REYNA REYES Diego & Ryan VAN GREUNEN
AIC-4201C
Deep Learning

Assignment: Building a deep learning framework

January 15th, 2023

ABSTRACT

The aim of this project is to learn how to build a deep learning framework from scratch. This will be done in the form of a Jupyter notebook.

QUESTIONS

Quiz 1

The first step of this process is to compute the softmax of the rows of a matrix. When considering a layer of a neural network we can conclude that it is made up of two operations. Firstly, a scaler product using parameters (weights and biases) and secondly an activation function that does not include parameters. The softmax function is one of a few different types of activation functions. Softmax activation belongs to the non-separable type of activation functions which means that an operation is applied to a vector as a whole. The softmax function manipulates a vector so that the total of the vector sums up to one and is a generalisation of the sigmoid function which considers more than one dimension. The operation of the function can be seen in the equation below.

$$g(z) = \left[\frac{e^{z_1}}{e^{z_1}+\cdots+e^{z_K}} \quad \frac{e^{z_2}}{e^{z_1}+\cdots+e^{z_K}} \quad \vdots \quad \frac{e^{z_K}}{e^{z_1}+\cdots+e^{z_K}} \right]$$

For the function in the Jupyter notebook, we firstly calculate the exponential of z. Secondly, we sum up the exponentials of the elements of z and thirdly we divide the two to arrive at the softmax of the rows of z.

Quiz 2

The second step from the Jupyter notebook is to compute the layer output from the layer input and the layer parameters. This is done via a function called dense_layer. The output of the

function is a matrix multiplication between the inputs and weights with the addition of the bias. This makes up the linear part of the layer. The non-linear part of the layer concerns the choice of the activation function type. Which can either be softmax or relu. The softmax function was defined in Quiz 1. The relu function is defined here and shows that the maximum is taken if the value is positive and set to zero if it is negative.

Quiz 3

In Quiz 3 we move from just examining an individual layer to considering a neural network made up of multiple layers. We created an initialise parameter's function that randomly initialises the layer parameters $W^{[1]}$, $b^{[1]}$ and $W^{[2]}$, $b^{[2]}$. The inputs to the function are the size of the network input, the size of the hidden layer (1st layer) and the size of the network output (2nd layer). $W^{[1]}$ is found using a random initialisation of a matrix with the number of rows taken as the size of the network input, the number of columns taken as the size of the hidden layer multiplied with the square root of the elementwise division of 2 over the size of the network input. $W^{[2]}$ is found using a random initialisation of a matrix with the number of rows taken as the size of the hidden layer, the number of columns taken as the size of the network output multiplied with the square root of the elementwise division of 2 over the size of the hidden layer. $b^{[1]}$ is found by creating an array of zeros with a row of size one and columns of the size of the hidden layer. $b^{[2]}$ is found by creating an array of zeros with a row of size one and columns of the size of the network output. The function returns the weights and the bias for both layers.

Quiz 4

In Quiz 4 the goal is to create a function that will compute the output of a two-layer neural network. This makes use of the dense_layer function from Quiz 2. For the case of this project, the first layer was chosen as relu and the second as softmax. A0 is set as the input to the network. The output of the first layer (A1) is defined by the inputs, the first weight and first bias with a selection of the relu activation function. The output of the second layer (A2) which is the output of the network is defined by the output of the first layer, the second weight and bias and the selection of the softmax activation function. The function then returns the output of the Neural network.

Quiz 5

In Quiz 5 the aim is to encode a vector of labels (from 0 to K-1) into a matrix of one-hot rows. The size of the elements in labels is simply labels.size. C is the maximum value found in labels with the addition of one. A zero matrix is then created with the number of rows being the number of elements in labels and the number of columns being the max value in labels plus one. Two

indexes can then be chosen to declare which element in a row needs to be encoded with one. Index 1 is chosen by the range of the number of elements in labels and Index 2 is chosen by the label that will have the one hot encoded value.

Quiz 6

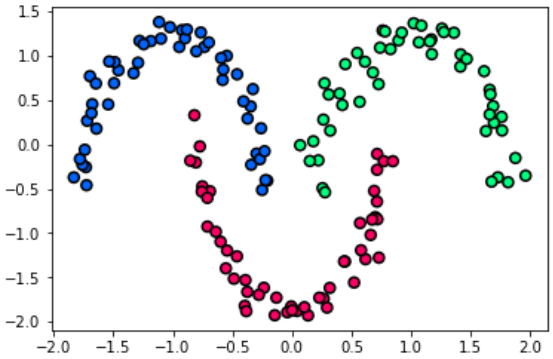
In Quiz 6 the goal is to compute the cross entropy between the output of the neural network and the true targets. Cross entropy is a loss function that is used with classification problems. Incorporating it into the function we say that the output is equal to the sum of the multiplication between the true target and the logarithm of the output divided by the number of samples in the matrices.

Quiz 7

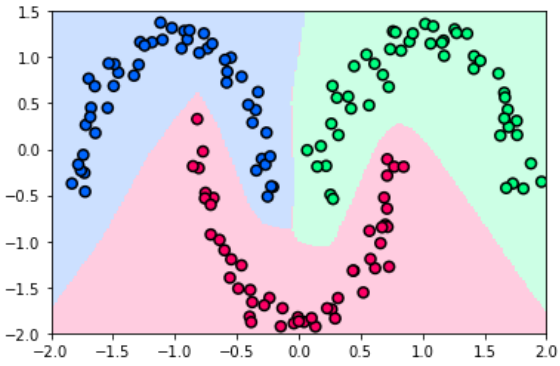
In Quiz 7 we move on to calculating the training loss. Here we need to define the cost function of the optimisation problem. The training loss is used to determine how well the model fits the training data. We start off by creating a prediction of the outputs using our two layer network function with the inputs and parameters. We can then use the cross entropy function that we created to determine the training loss between the predictions and the true outputs.

Quiz 8

In Quiz 8 we run the network on the inputs and return the most probable class for each of them. Here the probability that a sample belongs to a certain class (one of the three) will be determined. Every row (axis=1) will then be examined for the highest probability that a sample belongs to a specific class. This class is then returned. In the figure below we can see the original samples from the moon dataset.



The figure below displays the results of the implementation of Quiz 1 to Quiz 8.



Here we can see that the trained network correctly classified the training points using the predicted classes of either red, green or blue.

Quiz 9

In order to reduce the training time of our network, we implemented a modification to gradient descent called stochastic gradient descent. This variation finds the gradient using only a fraction (batch) of the training data. To divide the training data in batches, we shuffle them and then divide them using a batch size, it is possible for the last batch to be smaller depending on the number of inputs and the batch size.

In order to implement this into our framework, we had to implement a class that'll keep up the batches to be used on training, and that'll generate the cost function, this function being:

$$J(\theta) = \frac{1}{N}\sum_{n=1}^{N} - y_n^T log\,(f_\theta(x_n))$$

The index where every batch will start, and end were defined by:

$$idx_{begin}, idx_{end} = n * batchSize, (n+1) * batchSize$$

For every iteration, our class will do the following:

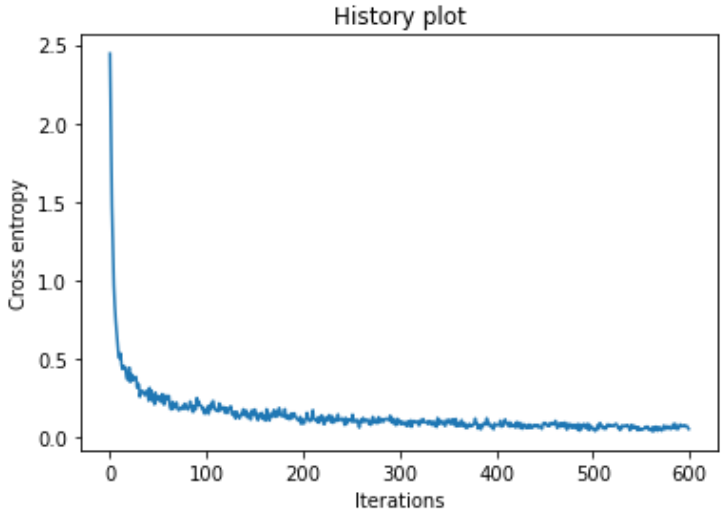
1. Extract a batch from the training data
2. Make a prediction using the training data
3. Calculate the cost with the cross entropy function
4. Return the cost for the given parameters

After creating our training class, we made a function to compute the parameters, using stochastic training. For this purpose, we used the Adam algorithm. This algorithm stands for Adaptative Moment Estimation. This algorithm keeps independent learning rates for different parameters, making it easier since there's no need for second gradients. For this reason, this implementation is ideal, since in future questions we'll have a big number of parameters to tune.

To use this function, we need to define certain parameters, like the batch size for our training class, and the learning rate and number of epochs.

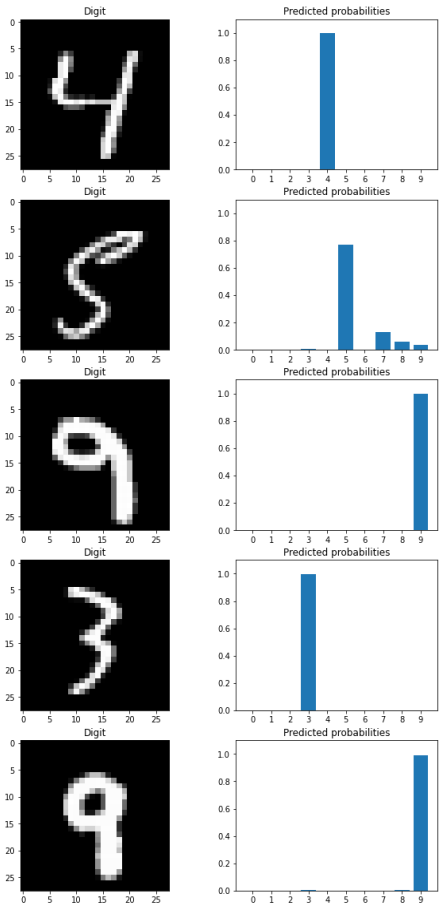
Having run our function to train our 2 layer network on the MNIST data set and with the parameters shown on the next table we found the result shown in the next graph:

Parameter	Value
Size of hidden layer (n_hidden)	50
Learning rate (lr)	0.01
Epochs	10
Batch size (batch_size)	1000



As we can see, thanks to our stochastic training, our cost function (cross entropy) has a tendency towards 0.

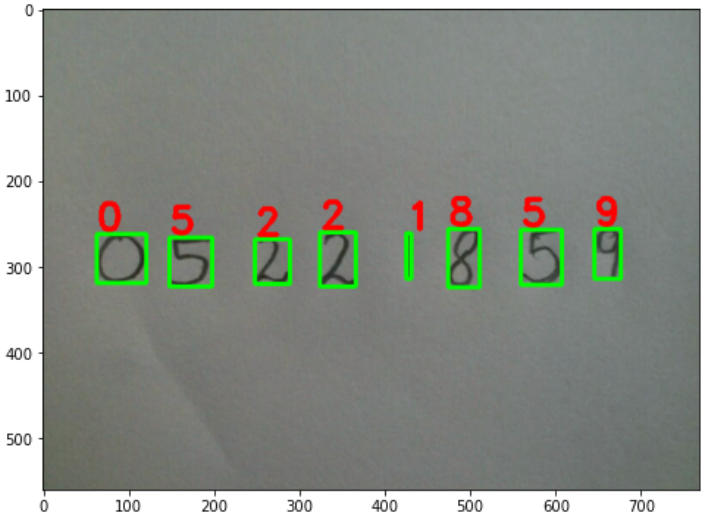
After this, we can test the performance of our trained parameters on our test data:

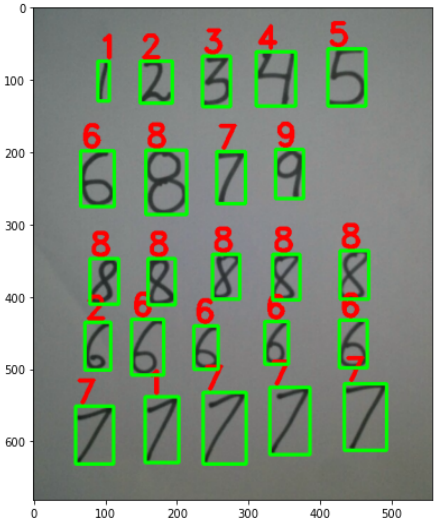


As we can see, our 2 layer network can identify with high accuracy, since its accuracy score is 97.11%.

Quiz 10

After training our parameters we can test using new test data. We'll test using an image of handwritten numbers, and we'll use OpenCV to identify the written numbers and prepare them for the expected input of our neural network. We'll apply our neural network to the extracted images, and finally, draw our prediction in the original image so we can clearly compare if our algorithm classifies them correctly. The results we got are:





As we can see, our neural network has trouble identifying certain characters, and this is given by it only being a 2 layer network, something we'll try to improve in our next question.

Quiz 11

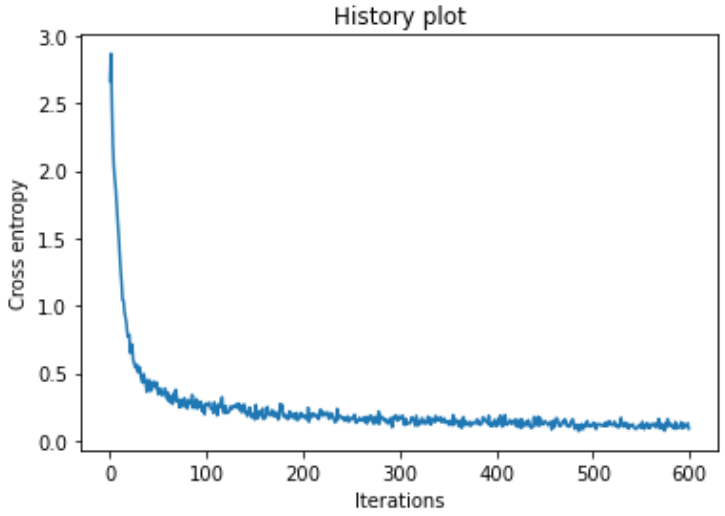
In this question we modified our previous code so it can run a multiple layer network. To achieve that goal, we modified the following functions:

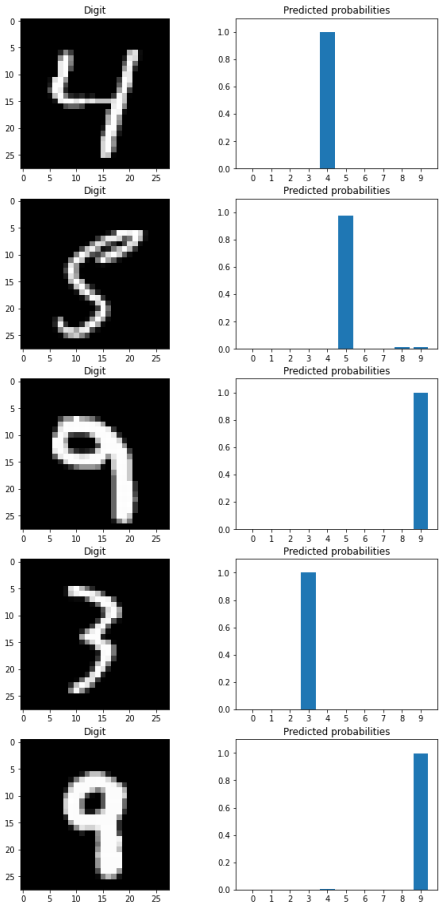
Function/Class	Modifications
dense_layer	Added new activation functions:

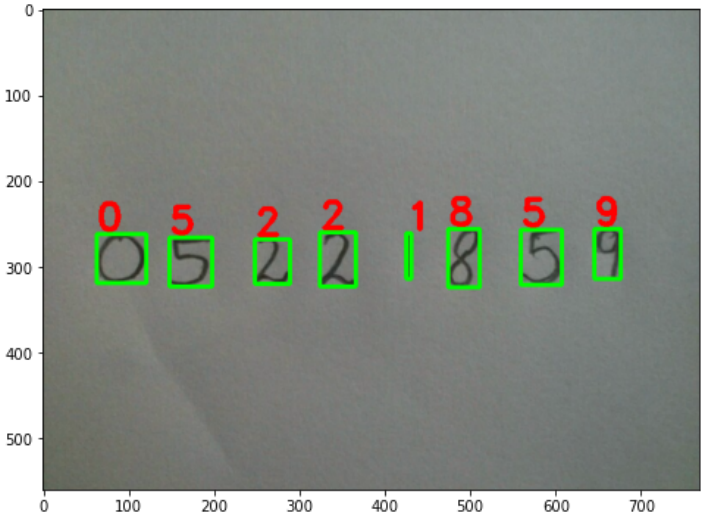
	$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$ $$leaky(x) = max(\lambda x, x)$$ $$elu(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$$ $$selu(x) = \begin{cases} \lambda x, & x \geq 0 \\ \lambda\alpha(e^x - 1), & x < 0 \end{cases}$$
dropout	Implemented a function to do inverted dropout for one layer.
intialize_parameters	Adapted so initialises as many w and b arrays as hidden layers.
multilayer_network	Adapted the input parameter to be an array containing the size of every hidden layer.
StochasticOptimizationProblem	Changed it so the predictions is made using the multilayer_network
stochastic_training	There was no need to change this one since the inputs for initialise parameters only changed data type.
digit_recognition	Changed it so it receives the network prediction function as a parameter

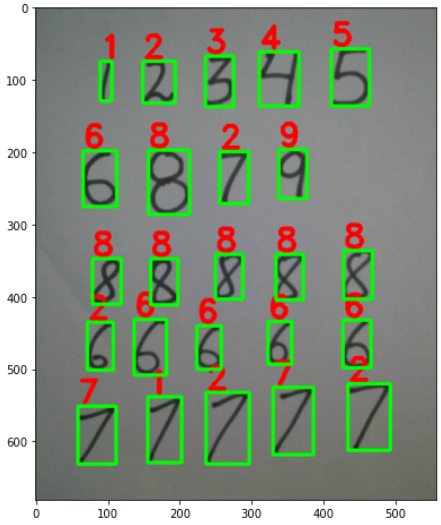
After modifying these functions, we found the next results, when applying the same tests as in questions 9 and 10, using the next parameters:

Parameter	Value
Size of hidden layers (n_hidden)	[370, 115, 50]
Learning rate (lr)	0.01
Epochs	10
Batch size (batch_size)	1000


History plot
Cross entropy
Iterations
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
100
200
300
400
500
600


Digit
Predicted probabilities





As we can see, our two layer network still struggles with certain handwritten numbers, but in general, we got an improved accuracy of 97.51%

Quiz 12

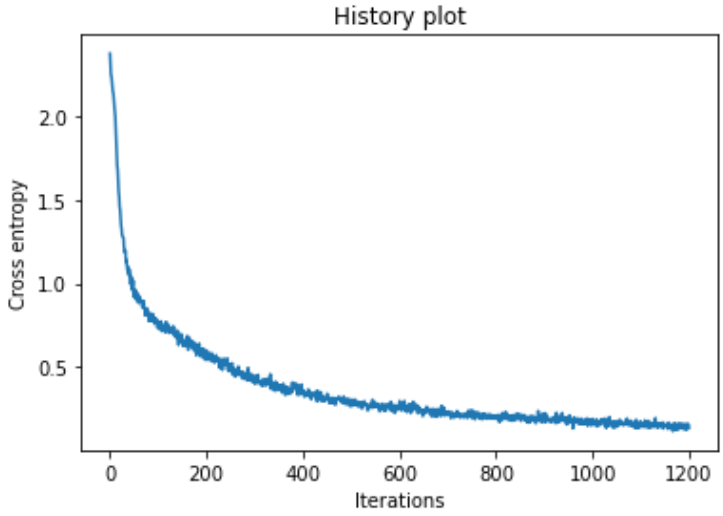
Finally, we implemented a LeNet-5 convolutional network, to make this possible we implemented and modified the next functions:

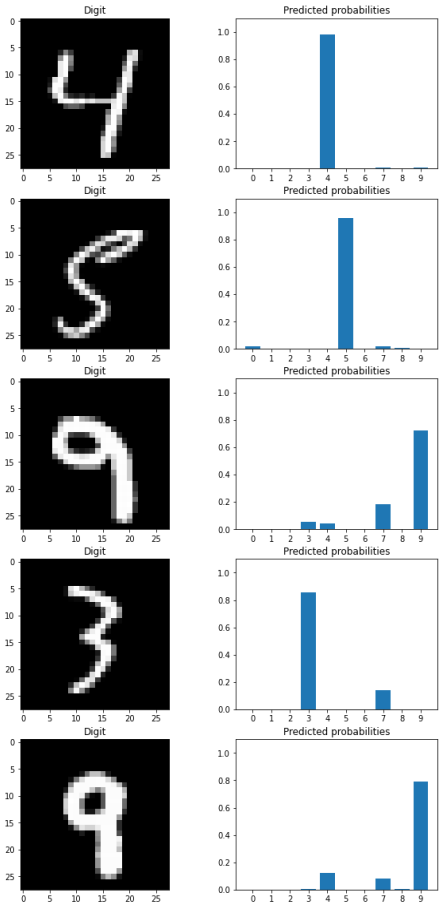
Function/Class	Parameter
convolution_layer	It applies a convolution between the input and a kernel, then adds a bias and finally, it applies

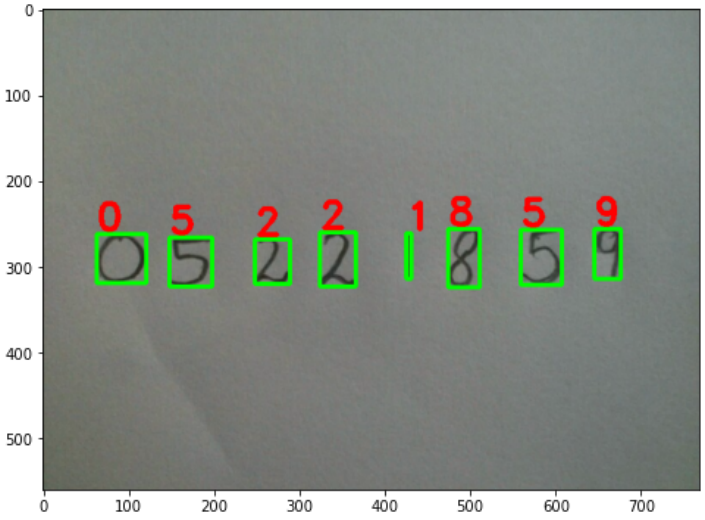
	a RELU activation
pooling_layer	Applies a maximum pooling to an input, using a stride of 2 and a window size of 2.
flatten_layer	Flattens the input so it's output is (1×N)
initialize_parameters_lenet	Initialises the parameters for the convolutional layer (kernels and bias) and the dense layer (weights and bias)
convolutional_network	Applies the LeNet-5 architecture using the following order of layers: Convolution layer Pooling layer Convolution layer Pooling layer Flatten layer Fully Connect tanh activation Fully Connect tanh activation Fully Connect softmax activation
StochasticOptimizationProblem	Changed it so it uses the convolutional network for the prediction
stochastic_training	Modified so it receives the parameters needed for initialize_parameters_lenet

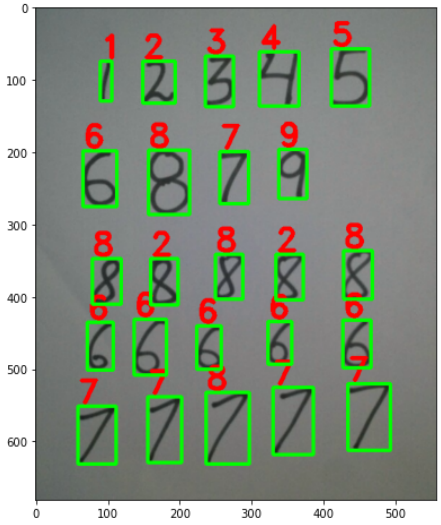
After modifying these functions, we found the next results, when applying the same tests as in questions 9 and 10, using the next parameters:

Parameter	Value
Learning rate (lr)	0.001
Epochs	50
Batch size (batch_size)	2500


History plot
Cross entropy
Iterations
2.0
1.5
1.0
0.5
0
200
400
600
800
1000
1200


Digit
Predicted probabilities





As we can see, our convolutional network still struggles with certain handwritten numbers, but they're different examples to the ones our previous networks struggled, it is assumed that with more training epochs the accuracy will improve, since this configuration obtained a 94.80% accuracy, but given the computing limitation of the computer where the algorithm was tested, it is impossible for us to train this algorithm more, given the number of parameters (44,426) that need to be trained.

References

Alpha (no date) *Exponential linear unit (ELU) layer - MATLAB - MathWorks France*. Available at: https://fr.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.elulayer.html (Accessed: January 15, 2023).

Brownlee, J. (2021) *Gentle introduction to the adam optimization algorithm for deep learning*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (Accessed: January 15, 2023).

Brownlee, J. (2021) *How to choose an activation function for deep learning*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/#:~:text=The%20Tanh%20activation%20function%20is,x%20%2B%20e%5E-x) (Accessed: January 15, 2023).

Papers with code - selu explained (no date) *SELU Explained | Papers With Code*. Available at: https://paperswithcode.com/method/selu#:~:text=Scaled%20Exponential%20Linear%20Unit&text=Scaled%20Exponential%20Linear%20Units%2C%20or,Source%3A%20Self-Normalizing%20Neural%20Networks (Accessed: January 15, 2023).

Team, G.L. (2022) *What is rectified Linear Unit (relu)?: Introduction to relu activation function*, *Great Learning Blog: Free Resources what Matters to shape your Career!* Available at: https://www.mygreatlearning.com/blog/relu-activation-function/#:~:text=Leaky%20ReLU%20activation%20function,-Leaky%20ReLU%20function&text=f(x)%3Dmax(0.01,for%20negative%20values%20as%20well. (Accessed: January 15, 2023).