

רשתות נוירונים לתמונות | תרגיל 2

שם: תמוז גיטלר | ת"ז: 312243439

23 באפריל 2023

Practical Questions:

Question 1 - Auto-Encoding

Intro)

Because the convolutional AE in this exercise is built to train on the mnist dataset and to reconstruct images from low-dimensional latent space this is a regression problem and not classification. therefore i decided to use the MSE loss instead of the BCE loss that was recommended in class to use when applying sigmoid on the last layer of the decoder.

In addition i found out that using AdamW optimizer that adds a weight decay term to the weight update rule in the L2 regularization term that is used in the original Adam algorithm improves the performense of the AE.

I) using lower and higher latent space dimension d

In the following testLoss and TrainLoss images provided below, i used the same AE model while changing the latent space dimension.

The model:

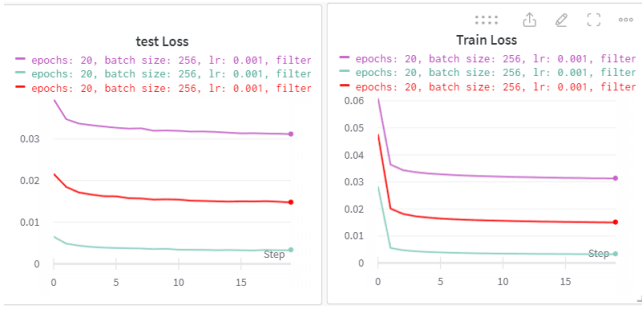
epochs: 20, batch size: 32, lr: 0.001, filter size: (3, 3)

AE color	Latent space dimension
pink	4
red	10
light blue	40

When defining the latent space dimension the programer sets the dimmension of the bottleneck layer (between the encoder and the decoder), which controls the amount of inforamtion the Encoder compresses from the input.

From exploring the reconstruction error over the test set while chainging the latent space dimension i deduce that using bigger dimension improves the model performance because the AE produces more complex representations of the provided data, which can cause overfit.

In contrast, when using small dimension the encoder might underfit, because he won't be able to learn enough features from the input data.



II) using a fixed latent dimension d but with encoder/decoder architecture with more or fewer layers/weights

The original AutoEncoder i built had 2 convolution layers and 1 fc layer.

AE color	Num of convolution layers
green	1
red (original model)	2
blue	3

When i used an AutoEncoder with 3 convolutional layers instead of 2 the model was able to capture complex patterns and dependencies in the data, hence the test and train loss were lower.

When i used an AutoEncoder with 1 convolutional layers instead of 2 the model was less capable of capturing complex patterns and dependencies in the data, hence the test and train loss were higer.



In concultion when building an AutoEncoder network and deciding the number of layers used in the model, there is a tradeoff between:

1. the model's capacity to capture complex patterns and dependencies in the data and the risk of overfitting.
2. the computational expence of training the model and the risk of underfitting.

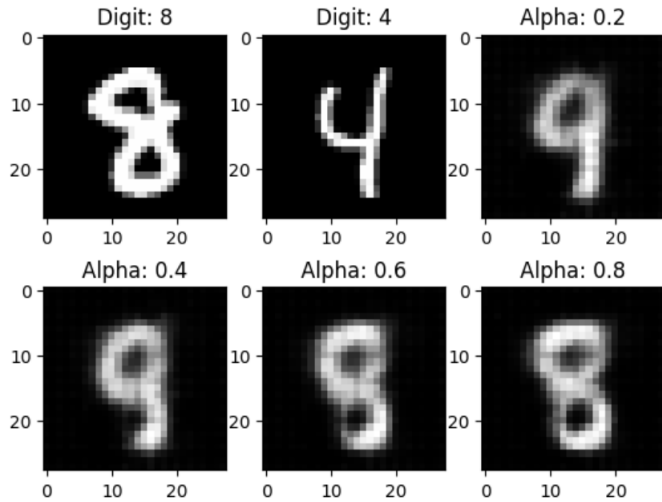
Question 2 - Interpolation

In the following images i repetead the same process (interpolation) with three autoEncoders that had the same structure but different latent dimension - $[4,10,40]$ (they were all trained on the same mnist train dataset)

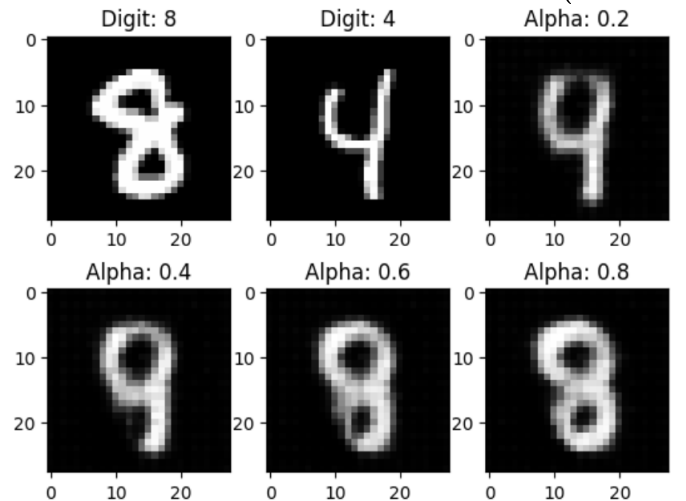
Because $\alpha \in [0, 1]$ i decided to interpolate with different α values $[0.2, 0.4, 0.6, 0.8]$ to see the values effect on the interpolated image.

Interpolation between digits 8 and 4

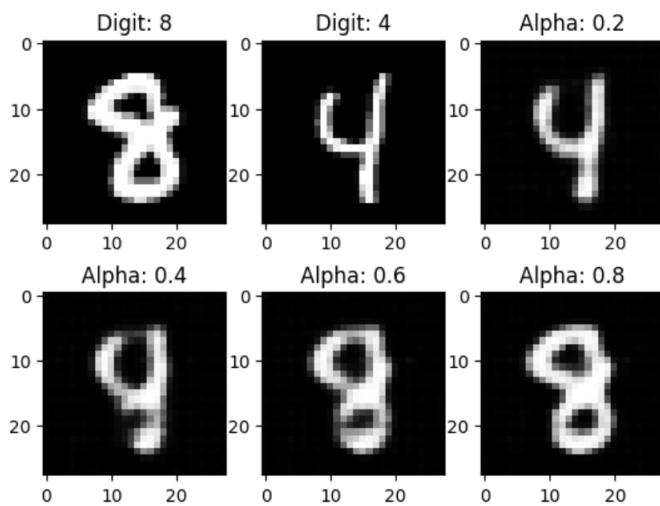
1. autoEncoder with latent dim: 4 (lower embedding dimension)



2. autoEncoder with latent dim: 10 (default value)

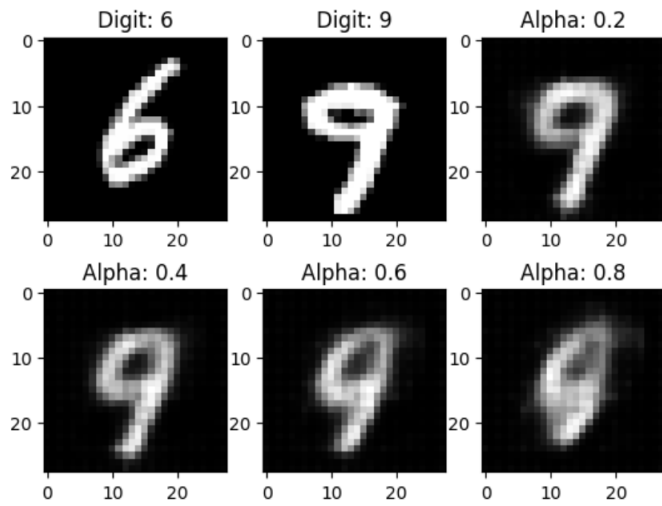


3. autoEncoder with latent dim: 40 (higher embedding dimension)

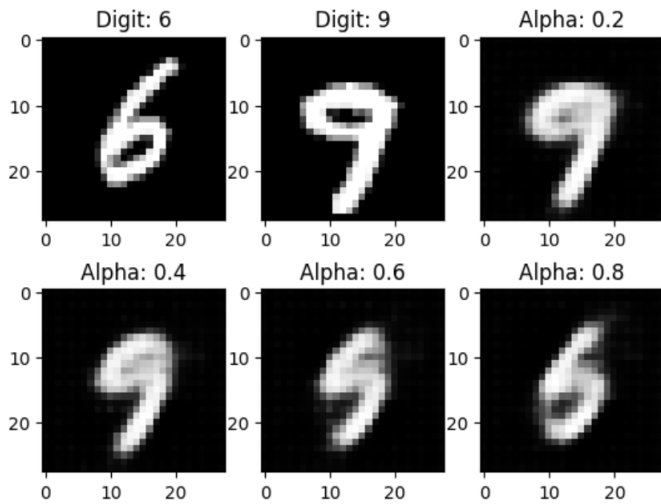


Interpolation between digits 6 and 9

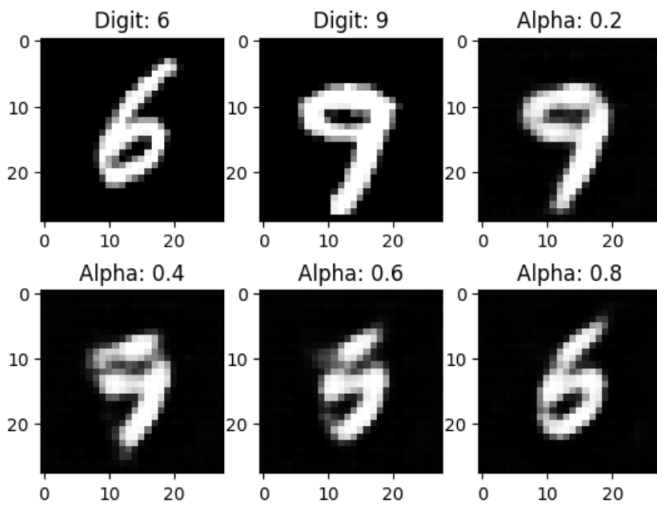
1. autoEncoder with latent dim: 4 (lower embedding dimension)



2. autoEncoder with latent dim: 10 (default value)



3. autoEncoder with latent dim: 40 (higher embedding dimension)



As i predicted, when α value was closer to 0 the autoEncoder reconstructed an image which looked closer to to digit1.




and when α value was closer to 1 the autoEncoder reconstructed an image which looked closer to to digit2.

when i used α with value around 0.5 the autoEncoder reconstructed an image that looked like a combination of both digits, which most of the time didnt look like a legal digit in range [0,9].

In conclution, when interpolation between two images from the mnist dataset, i would chose α that is close to 0 or 1, but not close to 0.5.

When reconstructing the interpolated digit the autoEncoder who reconstructed the best images (better quality) was the autoEncoder who had the higher embedding dimension (40).

for example, interpolation between digits 6 and 9 while using $\alpha = 0.2$:

Latent space dimension	4	10	40
Reconstructed Image			

My assumption on why using AutoEncoder with higher latent dimension reconstructs images with higher quality is that this autoEncoder

can represent more variations in the input images, which enables the decoder to generate more diverse and realistic (looks more like a legal digit) output images during interpolation. additionally, the AutoEncoder is able to better preserve the fine details of the input images (edges of the digits...).

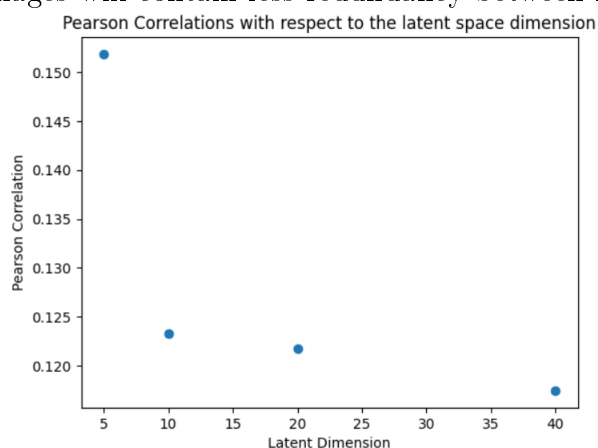
Question 3 - Decorrelation

To compute the pearson correlation i trained 4 autoEncoders that used the following latent dimension [5,10,20,40].

For each autoEncoder i created latent vectors (encoded images) based on 4,000 images from the training set. Then i calculated the correlation matrix (using np.corrcoef) for all the different pairs of coordinates in the encoded images.

and then i computed the overall correlation between the different coordinates which resulted a single value.

In the graph below we can see that when using encoders with higher latent dimension the Pearson correlation decreases (inverse relationship). my assumption is that higher dimensional encoders are able to learn **more** diverse set of features, which are less correlated between each other, that means that the encoded images will contain less redundancy between its coordinates.



Question 4 - Transfer learning

Parameters:

epochs: 10, batch size: 20, lr: 0.001, filter size: (3, 3), latent_dim: 10 , number of labels: 10 (as digits 0-9), training set: 80 images

MLP Arch:

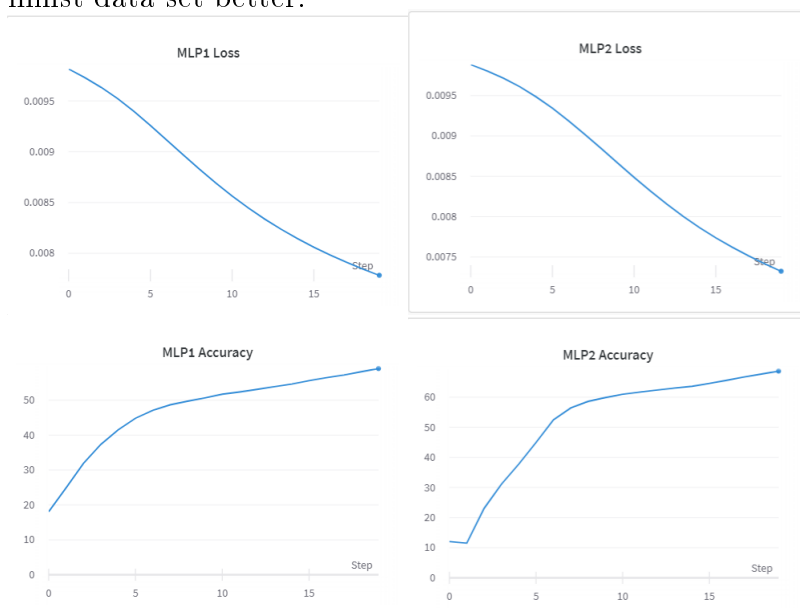
i used three fully connected layers, on each layer i activated Relu, and because we are labeling the images into digits i used Softmax at the end. the dimensions of the layers were:

in_features=10, out_features=20

in_features=20, out_features=30

in_features=30, out_features=10

as you can see below, the first model which excluded the encoders weights from the following training optimization perform worse (59% accuracy) than the second model (68% accuracy) who fine tune its encoder by letting him to optimize its weights. by fine tuning the encoders weights the model was able to classify the minst data set better.



Theoretical Questions:

Question 1

Definition of Linear function:

1. Additivity - $h(x + y) = h(x) + h(y)$

2. scalar multiplication - $h(ax) = a \cdot h(x)$

composition of linear functions is a linear function

let f,g be linear functions:

$$g : \mathbb{R}^n \rightarrow \mathbb{R}^m, f : \mathbb{R}^m \rightarrow \mathbb{R}^k, \quad (x, y) \in \mathbb{R}^n, a \in \mathbb{R}$$

proof:

1. Additivity

$$\begin{aligned} (f \circ g)(x + y) &= f(g(x + y)) \stackrel{*1}{=} f(g(x) + g(y)) \stackrel{*2}{=} \\ &= f(g(x)) + f(g(y)) = (T \circ g)(x) + (T \circ g)(y) \end{aligned}$$

*1 from additivity of g

*2 from additivity of f, because $g(x), g(y) \in \mathbb{R}^m$

1. scalar multiplication

$$(f \circ g)(ax) = f(g(ax)) \stackrel{*1}{=} f(a \cdot g(x)) \stackrel{*2}{=} a \cdot f(g(x)) = a \cdot (f \circ g)(x)$$

*1 from scalar multiplication of g

*2 from scalar multiplication of f, because $a \cdot g(x) \in \mathbb{R}^m$

composition of affine transformations remains an affine function

Definition of affine transformations :

1. $f(x) = Ax + a$ (for matrix A and vector a)

2. $g(x) = Bx + b$ (for matrix B and vector b)

proof:

$$\begin{aligned} (f \circ g)(x) &= f(g(x)) \stackrel{*1}{=} f(Bx + b) \stackrel{*2}{=} \\ &= A(Bx + b) + a = \underbrace{AB}_{matrix} \cdot x + \underbrace{(Ab + a)}_{vector} \end{aligned}$$

i got that for Matrix AB and vector Ab+a $(f \circ g)(x)$ is affine transformation because i can express it as multiplication Matrix multiplication - AB and vector addition $(Ab + a)$

*1 Additivity of g

*2 from g's affinity

Question 2

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

In the following iterative scheme we can see that θ^{n+1} is build from the previous weight θ^n and the derirative $\alpha \nabla f_{\theta^n}(x)$.

when $\alpha \nabla f_{\theta^n}(x)$ is equal to zero θ^{n+1} will be equal to θ^n :

$$\alpha \nabla f_{\theta^n}(x) = 0$$

\Downarrow

$$\theta^{n+1} = \theta^n - \underbrace{\alpha \nabla f_{\theta^n}(x)}_0 = \theta^n$$

$\alpha \nabla f_{\theta^n}(x)$ might be very small but not equal to zero, so the stopping condition should be $\|\alpha \nabla f_{\theta^n}(x)\| < \varepsilon$ for a small $\varepsilon > 0$. this will be achived when the optimization algorithm will reach to a statioary point.

of course we might stop before $\|\alpha \nabla f_{\theta^n}(x)\| < \varepsilon$ because we are updating θ^{n+1} over T itarations.

$$f(x + dx) = f(x) + \nabla f(x) \cdot dx + dx^T \cdot H(x) \cdot dx + O(\|dx\|^3),$$

$$H_{ij}(x) = \frac{\partial^2 f}{\partial x_i \partial x_j}(x)$$

B)

Recall that the Hessian matrix is a symmetric matrix (square matrix) containing all the second derivatives of the multivariate function. (matrix H in the second-order multivariate Taylor theorem).

1. Positive definite Hessian matrix: all eigenvalues are positive.
2. Negative definite Hessian matrix: all eigenvalues are negative.

for classifying a stationary point we need to look on the hessian matrix that is used in the second-order multivariate Taylor theorem.

1. local **minimum** condition: H is **positive definite**

if $\forall v \neq 0$ (vector) $v^T H v > 0$

1. local **maximum** condition: H is **negative definite**

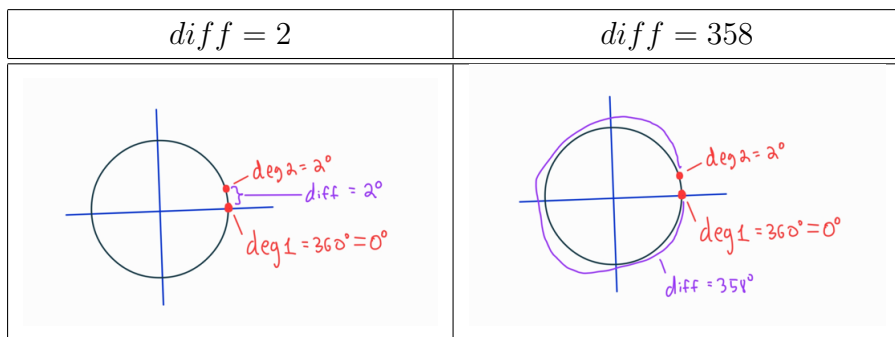
if $\forall v \in V$ (vector) $v^T H v < 0$

Question 3

Lets notice that in a circle, the difference between two degrees can be expresses with $degree \in [0, 180]$, because every difference between two angles that is greater than 180 degrees can be expressed with degree in range $[0, 180]$.

for example:

given $deg1 = 360$, $deg2 = 2$, the diff can be $358 \notin [0, 180]$ but using the same angles we can get diff of $2 \in [0, 180]$.



Pseudo code for differentiable loss function:

given trueDeg and predDeg, and using % as modulo.

1. compute $diff = abs(trueDeg - predDeg)$
2. $loss = min(360 - diff, diff)$
3. return loss

example with the following values: trueDeg=2 and predDeg=360

1. $diff = abs(trueDeg - predDeg) = abs(2 - 360) = 358$
2. $loss = min(360 - diff, diff) = min(2, 358) = 2$
3. return 2

Question 4 - Chain Rule

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

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

$$\begin{aligned}
&= \frac{\partial f}{\partial(x+y)} \cdot 1 + \frac{\partial f}{\partial(2x)} \cdot 2 + \frac{\partial f}{\partial(z)} \cdot 0 = \\
&= \frac{\partial f}{\partial(x+y)} + 2 \cdot \frac{\partial f}{\partial(2x)}
\end{aligned}$$

B) $f_1(f_2(\dots f_n(x)))$

chain rule - $\frac{\partial}{\partial x} f(g(x)) = f'(g(x)) \cdot g'(x)$

i will use the chain rule repeatedly on $f_1(f_2(\dots f_n(x)))$

$$\begin{aligned}
\frac{\partial}{\partial x} f_1(f_2(\dots f_n(x))) &= f'_1(f_2(\dots f_n(x))) \cdot f'_2(f_3(\dots f_n(x))) \cdot \dots \cdot f'_{n-1}(f_n(x)) \cdot f'_n(x) = \\
&= \frac{d}{dx} f_1(f_2(\dots f_n(x) \dots)) = \frac{df_1}{df_2} \frac{df_2}{df_3} \dots \frac{df_{n-1}}{df_n} \frac{df_n}{dx}
\end{aligned}$$

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

$$\begin{aligned}
&\frac{\partial}{\partial x} f_1(x, f_2(x, f_3(\dots f_{n-1}(x, f_n(x)))))) = \\
&= \frac{\partial f_1}{\partial x} + \frac{\partial f_1}{\partial f_2} \cdot \frac{\partial f_2}{\partial x} + \dots + \frac{\partial f_1}{\partial f_2} \cdot \frac{\partial f_2}{\partial f_3} \cdot \dots \cdot \frac{\partial f_{n-1}}{\partial f_n} \cdot \frac{\partial f_n}{\partial x}
\end{aligned}$$

D) $f(x + g(x + h(x)))$

$$\frac{\partial}{\partial x} f(x + g(x + h(x))) = 1 + \frac{\partial g}{\partial(x + h(x))} \cdot (1 + \frac{\partial h}{\partial x})$$