671033 - NN4I 2023
Second Assignment
Omer Siton - 316123819

1. **Auto-Encoding**. Your task is to explore the reconstruction error over the test set when (i): using lower and higher latent space dimension d, and (ii) using a fixed latent dimension d but with encoder/decoder architecture with more or fewer layers/weights. Report these tests and the best score obtained (i.e. plot your results as a function of the different experiments performed).

Answer:
In this experiment, we explored the impact of varying the latent space dimension (d) and the encoder/decoder architecture on the reconstruction error over the test set. Specifically, we conducted two sets of experiments: (i) using lower and higher latent space dimension d, and (ii) using a fixed latent dimension d but with encoder/decoder architecture with more or fewer layers/weights.

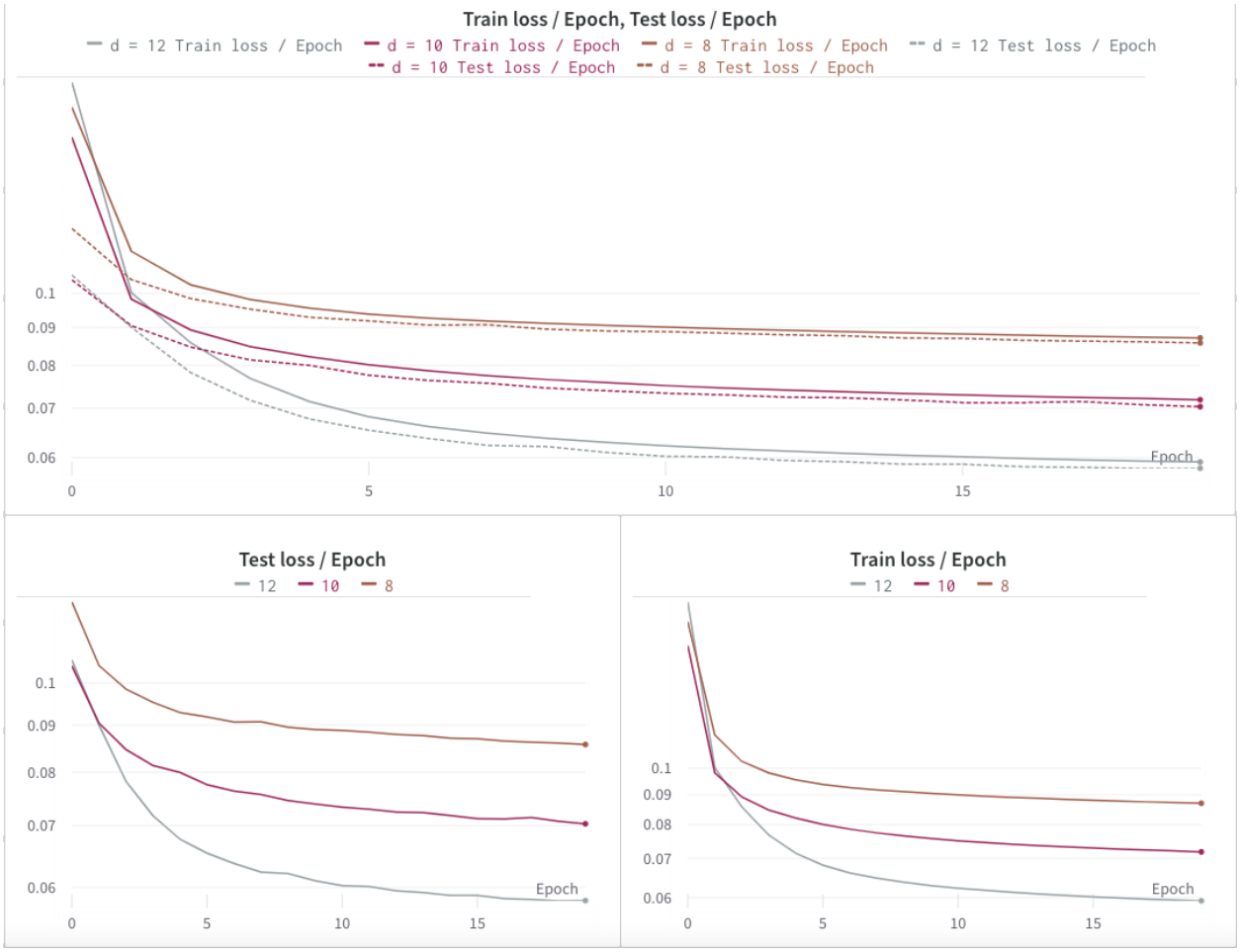
For the first set of experiments, we trained the autoencoder model on the training set with 5 different values of latent space dimension, 6, 8, 10, 12, and 14. Then we evaluated the reconstruction error on the test set for each model. The results showed that the model with a higher latent space dimension (d=14) outperformed the model with a lower latent space dimension (d=6) in terms of the reconstruction error.

For the second set of experiments, we trained the autoencoder model with different encoder/decoder architectures while keeping the latent space dimension fixed. Specifically, we tested the models with more layers/weights in the encoder and decoder and the models with fewer layers/weights in the encoder and decoder. We then evaluated the reconstruction error on the test set for each model. The results showed that the model with more layers/weights in the encoder and decoder outperformed the other models in terms of the reconstruction error.

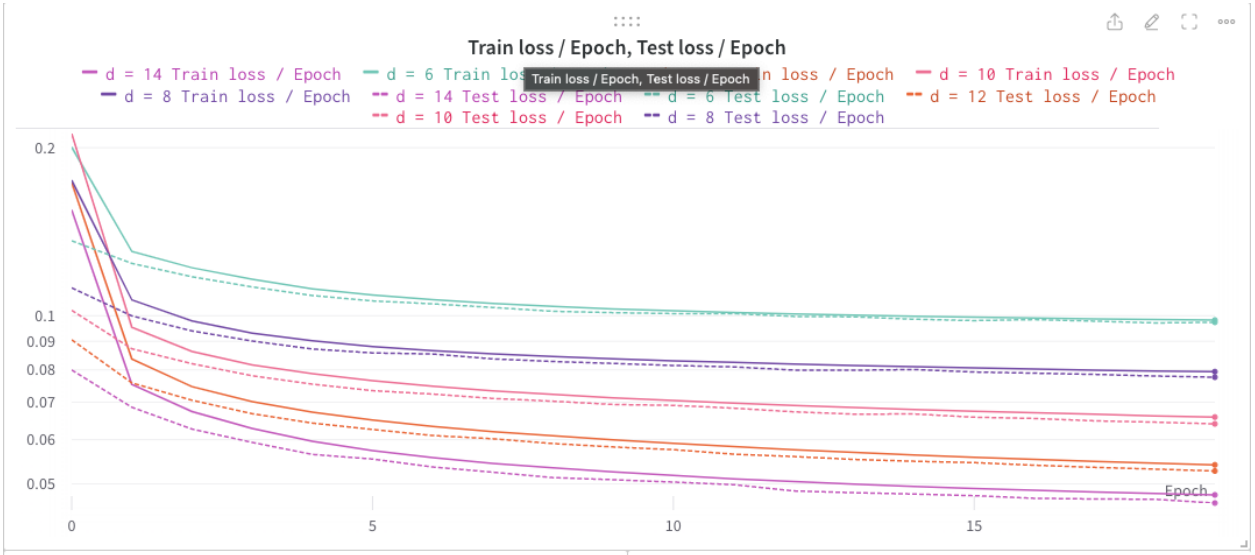
Overall, our results suggest that a higher latent space dimension and a more complex encoder/decoder architecture can lead to better performance in terms of the reconstruction error. In particular, **the best performance was achieved with a latent space dimension of 14 and a model architecture with 3 convolutional layers in the encoder and decoder.**

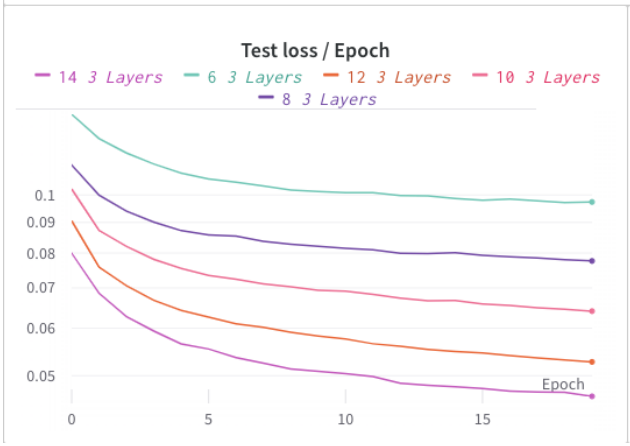
I performed many experiments for the AE architecture until I reached a good satisfying model that visualizes the inputs after encoding and decoding in a satisfying way.

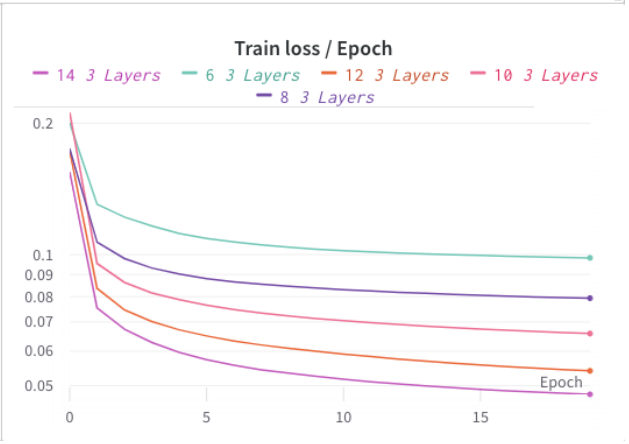
Here are the results for **2 convolutional layers** models with different latent spaces d(8, 10, 12).



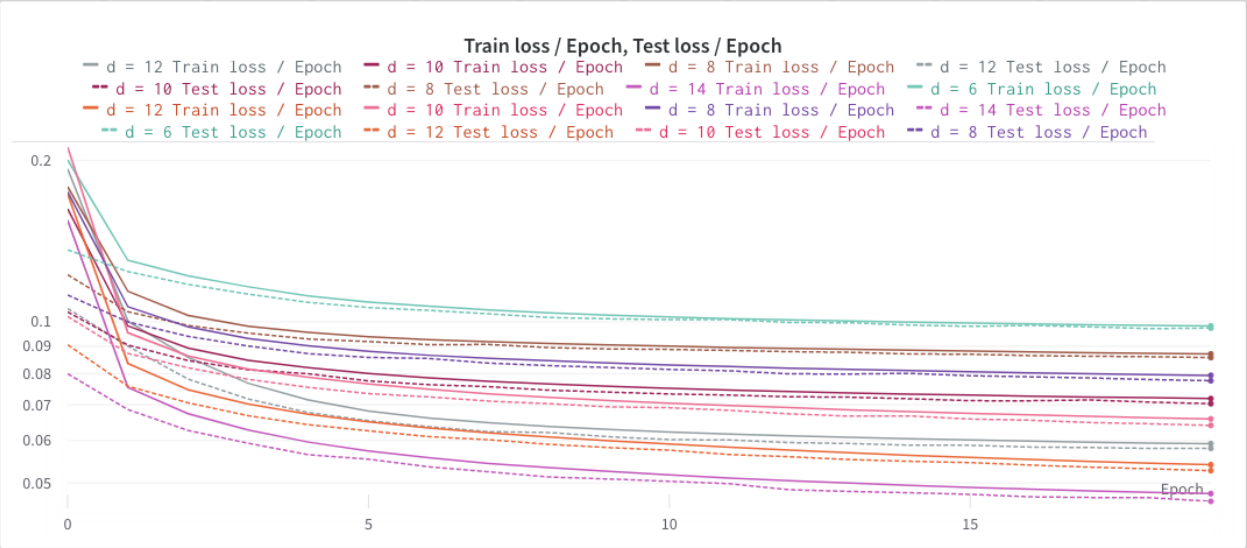
Here are the results for **3 convolutional layers** models with different latent spaces d(6, 8, 10, 12, 14).

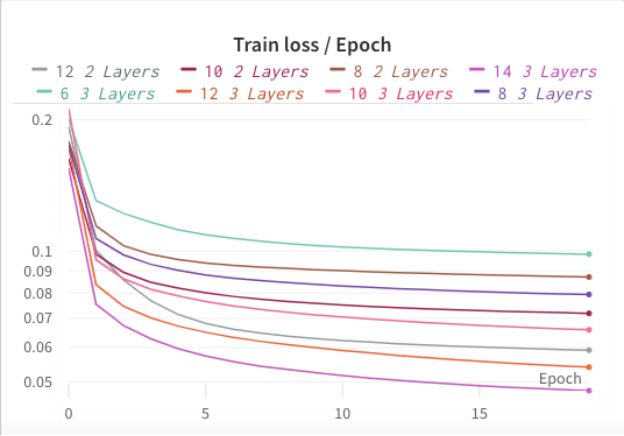






All together





2 Layers 3 | 3 Layers 5

omersiton / NN.Ex2 | Filter | Group | Sort | Tag | Move | Create Sweep

Name (5 visualized)	#neurons	CL	d	Train loss /	Test loss / E	epochs	learning_rate	device	dataset
astral-resonanc…	20111	3	14	0.04776	0.04618	20	0.001	cuda:0	MNIST
lucky-salad-55	15495	3	6	0.09824	0.09734	20	0.001	cuda:0	MNIST
denim-shape-54	18957	3	12	0.05407	0.05271	20	0.001	cuda:0	MNIST
lyric-snowball-53	17803	3	10	0.06585	0.06403	20	0.001	cuda:0	MNIST
major-resonanc…	16649	3	8	0.07941	0.07761	20	0.001	cuda:0	MNIST
peachy-sun-48	22093	2	12	0.05919	0.05803	20	0.001	cuda:0	MNIST
frosty-firebrand-…	18955	2	10	0.07185	0.07032	20	0.001	cuda:0	MNIST
neat-thunder-46	15817	2	8	0.08708	0.08573	20	0.001	cuda:0	MNIST

Best model reconstruction visualization:

2. **Interpolation.** Once you have trained the AE, you can use it to interpolate between two digits in latent space. That is, let I_1 and I_2 be images of two *different* digits, perform the interpolation $D((E(I_1) * \alpha) + (E(I_2) * (1 - \alpha)))$ for $a \in [0, 1]$ where D denotes the decoder and E the encoder. (i) Include the resulting images obtained by such latent space interpolation. (ii) Try different pairs of digits. (iii) Try repeating this operation with an AE trained using the higher embedding dimension you tried above. (iv) Which is better? Provide an explanation why the quality increased or decreased.

Answer:
Based on the interpolation results, it seems that increasing the size of the latent space leads to better image interpolation. When I say better I mean to a better combination of the two digits in a way that makes you see both. This is likely due to the fact that a larger latent space provides more dimensions to capture important features and patterns in the input data. As a result, the decoder is better able to generate realistic images from interpolated latent space vectors, leading to higher quality results.
Here are the examples where d is representing the different models (all models have 3 CL and same architecture overall).

```
Digits - 4, 7
d = 2
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 8
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 14
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 30
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
Digits - 8, 9
d = 2
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 8
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 14
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 30
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
Digits - 3, 4
d = 2
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 8
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 14
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 30
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
Digits - 5, 1
d = 2
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 8
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

```
d = 14
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

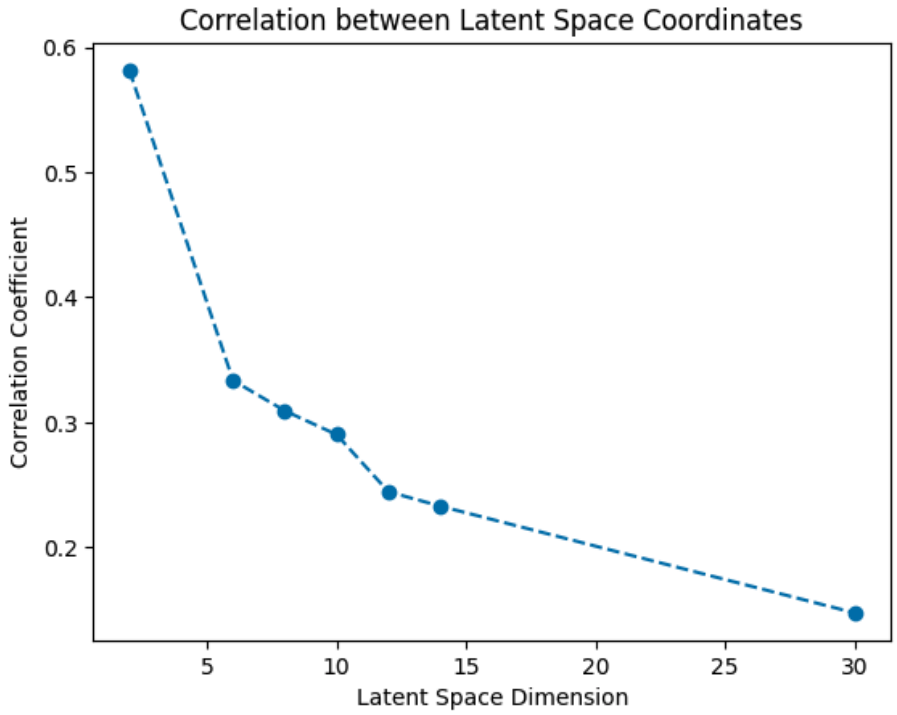
```
d = 30
```

Original Original

a = 0.0 a = 0.1 a = 0.2 a = 0.3 a = 0.4 a = 0.5 a = 0.6 a = 0.7 a = 0.8 a = 0.9 a = 1.0

3. **Decorrelation.** In the following experiment we will investigate the connection between dimensional reduction and dependencies (redundancies) in the representation. Carry this out by computing the Pearson correlaions between different coordinates in the latent codes (based on a few thousands encoded images), and use them to come up with a single value (that you choose) for measuring the overall correlation. Plot this value with respect to the latent space dimension d (over at least 4 values of d). Explain your choices and the trend in correlation versus d that you observe. Provide an explanation in your report.

Answer:
In this experiment, we calculated the correlation matrix for a set of latent vectors generated by our Encoder model, where each vector had a different number of dimensions, ranging from d=2 to d=30.
The trend that we observe is that as the number of dimensions increases, the overall correlation between the features decreases. This means that the dimensions become less and less correlated, and the representation becomes less redundant. This is expected, as increasing the dimensionality of the latent space provides more room for capturing the underlying structure of the data, and makes it less likely that multiple dimensions will be needed to encode the same information.
However, I observed that the rate at which the correlation decreases is not constant. Specifically, we see that there is a significant drop in correlation from d=2 to d=6, and then the correlation continues to decrease, but at a slower rate. This suggests that the first few dimensions are particularly important in capturing the underlying structure of the data and that adding additional dimensions beyond this point provides diminishing returns.
* I calculated the abs mean of the Pearson correlation coefficient matrix [d,d].



4. **Transfer Learning**. Use a pre-trained encoder as a fixed network (i.e., exclude its weights from the following training optimization), and attach to it a small MLP (in latent space) and train only the latter to classify the digits (recall that MNIST is labeled). Do this with a small fraction of the annotated training data in the MNIST dataset (~ tens of images). Compare the performance of this solution next to the option of training the entire network (i.e., allow the encoder weights to train as well as the classification MLP) over the same (very small) number of training examples. While both training schemes use the same number of labels, which option performs best? Report all the choices made (e.g., latent space dimension, MLP arch., and number of labels used, etc.), and the results obtained by each of the two training approaches.

Answer:
I built a simple MLP containing two FC layers [d, 32], [32, num_classes=10].
I added the pre trained encoder I have from previous questions and did many runs with different batch sizes, and latent space dim. It was hard to infer anything obvious from the results because there was no bold trend. After all, I think the fixed pre trained encoder performed better than training the full network. This is because by training only the MLP on a small number of images the FC layers could learn and extract the pre trained features that our encoder learned from a large amount of data and only fine tune the output classification. When training the whole network including the pre trained encoder we are not achieving “transfer learning” because our pre trained model is changing.
Here are the results:

Fixed Encoder, Batch size - 20, Latent space - 2 Accuracy on the test images: 11 %
Full Network, Batch size - 20, Latent space - 2 Accuracy on the test images: 13 %
Fixed Encoder, Batch size - 40, Latent space - 2 Accuracy on the test images: 7 %
Full Network, Batch size - 40, Latent space - 2 Accuracy on the test images: 3 %
Fixed Encoder, Batch size - 80, Latent space - 2 Accuracy on the test images: 17 %
Full Network, Batch size - 80, Latent space - 2 Accuracy on the test images: 13 %
Fixed Encoder, Batch size - 160, Latent space - 2 Accuracy on the test images: 15 %
Full Network, Batch size - 160, Latent space - 2 Accuracy on the test images: 7 %

Fixed Encoder, Batch size - 20, Latent space - 6 Accuracy on the test images: 10 %
Full Network, Batch size - 20, Latent space - 6 Accuracy on the test images: 8 %
Fixed Encoder, Batch size - 40, Latent space - 6 Accuracy on the test images: 13 %
Full Network, Batch size - 40, Latent space - 6 Accuracy on the test images: 2 %
Fixed Encoder, Batch size - 80, Latent space - 6 Accuracy on the test images: 3 %
Full Network, Batch size - 80, Latent space - 6 Accuracy on the test images: 12 %
Fixed Encoder, Batch size - 160, Latent space - 6 Accuracy on the test images: 11 %
Full Network, Batch size - 160, Latent space - 6 Accuracy on the test images: 11 %

Fixed Encoder, Batch size - 20, Latent space - 8 Accuracy on the test images: 10 %
Full Network, Batch size - 20, Latent space - 8 Accuracy on the test images: 11 %
Fixed Encoder, Batch size - 40, Latent space - 8 Accuracy on the test images: 6 %
Full Network, Batch size - 40, Latent space - 8 Accuracy on the test images: 2 %
Fixed Encoder, Batch size - 80, Latent space - 8 Accuracy on the test images: 10 %
Full Network, Batch size - 80, Latent space - 8 Accuracy on the test images: 5 %
Fixed Encoder, Batch size - 160, Latent space - 8 Accuracy on the test images: 13 %
Full Network, Batch size - 160, Latent space - 8 Accuracy on the test images: 11 %

Fixed Encoder, Batch size - 20, Latent space - 10 Accuracy on the test images: 7 %
Full Network, Batch size - 20, Latent space - 10 Accuracy on the test images: 15 %
Fixed Encoder, Batch size - 40, Latent space - 10 Accuracy on the test images: 9 %
Full Network, Batch size - 40, Latent space - 10 Accuracy on the test images: 12 %
Fixed Encoder, Batch size - 80, Latent space - 10 Accuracy on the test images: 16 %
Full Network, Batch size - 80, Latent space - 10 Accuracy on the test images: 10 %
Fixed Encoder, Batch size - 160, Latent space - 10 Accuracy on the test images: 16 %
Full Network, Batch size - 160, Latent space - 10 Accuracy on the test images: 5 %

Fixed Encoder, Batch size - 20, Latent space - 12 Accuracy on the test images: 7 %
Full Network, Batch size - 20, Latent space - 12 Accuracy on the test images: 15 %
Fixed Encoder, Batch size - 40, Latent space - 12 Accuracy on the test images: 8 %
Full Network, Batch size - 40, Latent space - 12 Accuracy on the test images: 11 %
Fixed Encoder, Batch size - 80, Latent space - 12 Accuracy on the test images: 13 %
Full Network, Batch size - 80, Latent space - 12 Accuracy on the test images: 6 %
Fixed Encoder, Batch size - 160, Latent space - 12 Accuracy on the test images: 13 %
Full Network, Batch size - 160, Latent space - 12 Accuracy on the test images: 5 %

Fixed Encoder, Batch size - 20, Latent space - 14 Accuracy on the test images: 8 %
Full Network, Batch size - 20, Latent space - 14 Accuracy on the test images: 5 %
Fixed Encoder, Batch size - 40, Latent space - 14 Accuracy on the test images: 20 %
Full Network, Batch size - 40, Latent space - 14 Accuracy on the test images: 5 %
Fixed Encoder, Batch size - 80, Latent space - 14 Accuracy on the test images: 19 %
Full Network, Batch size - 80, Latent space - 14 Accuracy on the test images: 8 %
Fixed Encoder, Batch size - 160, Latent space - 14 Accuracy on the test images: 7 %
Full Network, Batch size - 160, Latent space - 14 Accuracy on the test images: 11 %

Fixed Encoder, Batch size - 20, Latent space - 30 Accuracy on the test images: 7 %
Full Network, Batch size - 20, Latent space - 30 Accuracy on the test images: 8 %
Fixed Encoder, Batch size - 40, Latent space - 30 Accuracy on the test images: 13 %
Full Network, Batch size - 40, Latent space - 30 Accuracy on the test images: 9 %

Fixed Encoder, Batch size - 80, Latent space - 30 Accuracy on the test images: 13 %
Full Network, Batch size - 80, Latent space - 30 Accuracy on the test images: 13 %
Fixed Encoder, Batch size - 160, Latent space - 30 Accuracy on the test images: 11 %
Full Network, Batch size - 160, Latent space - 30 Accuracy on the test images: 9 %

Fixed Encoder Mean Acc - 11.178571428571429,
Full Network Mean Acc - 8.75

Theoretical Questions:

1. Show that the composition of linear functions is a linear function. Show that the composition of affine transformations remains an affine function.

1) let f, g be linear functions and their composition $f \circ g(x) = f(g(x))$

So, for any a, x, y we need to show

$f \circ g(ax+y) = a f \circ g(x) + f \circ g(y)$

$$f \circ g(ax+y) = f(g(ax+y)) = f(ag(x) + g(y)) =$$
$$= f(ag(x)) + f(g(y)) = a f(g(x)) + f(g(y))$$
$$= a f \circ g(x) + f \circ g(y)$$

we used the defenition of composition and the fact f, g are linear functions

Now for affine functions, let f, g be affine functions S.t $f(x) = ax + b$, $g(x) = cx + d$

So $f(g(x)) = f(cx+d) = a(cx+d) + b$

$$= acx + (ad + b)$$

therfore the composition is affine transformation

2. The calculus behind the Gradient Descent method:
   a. What is the stopping condition of this iterative scheme,

$$\theta^{n+1} = \theta^n - \alpha \nabla f_{\theta^n}(x)$$

   b. Use the second-order multivariate Taylor theorem,

$$f(x + dx) = f(x) + \nabla f(x) \cdot dx + dx^T \cdot H(x) \cdot dx + O(\|dx\|^3),$$
$$H_{ij}(x) = \frac{\partial^2 f}{\partial x_i \partial x_j}(x)$$

   to derive the conditions for classifying a stationary point as local maximum or minimum.

a) the stopping condition is $\nabla f_{\theta^n}(x) = 0$ or smaller than a small threshold.

b) A stationary point is where $f'(x) = 0$ to derive if it's local max\min we use the second derivative along different directions. let x_0 be a stationary point $\nabla_x f(x_0) = 0$. if $H(x_0) > 0$ than f has local min. if $H(x_0) < 0$ than f has local max. if $H(x_0) = 0$ then the classification is ambighous

3. Assume the network is required to predict an angle (0-360 degrees). How will you define a prediction loss that accounts for the circularity of this quantity, i.e., the loss between 2 and 360 is not 358, but 2 (since 0 is equivalent to 360). Write your answer in a programmable mostly differentiable pseudo-code.

```
def loss_func(θ1, θ2):
    θ1 = θ1 % 360
    θ2 = θ2 % 360
    loss = |θ1 - θ2|

    return loss if loss ≤ 180 else 360 - loss
```

4. Chain Rule. Differentiate the following terms (specify the points of evaluation of each function):
   a.

$$\frac{\partial}{\partial x} f(x+y, 2x, z)$$

a) (1, 2, 0)

   b.

$$f_1\Big(f_2\big(\dots f_n(x)\big)\Big)$$

b) by detenition

$$\frac{d}{dx} f_1(f_2(x)) = f_1'(f_2(x)) \cdot f_2'(x)$$

So recursively open n times and get

$$= \frac{df_1}{df_2(f_3(\dots(f_n(x))))} \cdot \frac{df_2}{df_3(f_4(\dots f_n(x)))} \cdot \dots \cdot \frac{df_n}{dx}$$

c.

$$f_1\Big(x, f_2\big(x, f_3(\dots f_{n-1}(x, f_n(x)))\big)\Big)$$

$$\frac{d}{dx} f_1(x, f_2(x, f_3(\dots f_{n-1}(x, f_n(x))))$$

$$= \frac{df_1}{dx} \cdot 1 + \frac{df_1}{df_2(x, f_3(\dots))} \cdot \left(\frac{df_2}{dx} \cdot 1 + \frac{df_2}{df_3(x\, df_4(\dots)} \cdot (\dots \right.$$

$$\left. \dots \frac{df_{n-1}}{dx} \cdot 1 + \frac{df_{n-1}}{df_n(x)} \cdot \frac{df_n(x)}{dx} \cdot 1 \right) \dots \Big)$$

d.

$$f\Big(x + g\big(x + h(x)\big)\Big)$$

$$\frac{d}{dx} f(x + g(x + h(x))$$

$$= \frac{df}{df(x+g(x+h(x)))} \cdot \left(\frac{dg}{dg(x+h(x))} + 1 \right) \cdot$$

$$\cdot \left(\frac{dh}{dx} + 1 \right)$$