

# CS577 Assignment 1: Final report

Theo Guidroz  
Department of Computer Science  
Illinois Institute of Technology

February 12, 2021

## Abstract

This is a report for Assignment 1 of CS 577. The assignment served as an introduction to Keras and training basic models.

## 1 Problem Statement

### 1.1 Question 1

Getting used to models by modifying the parameters and choosing different input data on Playground. The goal of this problem was to minimize the test loss.

### 1.2 Question 2

Training a multi-classification model to read images and label them by airplane, automobile, or bird.

### 1.3 Question 3

Training a binary classification model to detect spam and non-spam emails.

### 1.4 Question 4

Training a regression model to predict violent crimes per population

## 2 Implementation details

### 2.1 Question 1

This problem was quite intuitive and was solved using a trial and error method. The parameters such as the learning rate, the features fed in, activation, and network architecture were modified to minimize the test loss. While the first three data type were easy to optimize, the spiral data had to be tweaked several times before training correctly. Trial and error worked great to solve this problem

### 2.2 Question 2

This problem was solved using the steps covered in Professor Agam's slides:

1. The data was downloaded from the Keras package. [1]
2. A validation set was created
3. The labels sets were converted using the `to_categorical` function

4. A network architecture model was created with the first layer having an input of (32,32,3) and the last layer having 3 outputs and softmax as activation.
5. The model was compiled using rmsprop as optimizer, categorical\_crossentropy as loss function and accuracy as metrics
6. The model was trained using the fit() function and the data was recorded. The graph of training loss, validation loss versus epochs was recorded to identify overfitting.
7. Based on the graphs and value recorded, the model was retrained with different parameters. A table of the data recorded is available in the next section.
8. The final model was evaluated by running the test dataset.

The implementation of this solution went smoothly and no major issues were encountered.

### 2.3 Question 3

This question was similar to the classifying movie reviews done in class so this problem was solved using similar approaches:

1. Create the load\_spam\_data() function to import the data from the website.
2. Prepare the data by subtracting the mean and dividing by the standard deviation for the emails to normalize the features. Convert the labels to float.
3. Create the validation set by dividing the training data into two sets. The ratio of validation set to training set is 500:3500
4. Build a network architecture with the first layer having an input of (57,) and the last layer having one output and using the sigmoid activation.
5. Compile the model using an initial learning rate of 0.001, binary cross-entropy as loss function, and accuracy as metrics.
6. Train the model and record the loss functions to plot the graph of loss function versus epochs. This graph is very useful to determine if the model is overfitting.
7. Modify the values several times to optimize the performance of the model. A table of the different parameters used can be found in the next section.
8. Train the final model and evaluate the test dataset.

This solution was more challenging to implement than the previous question because it required knowledge of the pandas' library. Fortunately, the latter was very well documented which made it relatively easy to implement.[2]

### 2.4 Question 4

This question was similar to the regression problem seen in class so the solution follows the same method:

1. Load the data using pandas to read the CSV file. Unfortunately, there was a lot of data missing in this dataset so to overcome this problem, the average of each column was calculated and the empty cells in the table were filled with the average of their respective column.
2. The data had to be normalized to be trained and any non-numeric column had to be dropped. The third column was dropped and the data were normalized by subtracting the mean and dividing by the standard deviation. The load\_crime\_data was then implemented to create the datasets.
3. The network architecture was built with the first layer having an input of (127,) and the output layer having only 1 node.

4. K-fold validation was implemented in this problem to overcome the limitations of available data. The model was then trained several times using different parameters to identify which one best optimizes the Mean Average Error of the model. A table of the different parameters can be found in the next section.
5. The model was then evaluated on the whole dataset and the MAE was recorded at each iteration. The mean was then calculated and used to evaluate the model.

The data was prepared using the pandas library. The knowledge learned in the previous example was helpful in this process.

## 3 Results and discussion

### 3.1 Question 1









Data type	Learning rate	Activation	Regularization	Regularization rate	Features fed in	Number of training epochs	Network architecture (neurons per hidden layers)	Training loss	Test loss	Output
	0.03	ReLU	None	0.1	x1,x2	125	3,3,2	0.001	0.002	
	0.3	Tanh	None	0.03	x1,x2	21	2,2	0.000	0.000	
	0.1	Linear	None	0.1	x1,x2	77	1	0.000	0.000	
	0.1	Tanh	None	0	X1, x2, sin(x1), sin(x2)	193	8,2	0.005	0.008	

Figure 1: Table of the final output and parameter used

The parameters used to optimize the different models were tabulated as seen above. Since this question was mainly to get a better understanding of network architecture, a trial and error method was used to optimize the lost functions. Overall, the test values recorded were excellent with all of them recording a test loss of less than 0.01.

### 3.2 Question 2

Iteration #	Hidden layer 1 units	Hidden layer 1 activation	Hidden layer 2 units	Hidden layer 2 activation	Hidden layer 3 units	Hidden layer 3 activation	Number of training epochs	Best Validation Loss	Perform best at epochs
1	64	relu	32	relu	32	relu	30	0.5373	7
2	64	relu	32	relu	-	-	30	0.5655	16
3	32	relu	32	relu	32	tanh	30	0.5584	5
4	16	relu	16	relu	32	relu	30	0.5502	4

Figure 2: Table of parameters used to train the model

The different parameters used to optimize the model were recorded in the table above. This table was useful to determine the best parameters to evaluate the model.

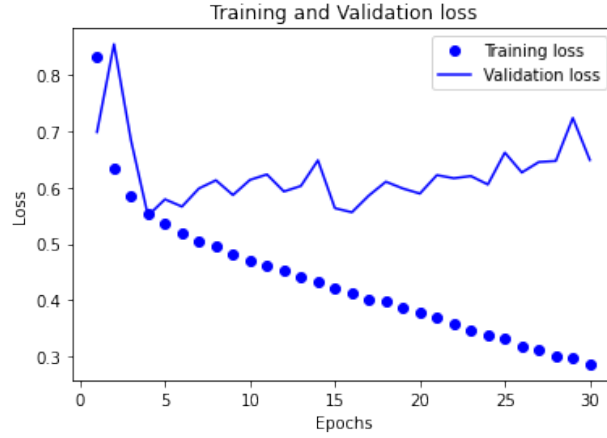


Figure 3: Graph showing the model overfitting around epochs 4

A similar graph was printed after each training. It helped visualize if overfitting occurred. This information was helpful to determine if the model had to be trained on more complex or simpler network architecture. On this graph, it can be seen that that overfitting occurs after epochs 4 since the training loss keeps decreasing after but the validation loss increases. The final accuracy of this model is 0.7847 and a loss of 0.5476

### 3.3 Question 3

Iteration #	Hidden layer 1 units	Hidden layer 1 activation	Hidden layer 2 units	Hidden layer 2 activation	Hidden layer 3 units	Hidden layer 3 activation	Number of training epochs	Best Validation Loss	Perform best at epochs
1	16	relu	16	relu	-	-	40	0.3885	7
2	8	relu	8	tanh	-	-	40	0.3169	14
3	16	relu	-	-	-	-	40	0.3816	10
4	8	relu	8	relu	-	-	40	0.3812	8

Figure 4: Table of parameters used to train the model

The different parameters used to optimize the model were recorded in the table above. This table was useful to determine the best parameters to evaluate the model. The graph shows that the selected model overfits

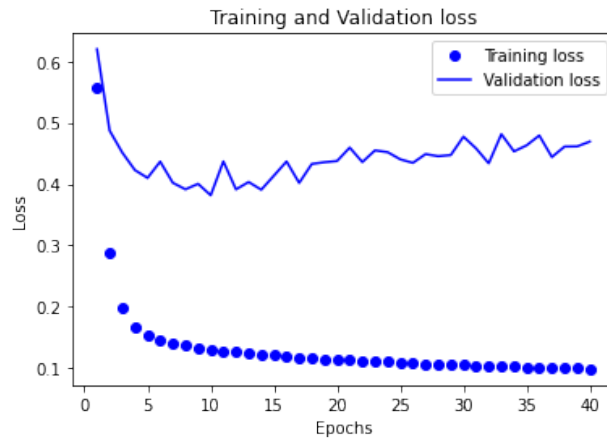


Figure 5: Graph showing the model overfitting around epochs 14

after epochs 8 since the validation loss increases from there.

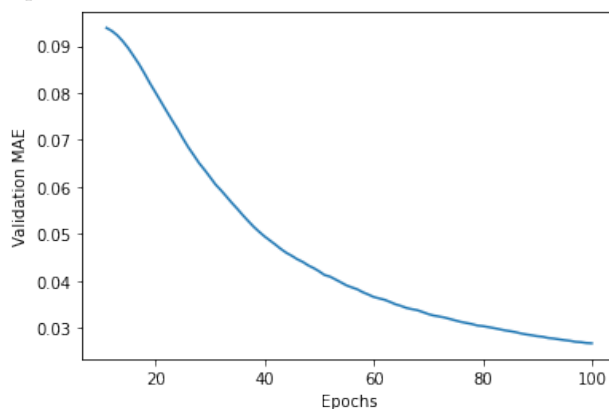
Overall, the model had a relatively high accuracy (0.8669) and a relatively low loss function (0.4538). These are satisfactory results for this model

### 3.4 Question 4

Iteration #	Hidden layer 1 units	Hidden layer 1 activation	Hidden layer 2 units	Hidden layer 2 activation	Hidden layer 3 units	Hidden layer 3 activation	Number of training epochs	Best Validation MAE
1	64	relu	64	relu	-	-	100	0.0699
2	32	relu	32	relu	32	relu	100	0.0438
3	64	relu	32	relu	32	relu	100	0.0518
4	16	relu	32	relu	16	relu	100	0.0412

Figure 6: Table of parameters used to train the model

The different parameters used to optimize the model were recorded in the table above. This table was useful to determine the best parameters to evaluate the model.



A similar graph was printed after each training. It helped visualize if overfitting occurred. This information was helpful to determine if the model had to be trained on more complex or simpler network architecture. The only issue about using this graph to identify overfitting is that the model is also trained on the validation set since we are using k-fold. Therefore, we might not be able to identify overfitting even if it occurs. Overall, the model performed well on the limited data provided to train it. The mean MAE was 0.0162 which is relatively close to 0. This means that the model is well optimized.

## References

- [1] *Keras documentation: Developer guides*. URL: <https://keras.io/guides/>.
- [2] *pandas documentation*. URL: <https://pandas.pydata.org/pandas-docs/stable>.