CS577
HW#1 Report

Khalid Saifullah
ID # A20423546
Date: 16th Feb, 2021

PART-1 : QUESTIONS

Answer to the Question No. 1

Training dataset – The sample of data used to fit the model. This is the actual dataset that we use to train the model (weights and biases in the case of a Neural Network). The model sees and learns from this data.

Validation dataset – the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters (such as optimal number of hidden units). The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

Testing dataset – the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The test dataset provides the gold standard used to evaluate the model. It is used only once a model is completely trained ~~used~~ (using the train and validation sets).

Answer to the Question No. 2

Data – defining the training data (input tensors and target tensors).

Model – designing the network of layers for mapping the input and output to targets.

Learning process – defining the loss functions to be minimized. We also need to specify the optimizer to use with learning rates and other hyperparameters of the optimizer.

Fitting – this is the training step of the neural network. Here, we need to define the number of epochs for which we need to train the neural network.

Answer to the Question No. 3

The number of units is one of the most basic parameters of a dense layer. In a dense layer, units are interconnected to each other. ~~and~~ More units means more complex network. However, units at the output layer is selected according to the output demand. ~~In Keras, once the first layer's unit number is set selected, it~~ The unit parameter actually dictates the size of the weight matrix and bias vector (the bias vector will be the same size, but the weight matrix will be calculated based on the size of the input data so that the dot product will produce data that is of output size, units).

The choice of activation function is most important for the output layer as it will define the format that predictions will take. For instance, ~~below~~ some common predictive modeling problem types and the structure and standard activation function that you can use in the ~~at~~ output layer are –

- Regression: Linear activation function or 'linear' and the number of neurons matching the number of outputs.
- Binary Classification (2 class): Logistic activation function or 'sigmoid' and one neuron the output layer.

- Multiclass Classification (>2 class): Softmax activation function or 'softmax' and one output neuron per class value, assuming a one-hot encoded output pattern.

Answer to the Question No. 4

Optimizer: adjusts the parameters for better performance.
Loss: shows the training and validation loss for model viability comparaison and how good the model fits.
Metrics: shows model performance.

Loss function is used to optimize the model and it is minimized by the optimizer whereas metrics just shows the model performance.

Answer to the Question No. 5

Input - this is the training data
Output - this is the labelled training data (classification).
Batch size - Instead of the whole dataset, a chunk of it is provided at a ~~size~~ time.
Epoch - this is the number of iterations
Validation data - this is used to validate the model in the training phase.

Answer to the Question No. 7

Overfitting: the model is too closely fit to the training dataset e.g. few examples but complex network.

Underfitting: the model neither fits the training nor the validation dataset.

Answer to the Question No. 8

Layers: In general, adding more layers gives better accuracy but makes the system complex at the same time so for smaller dataset lower number of layers performs better.

Units per layer: The operation becomes more precise but takes more time as well if units per layer is increased.

Activation functions: Various functions can be used according to different output results.

Loss: Loss function needs to be minimized in order to make the model efficient.

Answer to the Question No. 9

For a function x, the logistic function $h(x) > 0$ and $h(x) < 0$ will give binary results and $h(x) = 0$ will give a probability of 50%.

Answer to the Question No. 10

Labels are converted into integers ~~into~~ then into binary vectors where all expect the index of the integer is zero.

Answer to the Question No. 11

Softmax is a mathematical function that converts a vector of ~~w~~ ~~ve~~ probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. A neural network model requires an activation function in the output layer of the model to make the prediction.

The Softmax function is used as the activation function in the output layer of neural network models that predicts a multinomial probability distribution. That is, softmax is used as the activation function for multi-class classification problems where class membership is required on more than two class labels and the values sum to 1.

Any time we wish to represent a probability distribution over a discrete variable with n possible values, we may use the softmax function.

Answer to the Question No. 12

So, generally there are two common loss functions which we can use after the output layer, though there are dozen more offered by keras.

If your targets are one-hot encoded, use categorical crossentropy.

→ Examples of one-hot encodings

- [1, 0, 0]
- [0, 1, 0]
- [0, 0, 1]

But if your targets are integers, use sparse categorical crossentropy.

→ Examples of integer encodings. (for the sake of completion):

- [1]
- [2]
- [3]

The usage entirely depends on how you load your dataset. One advantage of using sparse categorical cross entropy # is it saves time in memory as well as computation because it simply uses a single integer for a class, rather than a whole vector.

We use categorical cross entropy when we have few number of output classes generally 3-10 classes. However, we use sparse categorical cross entropy loss function when labels are mutually ~~exp~~ exclusive of each other that is when each sample will belong only to one class, when number of classes are very large., this can speed up the execution and save lot of memory by avoiding lots of logs and sum over zero values, which can sometimes lead to loss becoming NANs after some point during training.

Answer to the Question No. 13

Accuracy of a random classifier (random guess) is $1/k$, where k is the number of classes in the dataset. So, the accuracy of a random classifier having 5 classes is 0.2.

Answer to the Question No. 14

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here, X_{max} and X_{min} are the maximum and the minimum values of the feature respectively.

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

$$X' = \frac{X - \mu}{\sigma}$$

μ is the mean of the feature values and σ is the standard deviation of the feature values.

Machine learning algorithms like linear regression, logistic regression, neural networks etc. that use gradient descent as an optimization technique require data to be scaled. To ensure that gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features, we scale the data before feeding it to the model. Having features on a similar scale can help the gradient descent converge more quickly towards the minima.

Distance algorithms like KNN, K-means and SVM are most affected by the range of features. This is because behind the scenes they are using distances between data points to determine their similarity. Since both the features have different scales, there is a chance that higher weightage is given to features with higher magnitude. This will impact the performance of the machine learning algorithm and obviously, we do not want our algorithm to be biased towards one feature. Therefore, we scale our data before employing a distance based algorithm so that all the features contribute equally ~~result~~ to the result.

Answer to the Question No. 15

There are ~~three~~ metrics which are generally used for evaluation of Regression problems (like Linear Regression, Decision Tree Regression, Random Forest Regression etc.).

Mean Absolute Error (MAE): This measures the absolute average distance between the real data and the predicted data, but it fails to punish the errors in prediction.

Mean Square Error (MSE): This measures the squared average distance between the real data and the predicted data. Here, large errors are well noted (better than MAE). But the disadvantage is that it also squares up the units of data as well. So evaluation with different units is not at all justified.

Answer to the Question No. 16

Cross validation is a resampling procedure used to evaluate the machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. The general procedure is as follows:

1) Shuffle the dataset randomly
2) Split the dataset into k groups
3) For each unique group:
   - Take the group as a hold out or test dataset
   - Take the remaining groups as a training dataset
   - Fit a model on the training set and evaluate it on the test set
   - Retain the evaluation score and discard the model
4) Summarize the skill of the model using the sample of model evaluation scores.

~~Cross validation is a good technique to test a model on its predictive performance.~~ While Cross validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

Answer to the Question No. 17

Each observation in the data sample is assigned to an individual group and stays in that group for the during of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times. Compute the average of the k recorded errors. This is called the cross-validation error serving as the performance metric for the model. The models are then discarded after they are evaluated as they have served their purpose. The final model is then trained on the whole training set, ready to test the performance of the model on the test dataset. Or, the final model can be trained on all the data including the training, validation and testing data.

Answer to the Question No. 6

Firstly, all the words or text strings will be mapped to integers, making a dictionary. Each example can then have a vector of size $1 \times m$, where m is the total number of unique words. In the vector, all the values will be 0 except for the words which are 1 when present in that particular example. This is the binary feature vector.

PART - 2: PROGRAMMING
CS577
HW#1 Report

Khalid Saifullah
ID# A20423546
Date: 14th Feb, 2021

Task 1

	Circle	Gaussian	Exclusive OR	Spiral
Learning Rate	0.03	0.1	0.03	0.03
Activation	ReLU	Linear	Tanh	Tanh
Regularization	None	None	None	L2
Regularization rate	0	0	0	0
Problem type	Classification	Classification	Classification	Classification
Features used	X1^2, X2^2	X1, X2	X1X2	X1, X2, X1^2, X2^2
Number of training epochs	747	105	200	564
Network architecture	1 hidden layer with 3 units	No hidden layers	No hidden layers	2 hidden layers with 8 units at first and then 2 units
Training loss	0.000	0.000	0.003	0.001
Test loss	0.000	0.000	0.003	0.012
Ratio of training to test data	50%	50%	50%	50%
Noise	0	0	0	0
Batch size	20	10	10	10

Reference: https://playground.tensorflow.org

Task 2

Abstract
The CIFAR10 dataset has 60000 color images of 32*32 size. There are 10 classes and each class has 6000 images. The goal of this problem is to create a fully connected neural network model that can classify these images successfully. Of the 60000 total images, 50000 images are for training and 10000 images are for testing the model efficacy. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

Problem Statement
The 10 classes of the dataset with corresponding class labels and names are given below:

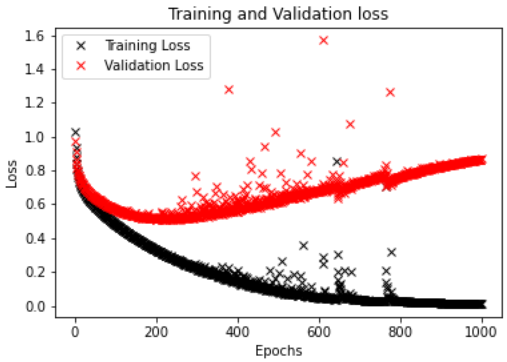
Class Label	Category
0	Airplane
1	Automobile
2	Bird
3	Cat
4	Deer
5	Dag
6	Frog
7	Horse
8	Ship
9	Truck

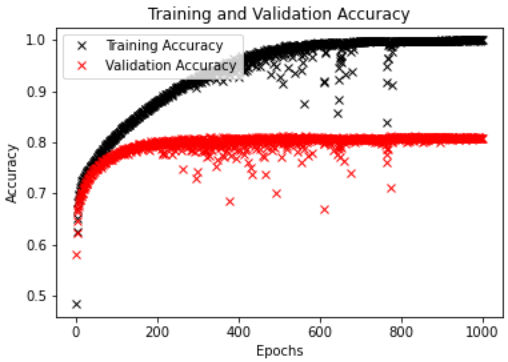
Proposed Solution
1. Preparing the dataset:
After importing and loading the dataset in our model from the built-in dataset collection of Keras, we have successfully set 10000 images for testing and 50000 for training our model. Of the 10 classes available, we have only used the first three classes for our case by dropping off all other classes. This results in ending up with 3000 for testing and 15000 for training images available for our model. Normalization of the pixels of all the images is carried to have values between 0 and 1 by dividing the individual pixel intensities with 255, which is the maximum pixel intensity. Moreover, one hot encoding is performed for the 3 classes. Note that the images were all reshaped to match our network.

2. Defining the model:
Complex classification problems which are typically not linearly separable can be modelled comfortably by the multilayered perceptrons (MLPs). Two hidden dense sequential layers are used in the model each having 256 hidden units. As for the activation function, softmax function

is used for the output layer and rectified linear unit (ReLU) function is used for the hidden layers. An optimizer called the stochastic gradient descent (SGD) algorithm is used with a small learning rate and a large momentum and decay. We have selected 'categorical cross entropy' as our loss function and 'accuracy' for our metrics.

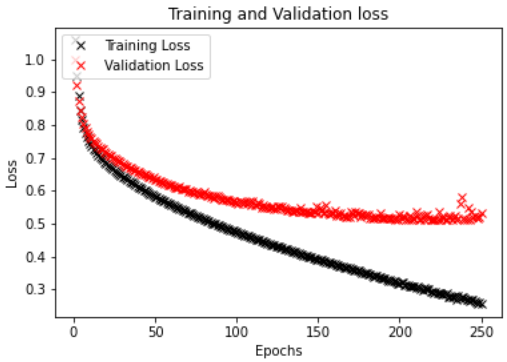
3. Running the model:
We have set aside 30% of our training data for the validation purpose while we have defined our model. We have used 1000 and 512 as the number of epochs and batch size for our model respectively. This gives us the following training and validation graphs for losses and accuracies over the number of epochs.

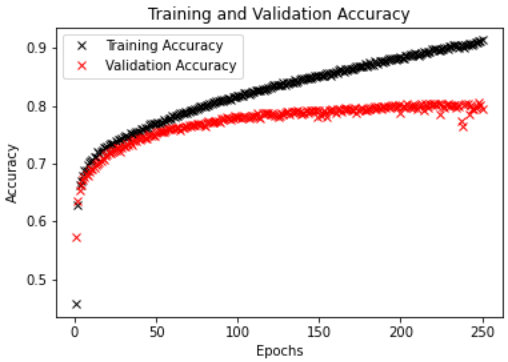




4. Fine Tuning the model:
Both the training and validation losses and accuracies become unstable after around 200 epochs. So, in the subsequent runs of the model, the number of epochs was set to 200. As for the learning rate, two different values were tried (0.01 and 0.001). The learning rate 0.001 performed slightly better. As for the optimizer algorithm, rmsprop gives noisy training and validation curves, whereas SGD doesn't have any such problem. Then, this fine tuned model was saved by the name 'cifar10_model.h5' for one last evaluation on the test set not seen by the model before. A table containing the hyperparameters used after fine tuning along with the output training and validation curves are as follows:

Hyperparameters	Selected Methods
Optimizer	Stochastic Gradient Descent (SGD)
Loss Function	Categorical Cross Entropy
Learning Rate	0.001
Metrics	Accuracy
Batch Size	512
Epoch	200





5. Model evaluation and results:
The final model was then loaded and evaluated by the test set. The accuracy found was approximately 81%.

Reference: https://www.cs.toronto.edu/~kriz/cifar.html

Task 3

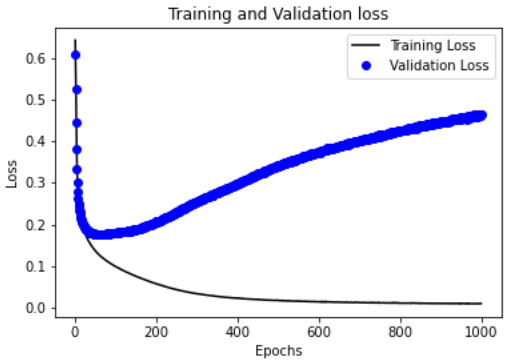
Abstract
The main target of this problem is to develop a model that classifies spam emails from non-spam emails. The dataset is called ‘spambase’ which is made available by the UCI repository. This contains a total of 4600 commercial labelled (i.e. spam or not spam) emails facilitating the necessary resources for building a spam filter.

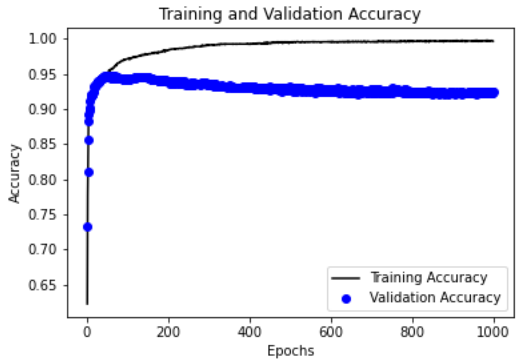
Problem Statement
The ‘spambase’ dataset available at the UCI repository contains 4,600 commercial emails where spammed emails are labelled as 1 and non spam emails are labelled as 0. The dataset also includes features which indicate the frequency of certain words used in the emails. The main target is to prepare the dataset so that a spam filter can be built.

Proposed Solution
1. Preparing the dataset:
Unlike Cifar10 dataset, the ‘spambase’ dataset does not come pre-processed from the dataset collections of Keras. Some preprocessing measures were taken, for instance, slicing the data for training and testing. In order to do so, the ‘spambase.data’ file was loaded as a dataframe. Then the train_test_split() function of the sklearn model was used to split 70% of the dataset for training and 30% for testing sets. The last column of the data frame includes either 1 or 0 which is our target label in order to show whether the email is spam or not. Again stratified sampling was chosen to preserve the relative class proportions. Another type of normalization technique called min-max normalization was trialed along with the standard scalar normalization to see changes in the performance of the model.

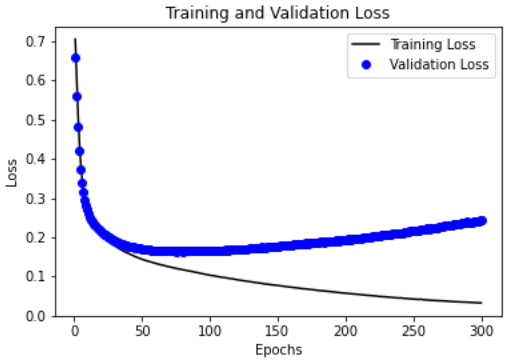
2. Defining the model:
This model was defined similar to that of task1. Complex classification problems which are not generally linearly separable can be modelled comfortably by the multilayered perceptrons (MLPs). Two hidden dense sequential layers are used in the model each having 256 hidden units. As for the activation function, rectified linear unit (ReLU) function is used for the hidden layers and softmax function is used for the output layer having only one unit. An optimizer called the stochastic gradient descent (SGD) algorithm is used with a small learning rate and a large momentum and decay. We have selected ‘binary cross entropy’ as our loss function and ‘accuracy’ for our metrics.

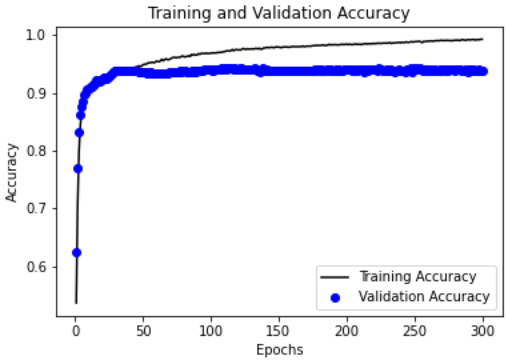
3. Running the model:
We have set aside 30% of our training data for the validation purpose while we have defined our model. We have used 1000 and 256 as the number of epochs and batch size for our model respectively. This gives us the following training and validation graphs for the losses and accuracies over the number of epochs.



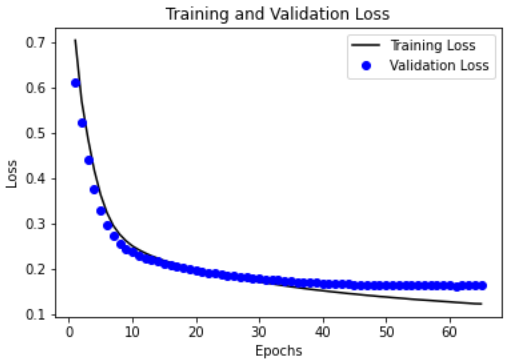


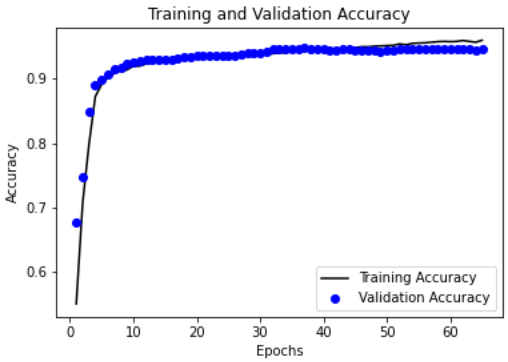
4. Fine tuning the model:
From the training and validation accuracy curves, it can be seen that it doesn't change much after around 200 epochs. Therefore, the epoch was set to 300 and other parameters were changed to see their effects. If the batch size was changed to 512 from 256, the curve becomes smoother. But, this means that the model would now generalise less to newer data. Thus, the batch size was kept constant at 256. Now, a different normalization technique was applied to see if there was an effect on the performance of the model.The curves below shows the performance of our model using the standard normalisation.





The minimum point converged somewhere between 50 and 70 epochs using the standard normalisation as can be seen from the two curves above. Then the learning rate was tweaked by changing it to 0.1, 0.01 and 0.001 and it was found that 0.01 gives the best results. The model optimizer was then tested with 'rmsprop'. As the hyperparameters were already fine tuned for the SGD algorithm, the performance of the rmsprop algorithm was not satisfactory. The model was run again after setting the number of epochs to 65 and saving it by 'spambase_model.h5' for the one time evaluation by the test data. The table below includes all the hyperparameters tuned for this model.





Hyperparameters	Selected Methods
Batch size	256
Optimizer	Stochastic Gradient Descent (SGD)
Loss Function	Binary Cross Entropy
Normalization	Standard Scaler
Learning Rate	0.01
Epoch	60

5. Model evaluation and results:
Finally the model was loaded and evaluated by the test dataset and the final model accuracy was around 93%.

Reference: https://archive.ics.uci.edu/ml/datasets/spambase

Task 4

Abstract
A well-known supervised learning technique called ‘regression’ can be used to extract useful information from a large data and effectively predict unknown values of interest. The UCI dataset named “crime and communities” gives the crime data of different communities with a total of 127 attributes, of which 5 are non-predictive. This resource is thus very useful when it comes to building a model for regression analysis and evaluation. The deep learning regression model built here can be used for predicting of the percentage of crime occurrence from the multi-modal data source which was obtained from the UCI repository.

Problem Statement
The UCI machine learning repository offers ‘crime and communities’ dataset having 128 attributes which also includes the target labels and instances of the year 1994. A summary of the dataset can be found in the table below where the values in parenthesis include the number of columns on the range.

Attributes	Category
Columns 0~4 (5)	Non-predictive
Rest (122)	Predictive
Column 128 (1)	Goal

The target is to train a deep learning regression model that predicts the crime rate or the number of violent crimes per 100K population given the features of the dataset.

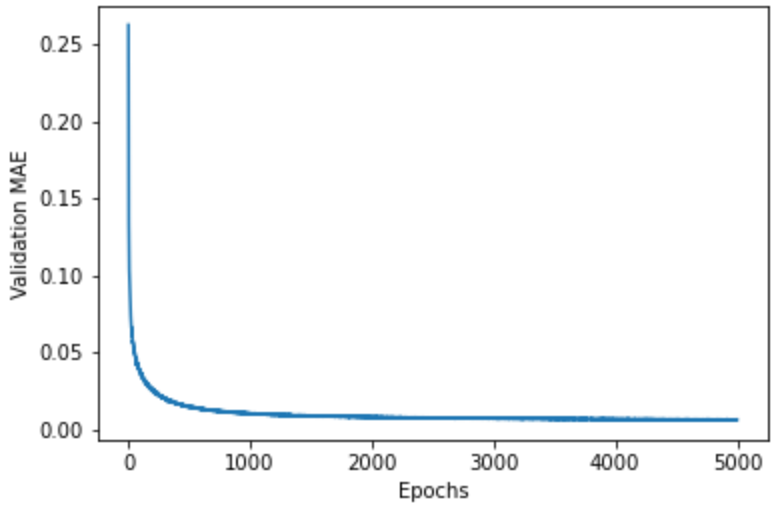
Proposed Solution:
1. Preparing the dataset:
Firstly, the dataset was prepared for the model after the ‘communities.data’ dataset was loaded into the data frame of the model. As several values were missing in the data frame, they needed to be taken care of before anything. They were categorized as ‘NaN’ while loading. Then, the non predictive columns and columns containing a lot of missing values were deleted from the dataframe. There was a single missing value in column 30, which was replaced by the mean of that column. After that, the data was separated into features and target labels. Then, 20% of the data was set aside for testing purposes. As for normalization, standard scalar normalization was used.

2. Defining the model:
The sequential model defined has two fully connected hidden layers with 64 hidden units or nodes. Rectified linear unit (ReLU) activation function is used for the hidden layers. However, no activation function was used at the output unit as we are interested in predicting numerical values rather than some probabilities since this is a regression problem. As for the optimization algorithm, ‘rmsprop’ was used and the loss function was ‘mean squared error’. The metric used to evaluate the performance of the model is mean absolute error (‘mae’). This particular metric was used because it provides an error value which is directly comprehended in the context of the problem i.e. the percentage of crime rate.

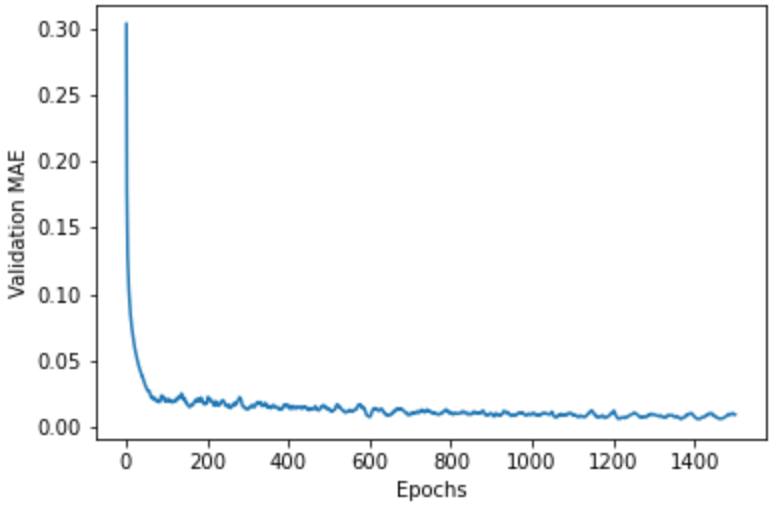
3. Running the model:
A cross-validation function was defined where the training dataset is divided into partial training and validation set in K number of folds (where K = 10). There was some kind of indexing error while the division of data was being manually done for partial training and validation set. The error part has been given below for reference.

```
partial_train_data = np.concatenate([train_data[:i*num_val_samples],
train_targets[(i+1)*num_val_samples:]], axis=0)
partial_train_targets = np.concatenate([train_targets[:i*num_val_samples],
train_targets[(i+1)*num_val_samples:]], axis=0)
```

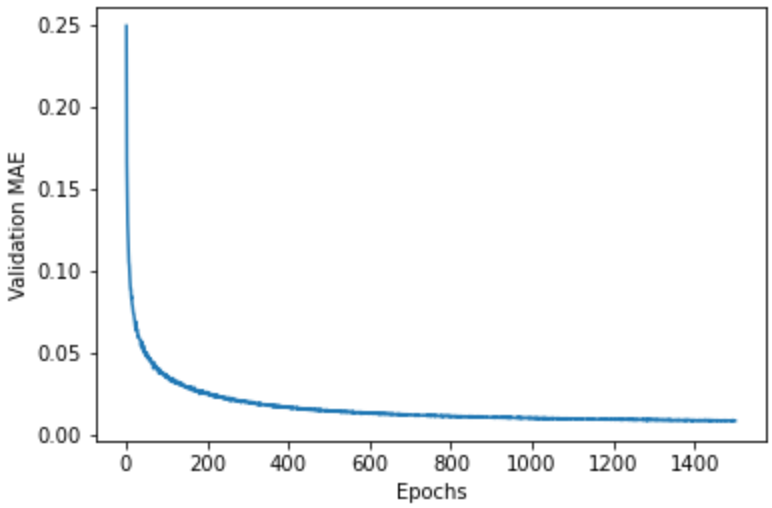
However, this was solved using the KFold function of the scikit learn library. A 10-fold cross validation function was defined. A batch size of 64 was chosen with 5000 epochs. The validation curve shows the performance of the model.



4. Fine tuning the model:
From the curve given above, we can see that the MAE decreases fast in the first 500 epochs and then slowly decreases afterwards. After 1000 epochs, the mean absolute error becomes almost constant. Thus, the number of epochs was set to 1500. The number of units in the first hidden unit was changed from 64 to 128 as well as the number of hidden layers were also increased from two to three but no big improvement was noticed. The optimizer algorithm was also changed from ‘rmsprop’ to ‘adam’ which gives the following curve.



The model gives a noisy response as can be seen from the curve above and hence the optimizer algorithm was changed back to ‘rmsprop’. The final fine tuned model was then saved by the name ‘crime_model.h5’ for the one last evaluation on the test data. The validation curve, and a table consisting of the fine tuned hyperparameters of the model are as follows.



Hyperparameters	Selected Methods
Optimizer	rmsprop
Hidden layers	2 (each having 64 units)
Loss Function	Mean Squared Error
Metrics	Mean Absolute Error
Epoch	1500

Batch size	64

5. Model evaluation and results:
The final model has an average validation score of 10.3% which was then evaluated on the test dataset. The final mean absolute error (mae) was found to be around 11%.

Reference: https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime