagrams, it can consider the shape and relationship between clusters.

Visualizing the latent spaces of the autoencoders yields very rich insights into how different bottleneck sizes force autoencoders to find different representations of data. One of the most visible emerging features as the bottleneck size increases is the separation between classes. Autoencoders with three and four neurons have very heavy degrees of overlap between classes, especially between the digits "4" and "9". This aligns with the prior visual exploration of autoencoder capabilities. On the other hand, even with bottleneck sizes of three and four neurons, the autoencoder can clearly separate digits like "0" and "1", indicated by their far distance from any other class. On the other hand, autoencoders with larger bottleneck sizes of 100 and 200 neurons have a much clearer separation of classes. Furthermore, each of the classes becomes slightly more "cohesive" in that they are less spread out.

Analysis of the latent space will be conducted on these t-SNE reduced-dimension representations of the coordinate space. There are several reasons for this choice. Firstly, because of the "Curse of Dimensionality" and the wide range of dimensions included in our analysis of latent spaces (1 dimension to 200 dimensions), it is not wise to compare statistical findings across the drastically different bottleneck sizes. Secondly, even if we were to use statistical methods on different dimensional latent spaces, we would find that they would not be very informational because of the nature of deep learning representations. It is well known that deep neural networks form representations of data on very complex manifolds, and thus performing simpler statistical calculations devised for simpler phenomena is ill-advised. For instance, the mean of all points on a circle is its center, but none of the points lie close to that mean. For evidence of this, view figures 3.9 and 3.10, which show a Principal Component Analysis reduction of latent spaces. The results demonstrate that using more linear statistical methods performs poorly at "understanding" the complex manifolds that lie in the latent spaces. However, the t-SNE algorithm is localized, meaning that it prioritizes the preservation of local structure over that of the global structure. It is thus able to "unravel" complex manifolds, yielding more accurate reductions of the relationships found by the autoencoder. Thirdly, being able to visually confirm our findings in two dimensions makes our analysis less prone to potential human errors that would not have been caught if working in un-intuitive higher dimensions. For these reasons, it is reasonable for our analysis to carry forth using t-SNE reduced representations of the autoencoder latent space.

A consequence of this decision, though, is that we will need to exclude the one-neuron and two-neuron bottlenecks from the analysis, since performing t-SNE on latent spaces of two dimensions and smaller is bad practice and cannot be compared with t-SNE results from higher dimensions. The autoencoder with two neurons would have been excluded from analysis anyway, since its training was disrupted, as demonstrated by the input/output visualization in Figure 2.3.

4 Analyzing Spread of the Latent Space

In this section, we will analyze how the "spread" of the latent space differs by the width of the autoencoder's bottleneck. Traditionally, the standard of spread for one-dimensional data has been standard deviation, defined as such:

$$\sigma = \sqrt{\frac{\sum (x_i - \bar{x})^2}{N - 1}}$$

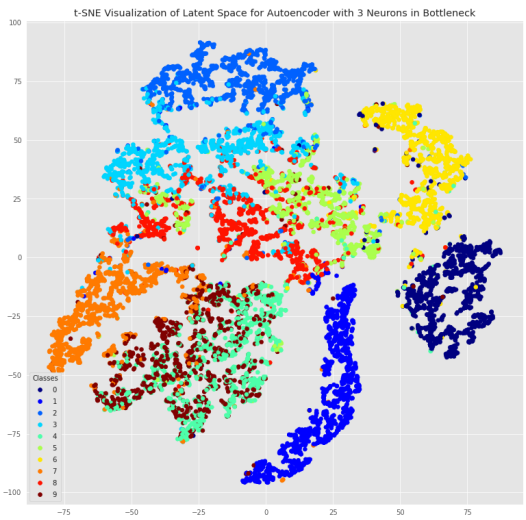

Figure 3.1: Bottleneck size of 3 neurons.

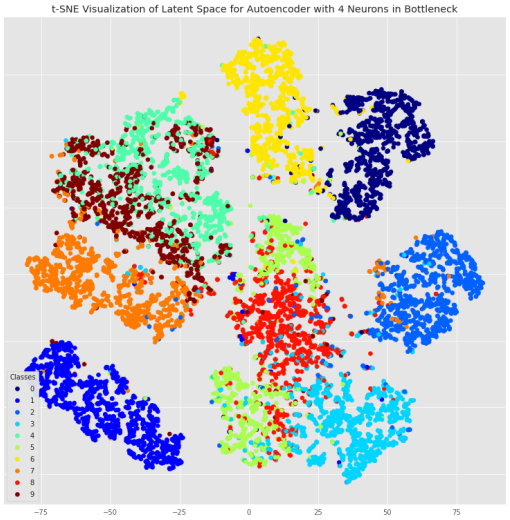

Figure 3.2: Bottleneck size of 4 neurons.

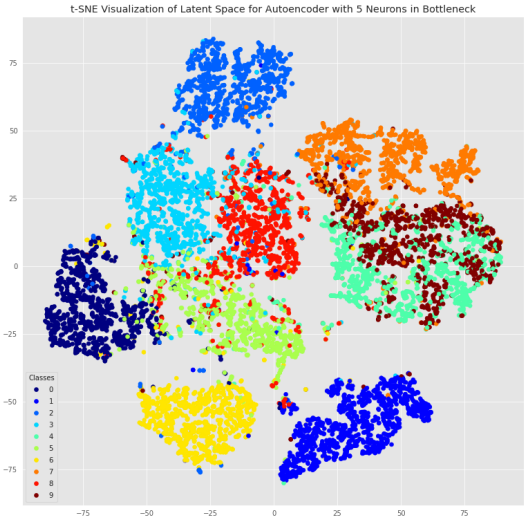

Figure 3.3: Bottleneck size of 5 neurons.

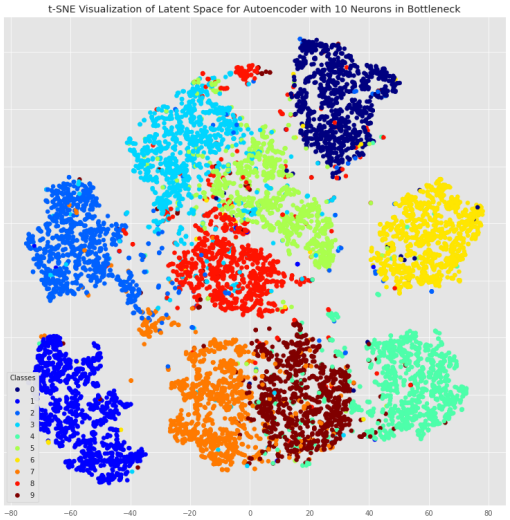

Figure 3.4: Bottleneck size of 10 neurons.

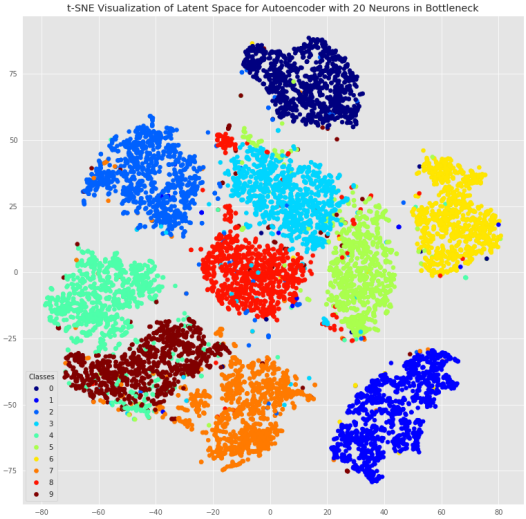

Figure 3.5: Bottleneck size of 20 neurons.

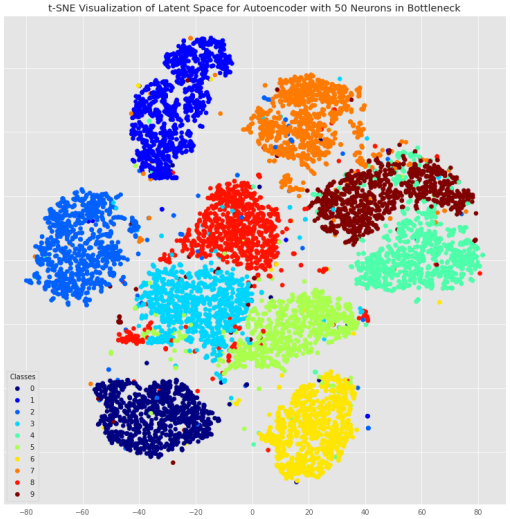

Figure 3.6: Bottleneck size of 50 neurons.

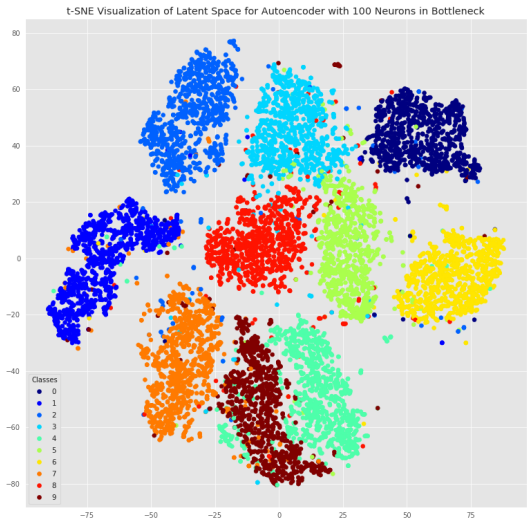

Figure 3.7: Bottleneck size of 100 neurons.

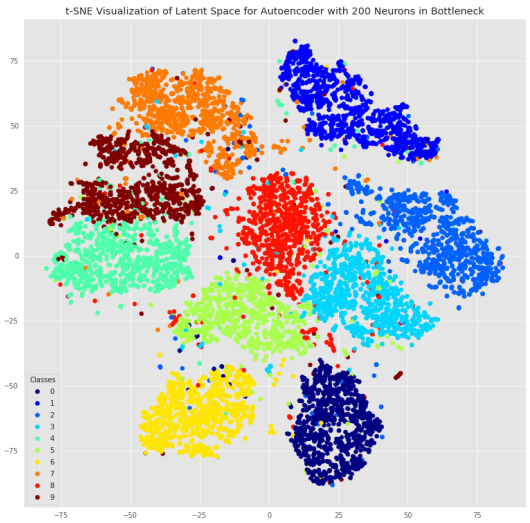

Figure 3.8: Bottleneck size of 200 neurons.

However, this does not work for two-dimensional data. Let us instead measure "spread" in a more general sense by the average distance from a "center point".

$$\frac{1}{N}\sum ||p_i - c||$$

Because we have reduced the latent space into two dimensions, we will need to adapt this notion of spread for a two-dimensional space. The center point can be defined as the average point, where $c = (\bar{x}, \bar{y})$ for all x and y coordinates. The distance between a particular point and the center can be calculated using Euclidean distance:

$$||p_i - c|| = \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}$$

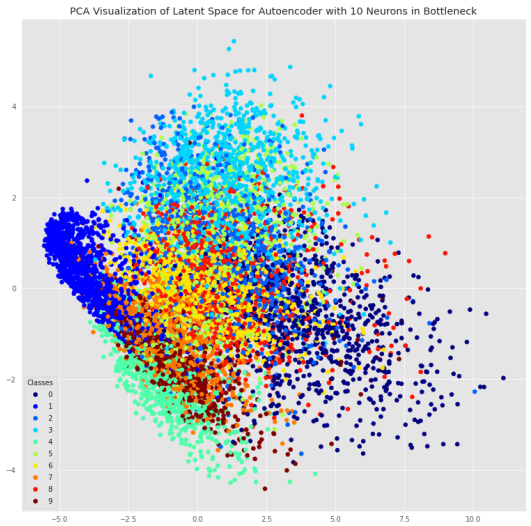

Figure 3.9: PCA visualization of 10-neuron latent space. Despite its inability to preserve local structure, we can still see that "1"s – a digit even the smallest-bottleneck autoencoders handled easily – form a distinct cluster.

Figure 3.10: PCA visualization of 200-neuron latent space. It is worth noting that even though PCA overlays various clusters on top of each other, which poses problems to statistical measures, as humans we can identify groups of digit clusters.

We can thus calculate the "spread" of the two-dimensional latent space as such:

$$s = \frac{1}{N} \sum \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}$$

This measure of spread will be used to analyze two contexts: firstly, the spread of the entire latent space without distinction for which digits each point belongs to, and secondly, the spread of data points that belong to a particular digit class. It should be noted that when the term "latent space" is used alongside spread, it refers to the t-SNE reduced latent space in two dimensions.

4.1 Spread Without Class Distinction

When considering the spread of entire latent spaces without regard for which class each data point belongs to, we find that there is not a significant decrease in the average distance from the mean of the entire latent space. Nevertheless, as demonstrated in figure 4.1, there seems to be a trend of decreasing in average distance from mean from 3 to 10 neurons, then a plateauing for larger bottleneck sizes.

Neurons	Avg. Distance from Mean
3	56.627600
4	55.017513
5	55.185037
10	53.422837
20	53.471431
50	52.561381
100	53.169275
200	52.400419

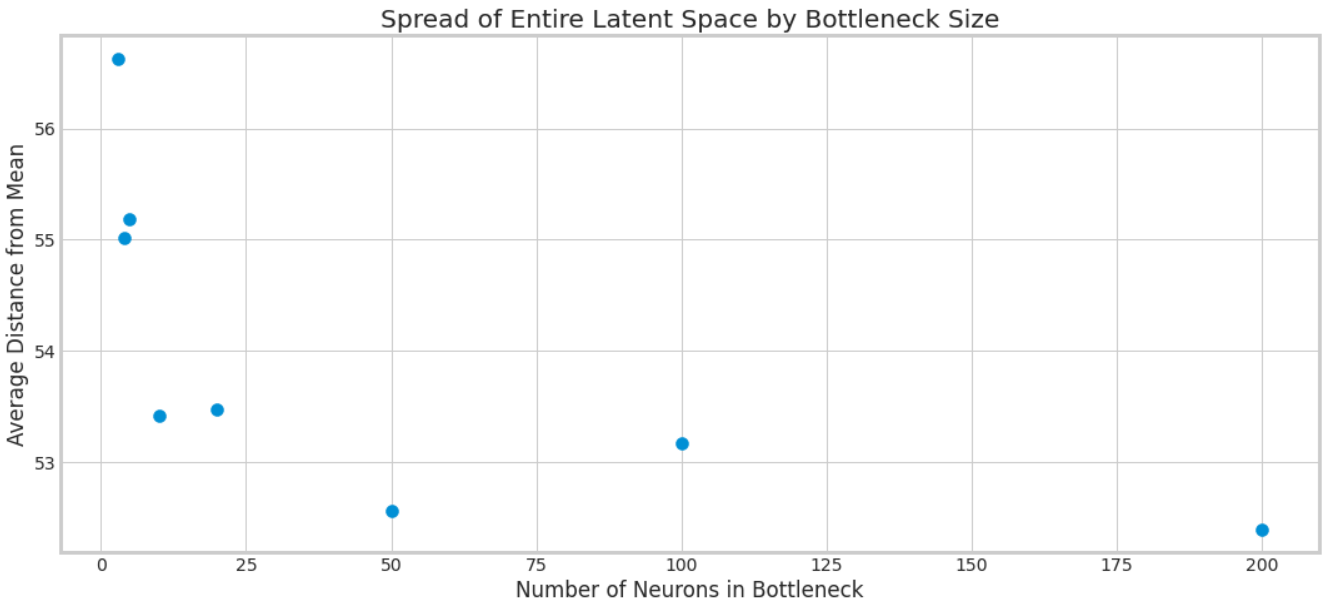

Figure 4.1: There is a downwards motion early on, then a plateauing.

Interestingly, this suggests that as the number of neurons in the bottleneck increases, spaces become less spread out and more "organized". One can confirm this visually with the emergence of cohesive blob-like shapes to replace more spread out and "structure-less" shapes in smaller bottleneck sizes in Figures 3.1 to 3.8. This suggests a greater structure and efficiency in representation attained with a larger bottleneck size.

4.2 Spread With Class Distinction

This subsection will consider the spread of "class clusters" – the set of points that belong to a particular digit. For each set of points (by autoencoder bottleneck size), the average distance from mean for 10 subsets of data points

were calculated, corresponding to which digit each of the subset data points fell into. The results are visualized in Figure 4.2 and listed in numerical form in a table below.

Separating the analysis by digit echos the general observation that as bottleneck size increased, the average distance from the mean drops suddenly, then seems to plateau. This is approximately true for each of the digits when individually analyzed. Curiously, the spread of the digit clusters seems to encounter a slight "bump" in the average distance from the mean as the number of neurons in the bottleneck increases. This small "bump" is a small increase following the steep drop in average distance from man, and precedes a continued dropping and then a plateau. This can be tracked by analyzing the average distance from means listed in the table. For instance, the spread of points in the digit 8 decreases as the number of neurons in the bottleneck increases from 22.51 to 18.25. At 20 neurons, it jumps to 20.34, then continues decreasing to 17.10, then plateaus at around 15.5. Although in terms of magnitude this small bump may not seem significant, it appears to varying degrees in every single digit except for the digit 6, which decreases steadily, and the digit 2, which curiously increases in its spread as the number of neurons in the bottleneck increases.

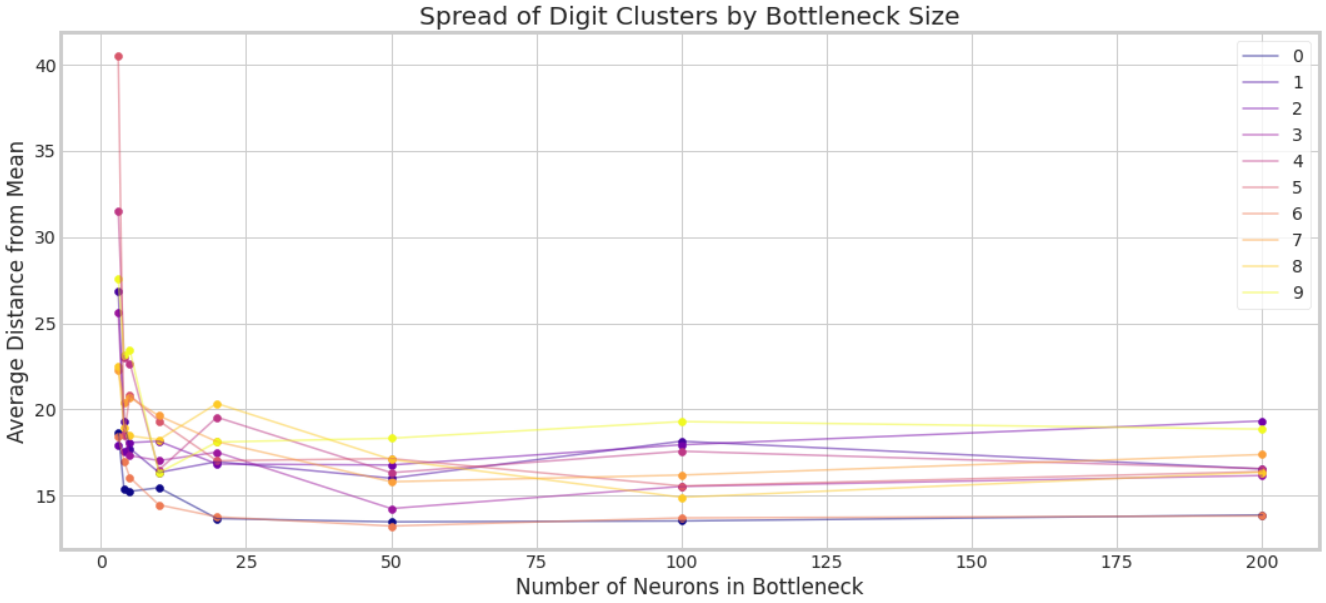

Figure 4.2: Connecting lines drawn to assist visual exploration, but do not imply the values of intermediate inputs.

Digit	3 neurons	4 neurons	5 neurons	10 neurons	20 neurons	50 neurons	100 neurons	200 neurons
0	18.68	15.36	15.25	15.46	13.68	13.48	13.53	13.88
1	26.86	19.31	17.74	16.35	16.97	16.01	18.16	16.55
2	17.96	17.57	18.08	18.16	16.84	16.78	17.95	19.33
3	25.65	18.60	17.33	17.03	17.52	14.26	15.54	16.17
4	31.53	22.98	22.67	16.49	19.55	16.33	17.58	16.57
5	40.51	18.51	20.82	19.34	17.03	17.15	15.56	16.40
6	18.42	17.00	16.05	14.47	13.77	13.24	13.71	13.83
7	22.25	20.42	20.70	19.63	18.13	15.81	16.20	17.39
8	22.51	18.93	18.47	18.25	20.34	17.10	14.90	16.36
9	27.57	23.18	23.43	16.33	18.11	18.33	19.30	18.86

Broadly, this trend is reminiscent of a deep learning phenomenon known as *deep double descent*, as visualized in Figure 4.3. In classical statistics, the bias-variance tradeoff suggests that models with too much bias underfit to the data (i.e. do not have the predictive power to model the data) and models with too much variance overfit to the data (i.e. have so much predictive power that it memorizes the data without learning underlying trends from it). Somewhere in between a high-bias and high-variance model is a model that has the capability to model the data, but is not complex enough to overfit to it. Deep learning has somewhat subverted this notion, though, as massively overparametrized neural networks have shown incredible performance on training data and test data. The Deep

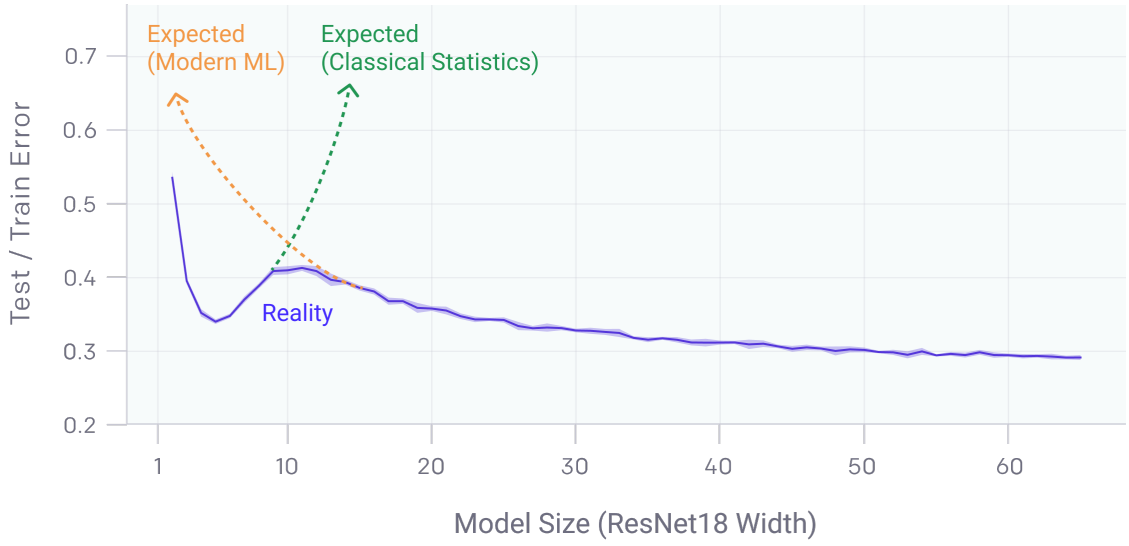

Figure 4.3: An illustration of deep double descent from OpenAI.

Double Descent phenomenon is a theoretically and empirically proven merging of classical statistics and modern deep learning paradigms on the relationship between the number of parameters in a model and its performance on test data. Ultimately, it produces something like a "W" without its right "/".

This visualization is similar to what has been demonstrated with spread over the number of neurons in the bottleneck. As has been explored in Section 4.1, it is reasonable to interpret the spread of a neural network's latent space as how efficient and confident its representations of inputs are. As such, even though autoencoders operate in an unsupervised setting, the spread of the encoder's representation of each digit roughly measures how well it would have performed at a task like digit classification in a supervised setting. Thus, it seems that these findings may have served as further demonstration of the existence of the deep double descent phenomenon in an unsupervised setting.

5 Analyzing Distance Between Centroids

Spread has given us one measure of how "cohesively" and efficiently the autoencoder is able to represent different digits. Yet, it is a measure of intercluster (cluster referring to a cluster of points belonging to the same digit class) efficiency; that is, how well the encoder groups inputs of the same digits together. As is common in machine learning research, the other component is intracluster efficiency, or how well the encoder separates data points of different digit classes.

While there are many measures of intercluster separation, one of the simplest is the distance between the centroids (means) of each cluster. This produces $9+8+\cdots+2+1=45$ different distances between each of the 10 centroids. We can average these distances to find the average distance between digit centroids, which serves as a measure of how well the encoder separates inputs of different classes.

Interestingly, we find that the distribution of the distances between each of the clusters is roughly normally distributed. A sample of distributions are displayed in Figures 5.1 to 5.4, which show a histogram with a kernel density estimate. This suggests that the mean will be an appropriate measure for the distances between centroids.

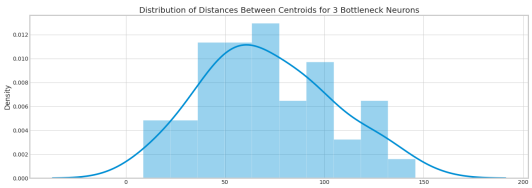

Figure 5.1: Bottleneck size of 3 neurons.

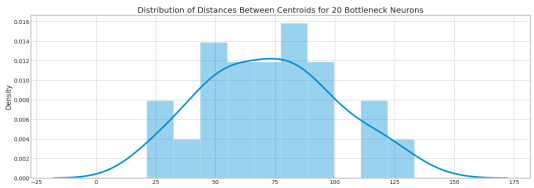

Figure 5.2: Bottleneck size of 20 neurons.

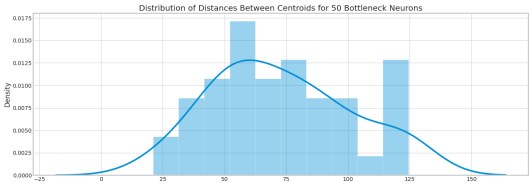

Figure 5.3: Bottleneck size of 50 neurons.

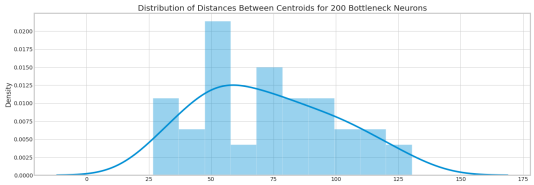

Figure 5.4: Bottleneck size of 200 neurons.

The mean of the distances between digit centroids for different bottleneck sizes are visualized in figure 5.5 and presented in numerical form in the table below. A visual inspection suggests that there is a similar deep double descent relationship here, in which the means of distances between clusters has the broader trend of decreasing. However, the average distance between digit centroids of 71.3254 for a 3-neuron autoencoder seems to disrupt this pattern.

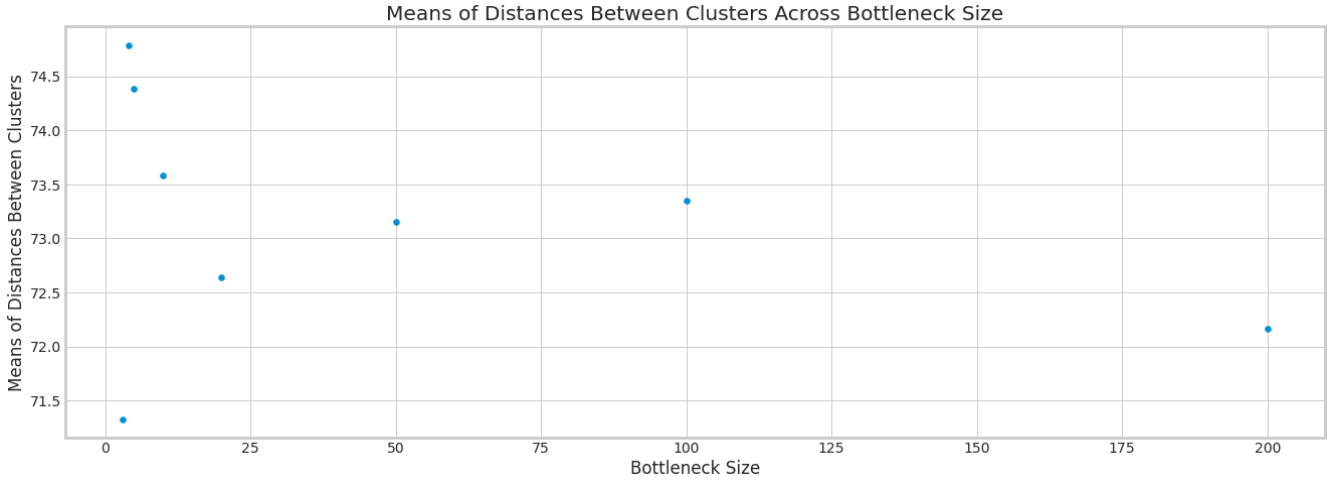

Figure 5.5: A relationship reminiscent of the deep double descent.

The Pearson correlation coefficient between the size of the bottleneck and the mean distance between centroids is -0.3355, suggesting a somewhat decreasing trend.

Neurons	Avg. Distance Between Digit Centroids
3	71.325413
4	74.791455
5	74.387227
10	73.585547
20	72.639914
50	73.149701
100	73.349573
200	72.163886

This is an unexpected result; one would have expected the distance between centroids to have increased given that the autoencoder is "better" at separating clusters of different digit classes. Moreover, it resembles the deep double descent curve, which suggests that the metric of mean distance between centroids may not be good for measuring model "understanding". The results likely suffer from lack of data; we would likely arrive at more conclusive results if we had generated more data for different bottleneck sizes.

It also seems natural given the roughly normal distribution of distances to analyze the standard deviation of distances between centroids. These are visualized in Figure 5.6 and presented numerically in the table below. The standard deviation of the distances between the centroids seems more conclusively to be decreasing. The Pearson correlation coefficient between the standard deviation of distances between clusters and the bottleneck size is -0.7222, suggesting a strong negatively correlated relationship.

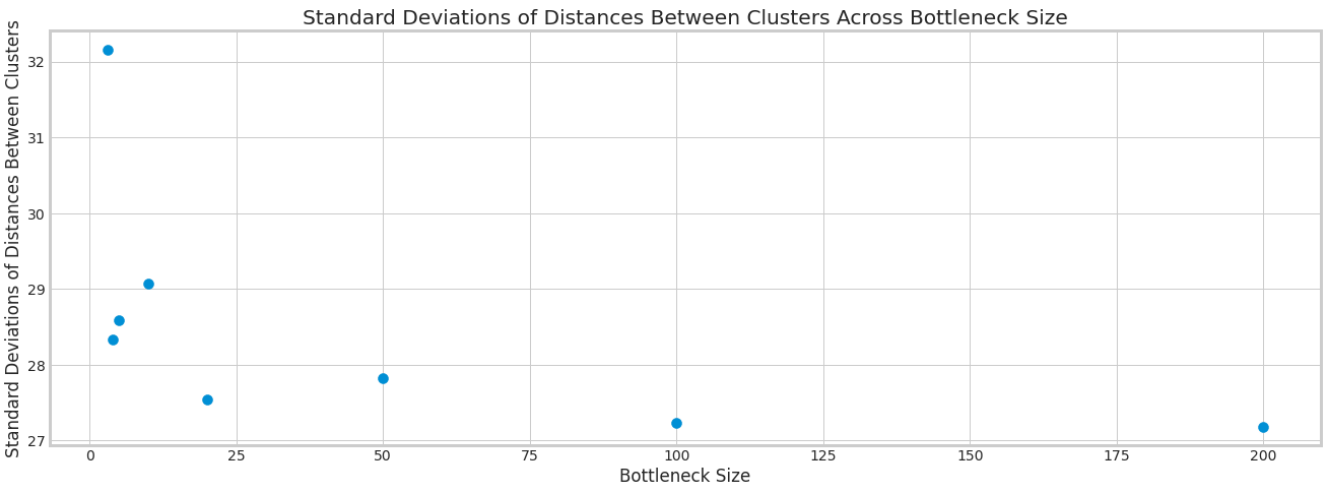

Figure 5.6: The standard deviation of distances between centroids seems more clearly to be decreasing.

Neurons	Standard Deviation of Distance Between Digit Centroids
3	32.164193
4	28.335293
5	28.589024
10	29.079061
20	27.547002
50	27.822664
100	27.236940
200	27.184293

This means that the distribution of distances between clusters across bottleneck size quite decidedly becomes less wide as the number of neurons increasing. Alternatively, we can interpret this as "the distances between each of the centroids become more uniform". This is indeed an interesting phenomenon to observe, as one might expect on an intuitive level for a model that has a better understanding of the data to arrange digits such that digits that are similar in structure are also closer together in the latent space. This has been observed with more complex embedding layers in larger neural network architectures employed on more complex data.

On a second look, however, these results seem to make sense. As demonstrated by the input/output visualizations in Figures 2.9 to 2.11, there is no meaningful difference between the predictive capability of autoencoders with 50, 100, and 200 neurons, although autoencoders with a larger number of neurons have a lower training loss. That is, the efficiency of a representation does not necessarily imply its performance. Autoencoders with a lower number of neurons work with a limited space, and thus they must find more efficient (but not necessarily accurate) representations by "merging together" similar digits. An analysis of Figures 3.1 through 3.8 show a visible "standardization" of the distances between clusters, whereas there is much more overlap in autoencoders with smaller bottleneck neurons, and hence more variation in distances between centroids and a higher standard deviation. In many ways, this inter-centroid phenomenon confirms prior intra-cluster analysis.

6 Summary of Findings

In Section 2.3. Specific Formulation of Inquiry, we outlined two major paths of inquiry – to explore the intracluster and intercluster relationships changed in the latent space as the number of neurons in the bottleneck of the autoencoder increased.

Using average distance from the mean as an intracluster measure of spread, we found that the spread of the entire latent space decreased as the number of neurons in the bottleneck increased, eventually plateauing at around 50 neurons. When analyzing the spread of digit clusters, we found in a deep double descent pattern in all but two of the digits, suggesting that the spread of the digit clusters may be correlated with the error of the representation in a supervised testing. This connection provides an interesting and novel link between the deep double descent phenomenon and unsupervised contexts.

Using the distance between centroids as an intercluster measure of spread, we found that the distribution of distances between digit clusters was roughly normally distributed. We found that the relationship between the bottleneck size and the mean distance centroids resembled that of the deep double descent; these results suggested that the mean distance of centroids may not be a good metric for evaluating a model's "understanding". We found that the standard deviation of distances between clusters decreases as the number of neurons in the bottleneck increases. This suggests that an increasing number of neurons puts less pressure on the autoencoder to develop extreme efficient representations and allows it to generate more uniform, or standardized, separations between digit clusters.

7 Further Inquiry

There are multiple paths of further inquiry and research that these results suggest.

- This paper only analyzed the behavior of eight autoencoders of different sizes. As such, our findings have had to generalize a fair bit across the behavior of these eight autoencoders. Further inquiry would explore if the findings of this paper hold up given more data for different bottleneck sizes.
- Each of the autoencoders were initialized randomly; this comes at the cost of potentially running into local minima or other problems by chance, as was demonstrated by the failed 2-neuron autoencoder in Figure 2.3. A follow-up would confirm if the findings in this paper hold up under multiple trials for each n-neuron bottleneck autoencoder. Such further inquiry would need to analyze the spread of results for autoencoders of one size; e.g. perhaps multiple trials for an autoencoder of bottleneck size 200 neurons yield less varied results for spread (distance from mean) than multiple trials for an autoencoder of bottleneck size 3 neurons.
- This paper used t-SNE dimensionality reduced data into two dimensions. Further inquiry would explore impacts of the Curse of Dimensionality and using simpler statistical measurements on complex manifolds on the results derived about the latent space. Such inquiry would provide a concrete grounding in the importance (or unimportance, perhaps) of the dimensionality reduction step.
- In Section 5, we analyzed the distance between centroids using the mean distance between centroids. However, this does not examine specific relationships between specific centroids, like between 4 and 9, which we have observed are perhaps the most difficult pair of digits for autoencoders to separate. Further inquiry would explore if it is possible to quantify how "difficult" a pair of digits are to separate by analyzing latent representation clusters.
- This paper used the MNIST dataset, which is a simple dataset, both in its size, image resolution, and number of classes. Further inquiry would repeat the steps in this paper on a much larger dataset, like ImageNet or CIFAR-100, and explore differences and similarities in findings based on the size of the dataset.

8 Code Reproduction Links

- Reproducing encoded data from the MNIST dataset.
- Reproducing t-SNE visualizations of latent space data.
- Reproducing PCA visualizations of latent space data.
- Reproducing intracluster and intercluster findings and visualizations.