GÖTEBORGS
UNIVERSITET

INTRODUCTION TO DATA SCIENCE & AI

ASSIGNMENT 7: NEURAL NETWORK

Anh Thu Doan
Exchange student from CY Tech - France to GU CSE Department
thudoann45@gmail.com
15 Hours

Khushi Chitra Uday
Exchange student from CY Tech - France to GU CSE Department
guschikh@student.gu.se
14 Hours

1) Explain the data pre-processing highlighted in the notebook.

   We can see that the data consists of images with maximum pixels of 255 and a minimum value of 0. The data is converted to values between 0 and 1 when we divide the data by 255. We can also see that the elements are integer, so first, we convert the elements to float 32 to perform normalisation. The input features are normalised on the same order of magnitude to make the training more accessible.

2) A. How many layers does the network in the notebook have? How many neurons does each layer have? What activation functions and why are these appropriate for this application? What is the total number of parameters for the network? Why do the input and output layers have the dimensions they have?

   - There are four layers in the network.
   - There are 784 neurons in the First(Input) layer. There are 64 neurons in the First(Dense) layer. There are 64 neurons in the Second layer and ten neurons in the Third(Output/Dense) layer.
   - Two activation functions are used Relu(rectified linear unit) and softmax.
     Relu returns the element-wise maximum of (0, x), where x is the input tensor. Using this function does not activate the negative value of the neurons.
   - Total number of parameters for the network:
     - Input layer = 0
     - 1st Dense layer = 64*784+1*64 = 50240
     - 2nd Dense layer = 64*64+1*64 = 4160
     - 3rd Dense layer = 10*64+1*10

     So the total number of parameters for the network is 55,050.
   - Why do the input and output layers have the dimensions they have?
     The input shape is a whole picture with dimensions of 28 by 28 pixels. The flattened layer reduces this matrix to a 1D array of length 28*28, i.e. (784, 1). As there are ten classes, the output layer has ten neurons.

   B. What loss function is used to train the network? What is the loss function's functional form(a mathematical expression)? And how should we interpret it? Why is it appropriate for the problem at hand?

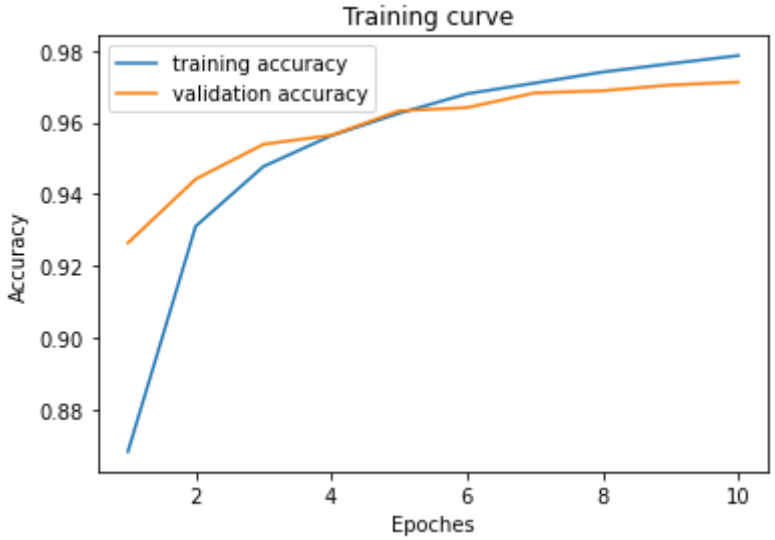
   - Categorical cross-entropy is the loss function used to train the network.
   - $Loss = \sum_{i=1}^{10} y_i \cdot log(\widehat{y_i})$

     where $\widehat{y_i}$ is the vector of predicted scalar values. y is the vector of zeros. 1 is showing the correct value.
   - By comparing the output layer, which represents the integer to which the image corresponds as a probability distribution, to the actual label as a one-hot vector. Since $y \cdot log(\widehat{y})$, it is only the elements where y = 1 (right

categorisation) are compared with the probability of the output layer. As a result, only the loss of the likelihood of the correct integer is assessed and minimized.

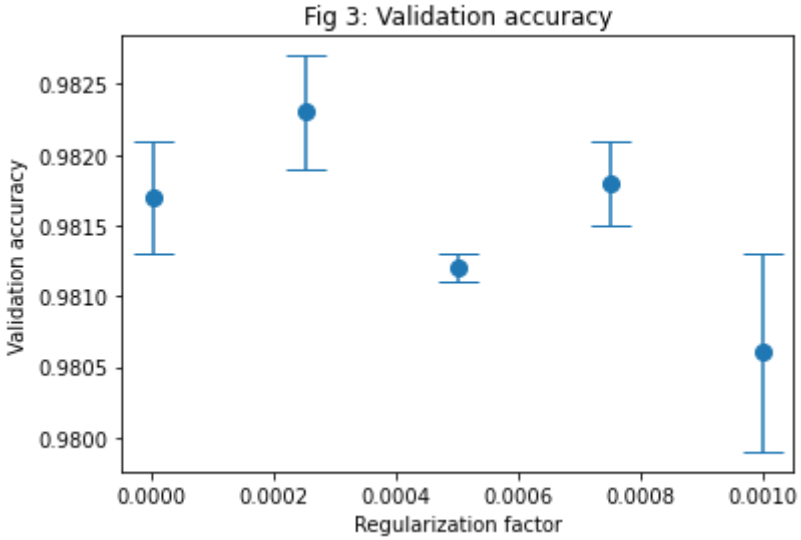
- This loss function is appropriate when softmax is used as the activation function and there are many output labels. This loss function is ideal for multi-class classification problems like this one. In other words, it is acceptable when there is one specific correct answer (class) and several incorrect answers - each digit can only belong to one class.

C.



Accuracy curve for learning rate 0.1 with 10 epochs.

D.



Maximal validation accuracy: 0.982699990272522.
Hilton validation accuracy: 0.9847.
Difference in validation accuracy from Hilton: 0.002000009727478047.

Since we don't know the no of epochs and the learning rate he used, we do not get the same accuracy. The learning rate decides how fast the accuracy converges, so if Hilton used another combination, it would differ from our results. He has also not stated what regularisation factor was used, which would result in him getting a higher validation accuracy.

3.

A.

a) First, we added a convolutional layer with 32 filters of size 3x3.
b) Then we added a MaxPooling layer with pool size (2,2) and stride (2,2) to downsize the input.
c) We added another convolutional layer with 64 filters of size 5x5.
d) Then we again added a MaxPooling layer with pool size (2,2) and default stride to further downsize the input.
e) We then added a flattened layer to decrease the dimensions.
f) We added a dense layer with 128 neurons.
g) We added another dense layer as the output layer with 10 neurons and activation function softmax to get the probability for each class."

B.

- Convolutional layers can be more effective than fully-connected layers in picture classification since they learn features from the input.
- A weight matrix is applied to sections of the input at a time in convolutional layers to build a feature map.
- A weight is applied to each input separately in fully connected layers. This makes completely connected layers broader, as there are no data assumptions.
- Because the same weight matrix is applied across the data, convolutional layers are well-suited for data that can be perceived as spatial.
- This suggests that some pattern is equally likely to exist at all data points. Convolutional layers function well since we know the data is images.

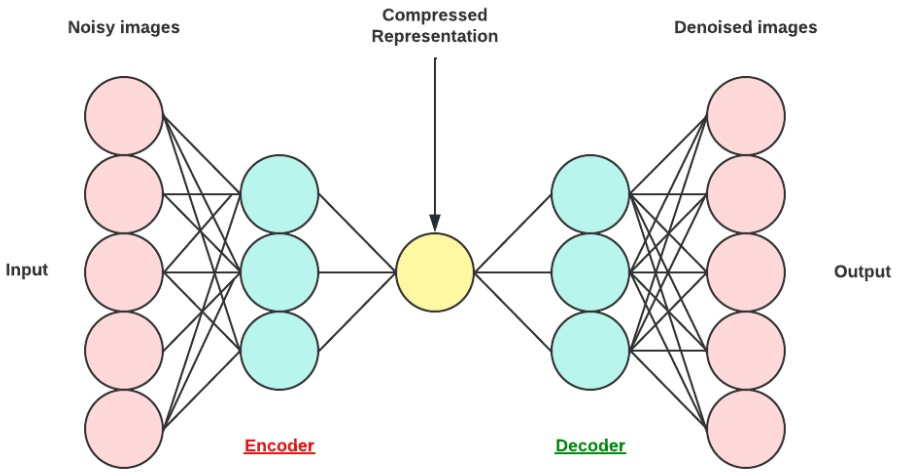
4.

A. Because x train is a 28x28 picture, we use x_train.reshape(-1,784) to convert it to a 1x784 vector, resulting in a flattened_x_train. We use the salt and pepper() method to "season" the vector with noise. The picture vector with noise is flattened_x_train seasoned as a result. The same is true for the x_test test data.

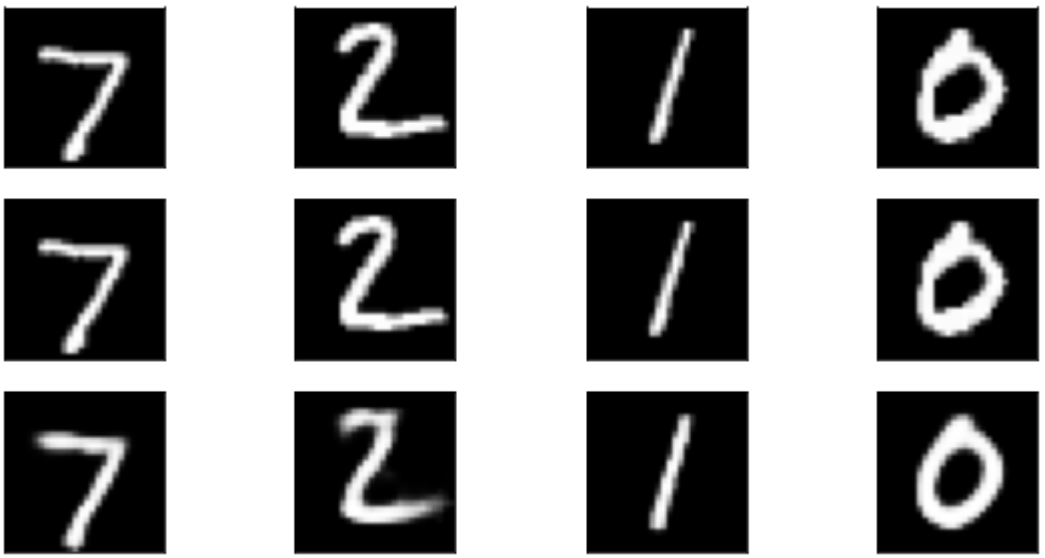
In terms of the model, it is an autoencoder. The model is made up of an encoder and a decoder. The encoder comprises two fully connected layers that each include ReLU activation functions. The first layer has 128 neurons, then mapped to a layer with

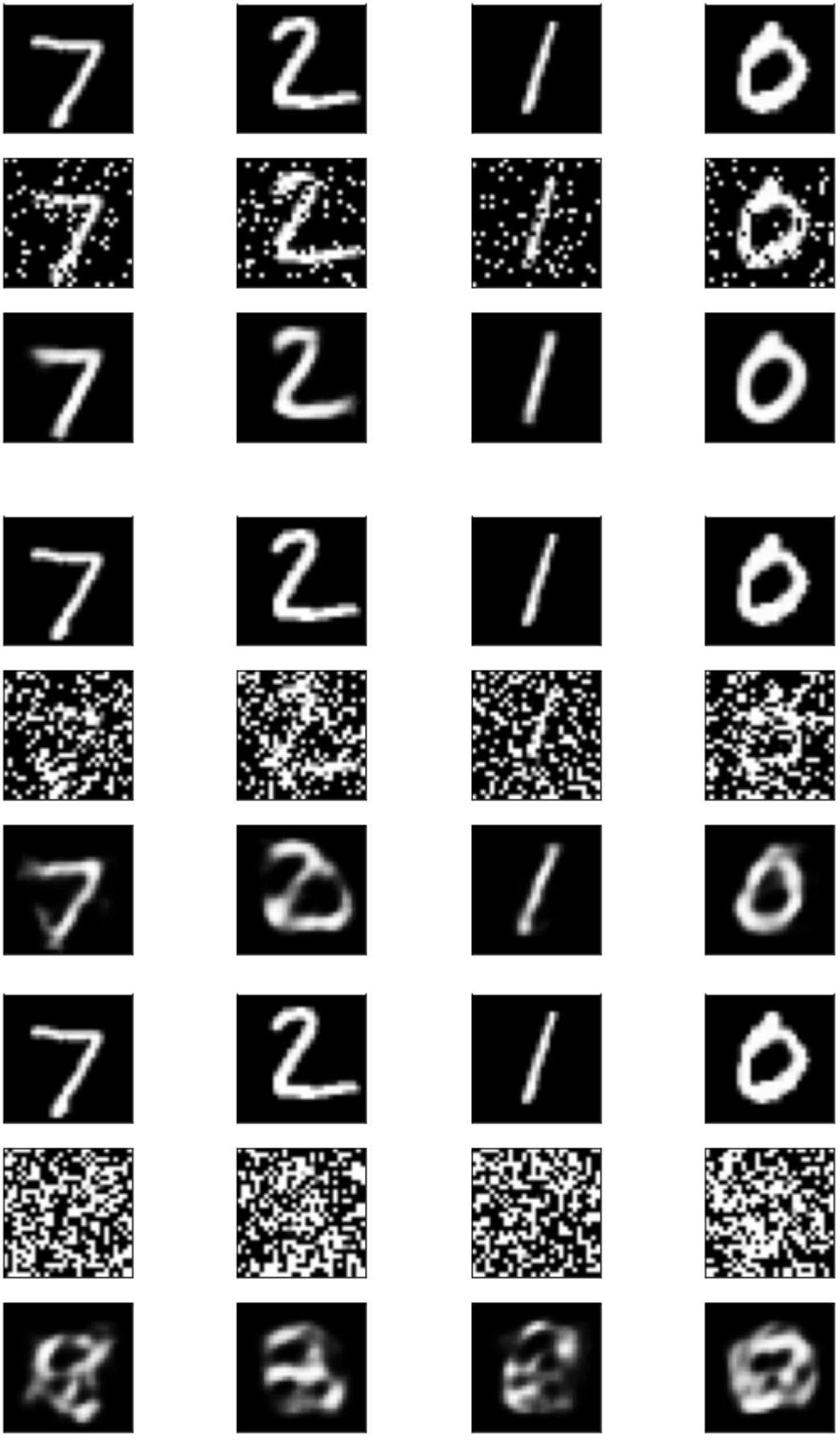
fewer neurons (96). The decoder layers map the encoder layer back to its original dimensions (28*28=784).

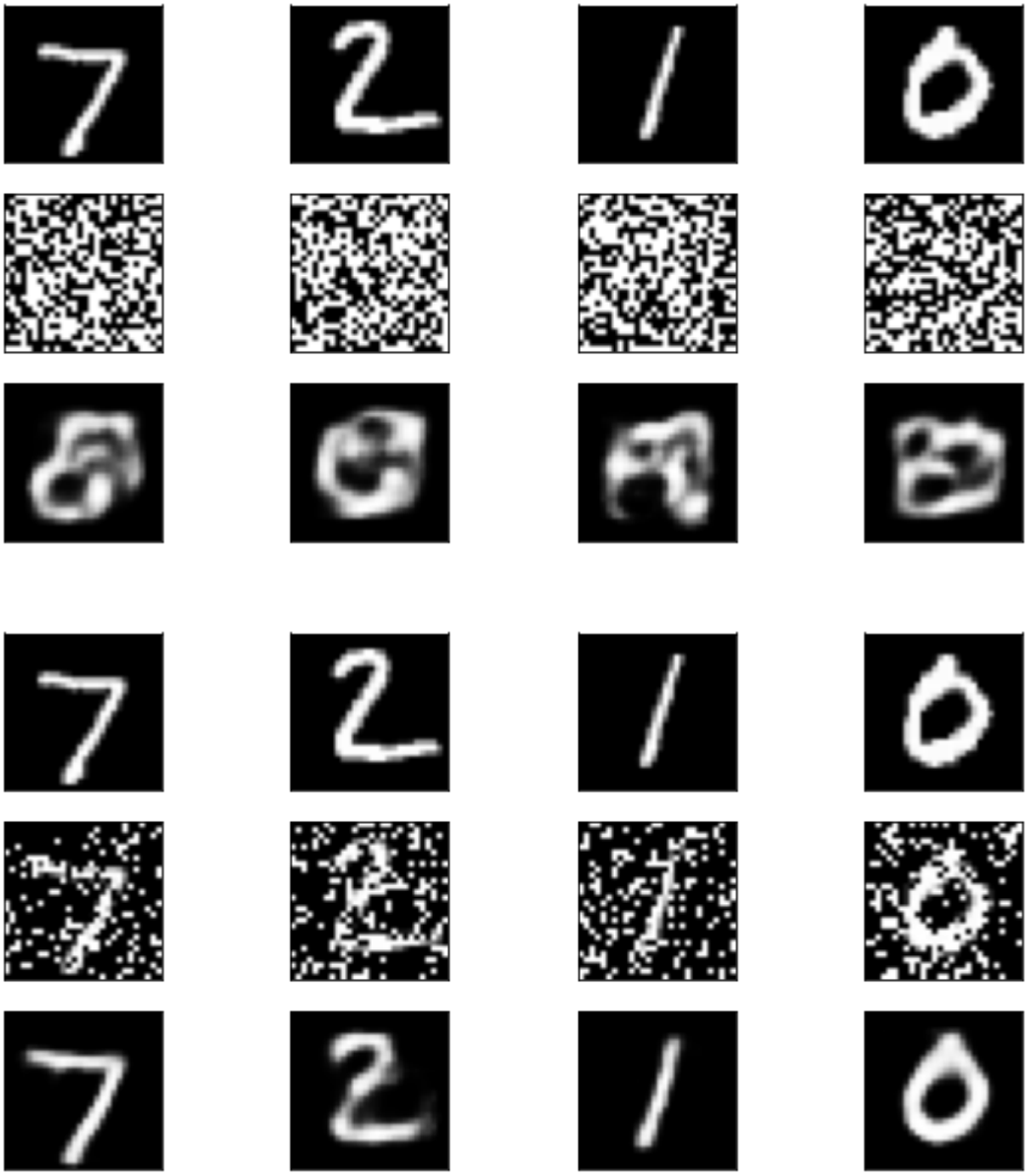
Explain the role of the loss function?
The binary cross-entropy loss computes the difference in cross-entropy between accurate and anticipated labels. When there are just two label classes, this is utilised.



B.







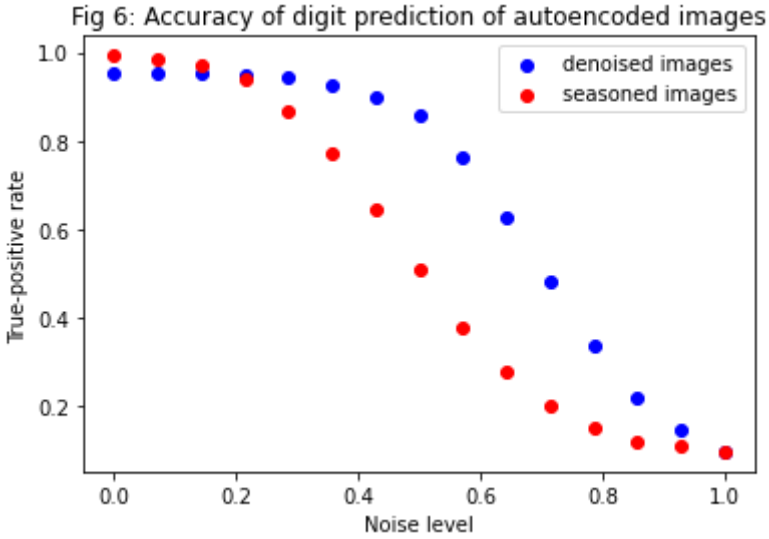
At what noise level does it become difficult to identify the digits?

Around noise level 0.6, it is difficult to identify the digits

At what noise level does the denoising stop working?

At 0.8 noise level.

C.



When the true positive rate (TPR) is plotted as a function of noise level, a function akin to an inverted shifted sigmoid curve is formed. When the autoencoder is used before evaluating the model from question 3, the TPR is relatively high until a noise level of 0.4 is attained. The function exhibits a significant drop at a noise level of 0.6. This is due to the high noise level, which, as described in question 4B, makes it difficult for even the human eye to determine which digit the image represents.

If the autoencoder is not employed and the images with extra noise are evaluated directly by the model, the TPR is significantly lower - especially in the 0.2 to 0.8 range. This demonstrates how autoencoders may be pretty helpful in removing noise and providing a more accurate evaluation of images.

D. The network is given both the original images x and their noisy variant x. The network attempts to reconstruct its output x' to be as close to the original image x as feasible. It learns how to denoise photos as a result.

Encoder and decoder networks are often trained together. The network is penalized by the loss function for producing output x' that differs from the original input x.

By doing so, the encoder learns to keep as much important information as possible within the constraints of the latent space while discarding unnecessary elements, such as noise. The decoder learns to rebuild the compressed latent information into a whole error-free input.

How to implement it?

We are implementing an autoencoder to denoise hand-written digits. The input is a 28x28 greyscale image, which generates a vector of 784 components.

The encoder network has 64 neurons in a single dense layer. As a result, the latent space will have a dimension of 64. Each neuron in the layer has a rectified units (ReLu) activation function that determines whether it should be activated ("fired") or not based on whether the neuron's input is essential for the autoencoder's prediction. The activation function also aids in normalising each neuron's output to a range between 1 and 0.

The decoder network is a single dense layer with 784 neurons, corresponding to a 28x28 greyscaled output image. A sigmoid activation function compares the encoder input versus the decoder output.