**Python and Deep Learning Course**

**Project Exam-2 Execution Report**

**Introduction:**

During the Deep learning lecture, we have learnt about key things mentioned below:

* What is deep learning
* Applications of deep learning and how it works
* Introduction to Keras Library and its basic operations.
* Neural Network
* Backpropagation
* Gradient Descent (Optimization Algorithm)
* Cost/Loss Functions
* Activation Function
* Linear Regression
* Cost/Loss Functions
* Learning Rate
* What is word embedding, types of ANNs and Recurrent Neural Network.
* Convolutional Neural Network, image classification with CNN
* Difference between FNN and RNN, what is RNN and LSTM.
* Basics of Autoencoders
* Role of Autoencoders in unsupervised learning
* Types of Autoencoders

This report illustrates how each task is performed such as way of approach, techniques applied, coding and executing the output. Every task detail is elaborated in the below sections.

**Team members:**

|  |  |
| --- | --- |
| **Name** | **Class Id** |
| Rajashekar Reddy Vemula | 47 |
| Shravyala Keesari | 22 |
| Sai Krishna Reddy Katta | 21 |
| Ashish Sharma | 44 |

**Objectives:**

The main objective of this assignment is to implement all the skills and knowledge which we have learnt during the 6 lectures of deep learning module.

The first question is to perform text classification using CNN model, apply Embedded layer and evaluating the model to see the overfitting problem the. Second question is performed in similar way to the first one, but instead of CNN model LSTM model is applied. The third question is image classification with CNN model, predicting one image from test data and also reporting the performance of the model with and without scaling. The fourth question works on creating a text generation language model and predicting the generated text which is passed on to the model. The fifth question is to apply the autoencoder and PCA on Cifar 10 data set and passing the result to CNN model and comparing the both models.

**Approaches/Methods**

We have used deep learning techniques learnt in class to solve these problems.

**Work Division:**

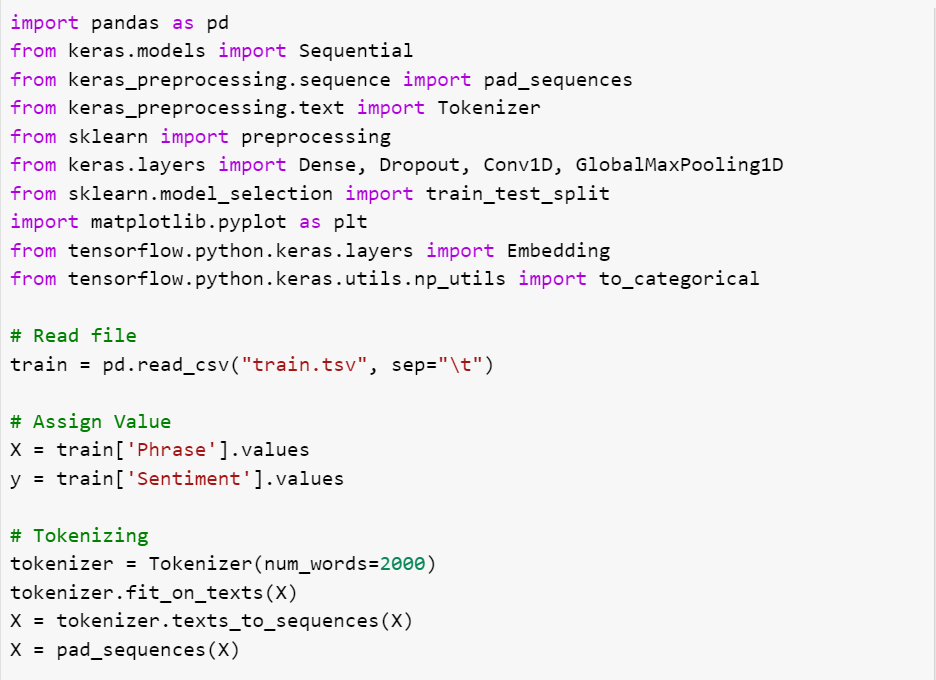
We have divided the task in such a way that one person one question and then we clubbed all ours tasks together into one single source file, a video explaining each code, a detailed report of execution.

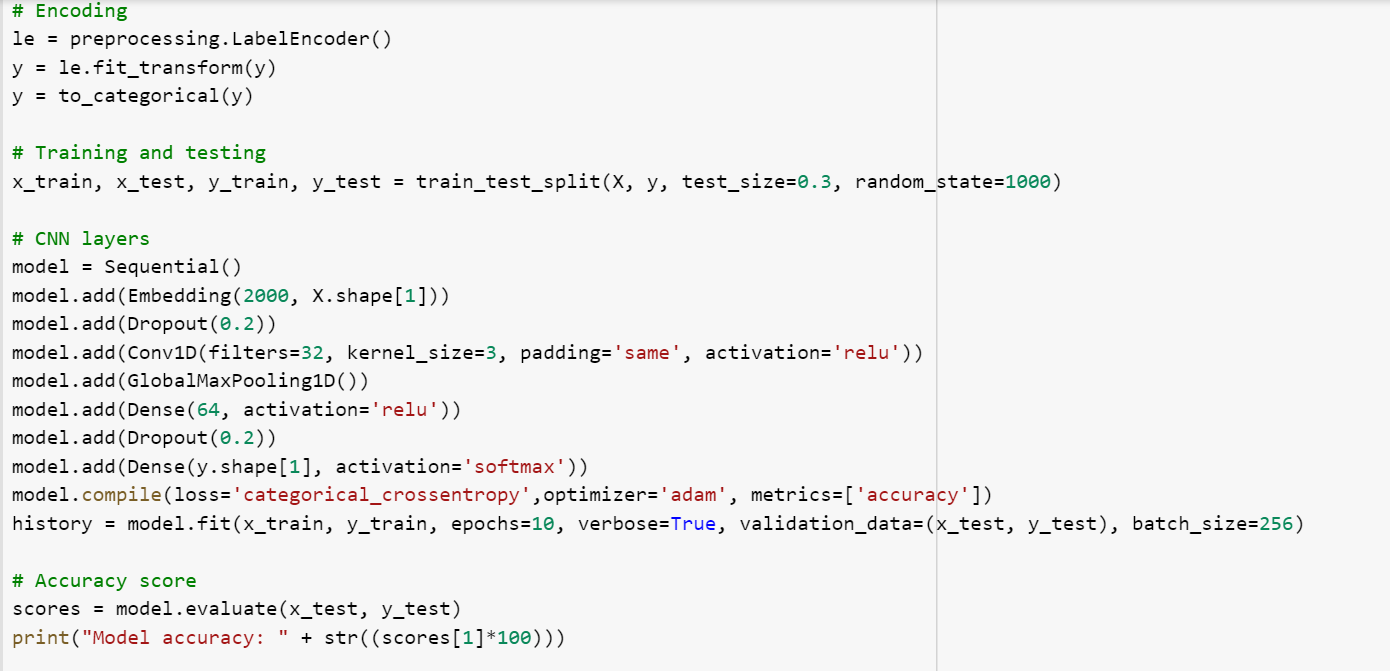
**Task 1:**

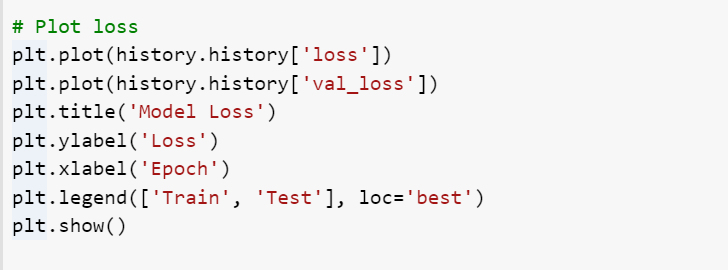
Implement text classification on the review’s sentiment dataset using CNN model.

1. Include Embedding layer in the design of your models and report if that leads to a better performance.
2. Plot loss of the model and report if you see any overfitting problem
3. What techniques you can apply to fix overfitting model

**Code:**







* 1. Yes, using embedding layer increased the accuracy of our model as shown below:

Accuracy without an embedding layer: 50.84%

A close up of a computer

Description automatically generated

Accuracy with an embedding layer: 63.62%

A screenshot of a computer

Description automatically generated

* 1. Our model shows an increase in loss after the second epoch which implies that there is an overfitting problem with our model.A screenshot of a cell phone

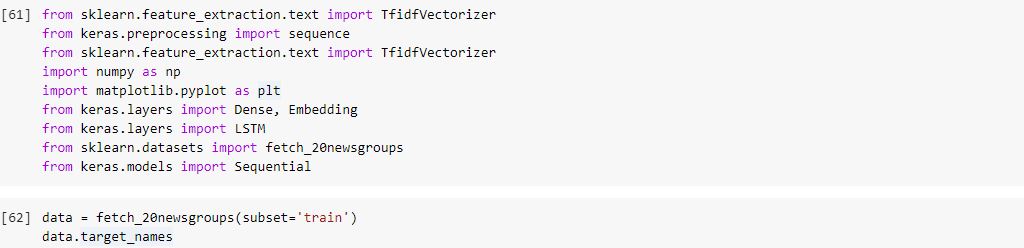
     Description automatically generated
  2. An overfitted model occurs when the learning rate is too high, in order to fix this problem the learning rate must be reduced.

**Task 2:**

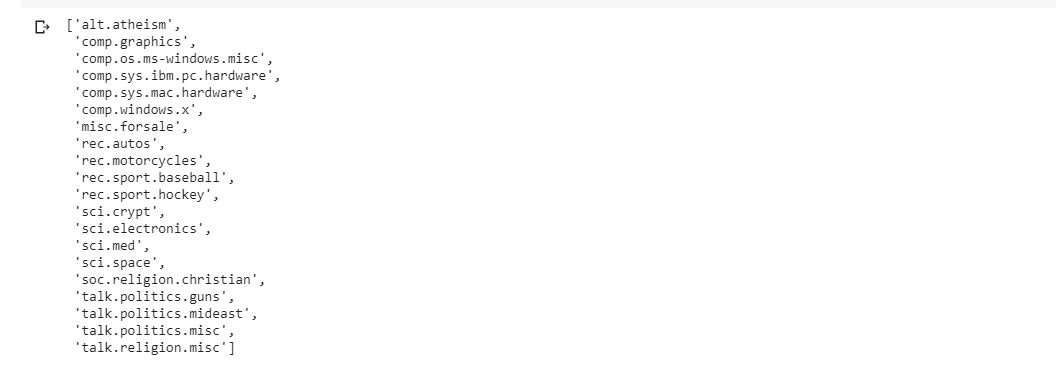
Implement text classification on the 20news\_group data set using LSTM model

1. Include Embedding layer in the design of your models and report if that leads to a better performance.
2. Plot loss of the model and report if you see any overfitting problem

**Code:**

****

All the libraries required for the task are been imported and data has been fetched from the 20newsgroup dataset.

****

All the target names available in the dataset have been printed, being a huge dataset it has many targets.

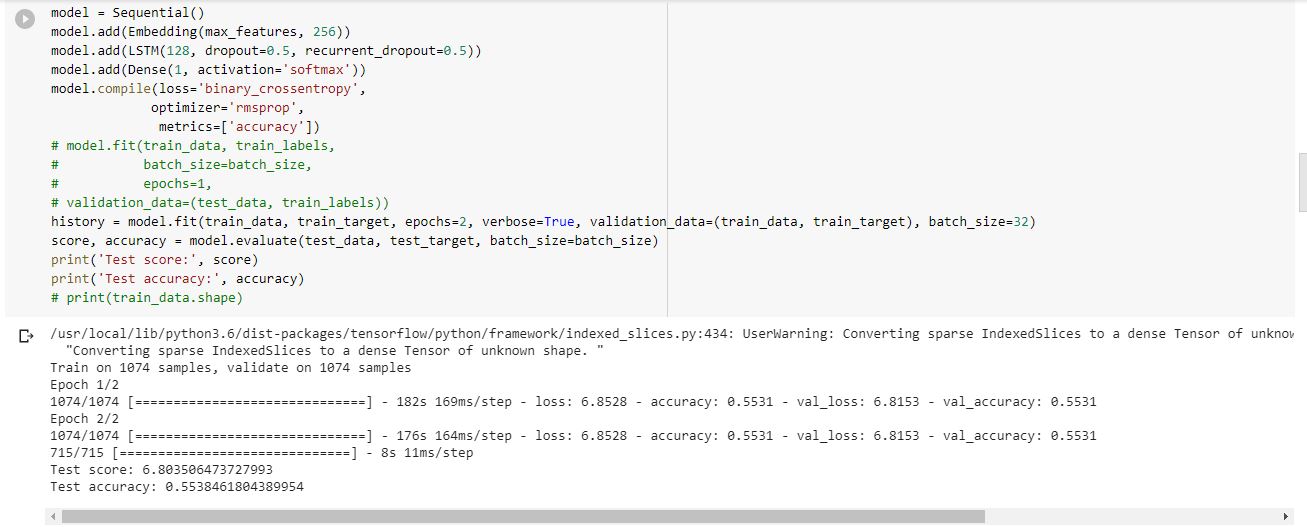
****

Max features are taken as 20000 means that only 20000 unique words would be taken into consideration in the dataset. Only two target values alt.atheism and rec.autos are considered because taking all the target values in the code makes the resources exhausted and runtime ends.

Vectorization has been done using TF-IDF with minimum value = 0.01 and maximum value = 0.95. todense() method is used to convert from sparse matrix to dense matrix has the vocabulary being huge in the dataset.

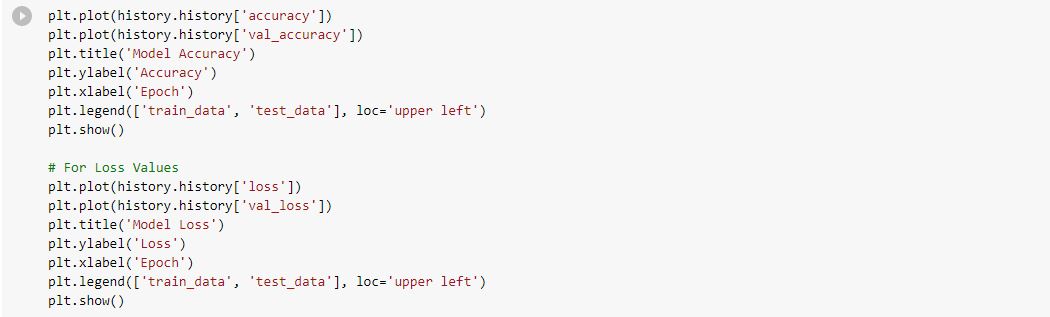
****

1074 rows are taken as the input from the dataset.

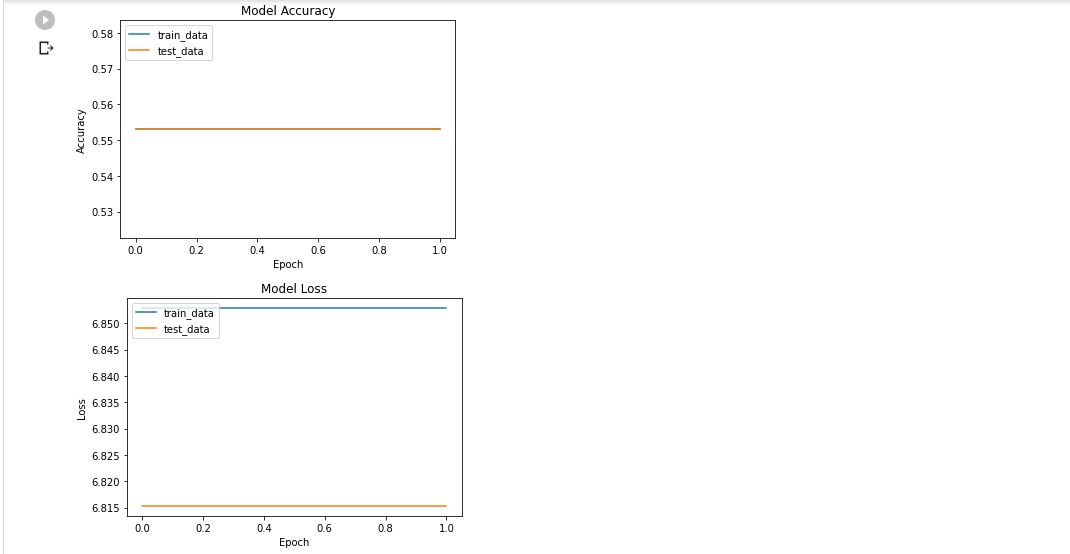
****

Model has been created with embedded layer in LSTM Model, model has been compiled, fitted and score and accuracy are calculated.

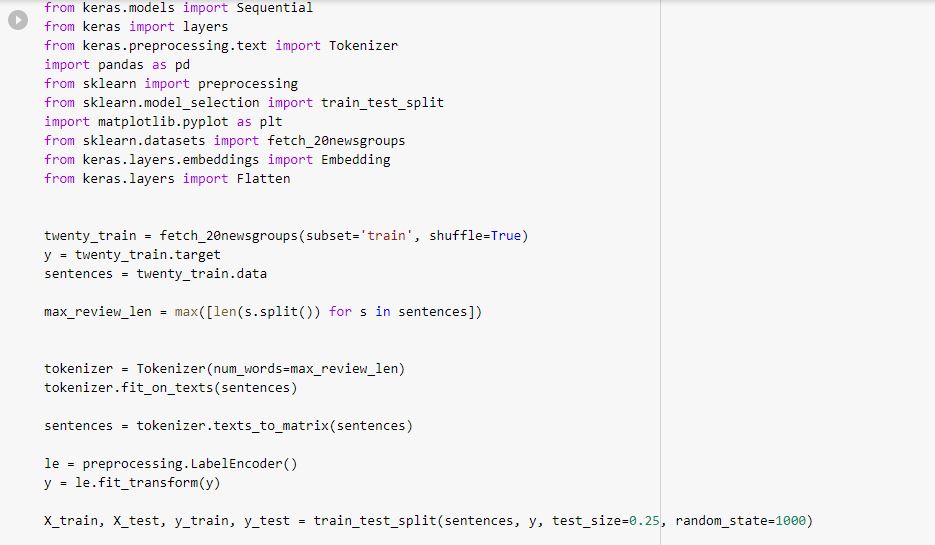
Accuracy is 0.55

****

Accuracy and loss are plotted using the history object.

****

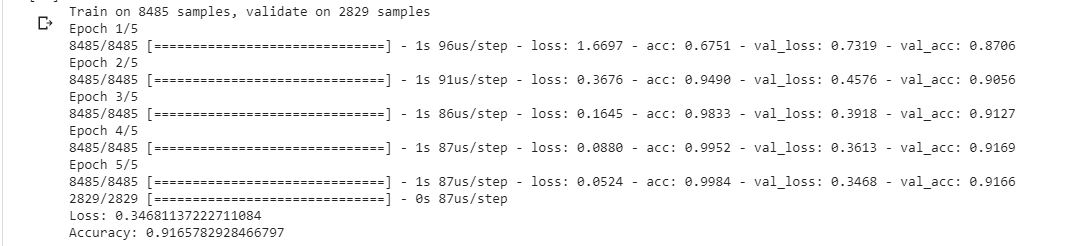
Accuracy and Loss graphs are plotted against the epochs.

****

Now, we are importing libraries for the other task with same dataset but without the embedded layer.

****

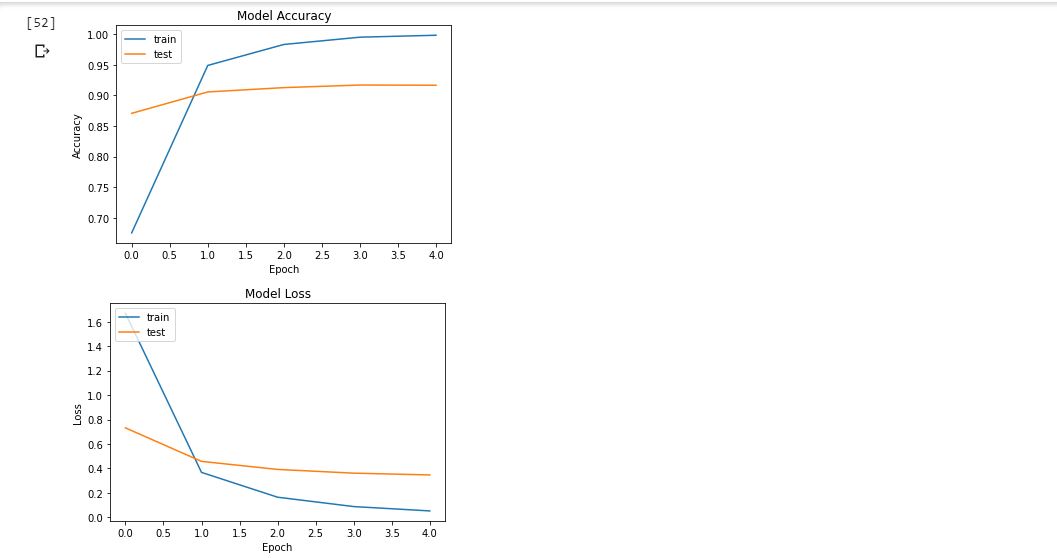
Model has been built, compiled and fitted. History object is used to plot the graphs of accuracy and loss.

****

Epochs have been run and the accuracy is calculated.

Loss is 0.34

Accuracy is 0.91

****

Accuracy and Loss graphs have been plotted using the history object.

Observation: Accuracy has been decreased when embedded layer is added in the LSTM model.

**Task 3:**

Implement image classification with CNN model, using one of the below datasets:

<https://www.kaggle.com/slothkong/10-monkey-species>

<https://www.kaggle.com/prasunroy/natural-images>

1. Report your classification result with and without doing scaling
2. Save the model and then predict on one of the test data. Report the prediction and check if it has been predicted correctly or not

**Code:**

Here we have considered monkey species data set. CNN model is applied and save. 4 test images are predicted, and prediction correctly shows the image belong to the specific group. Model performance is evaluated by reporting the loss and accuracy with scaling and without scaling. Below is the source code for this program. Scaling factor is removed from the same code for evaluating the model without scaling.









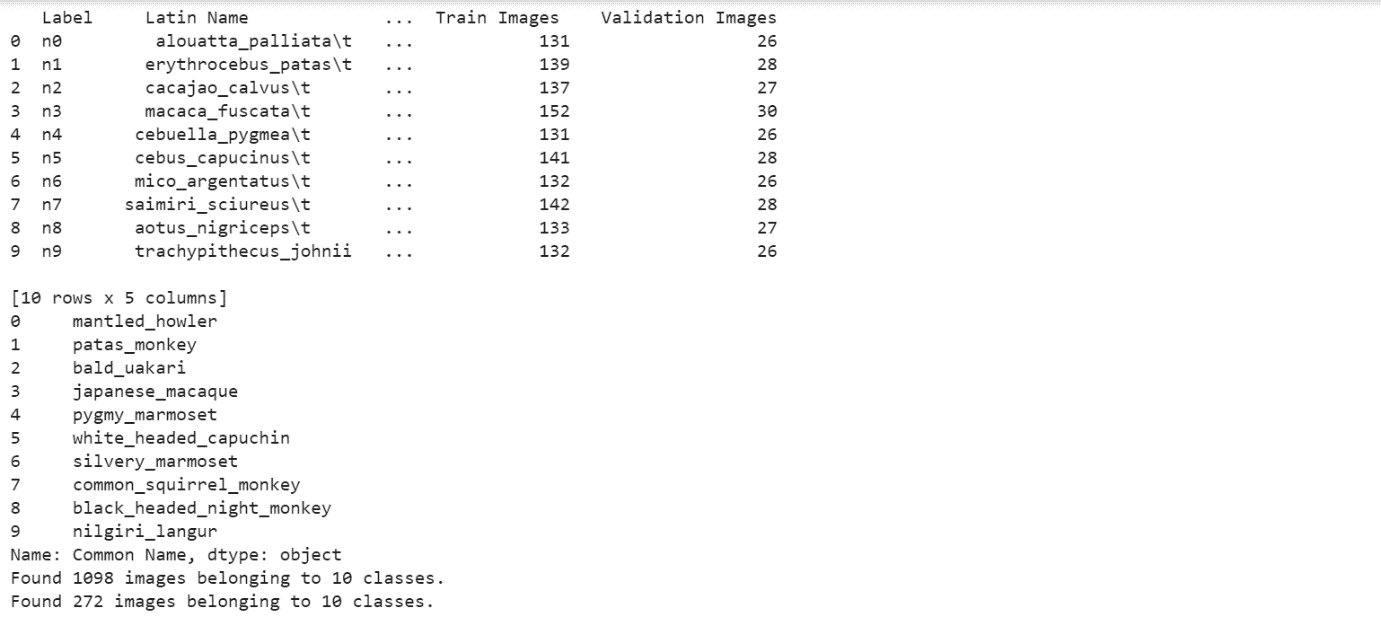




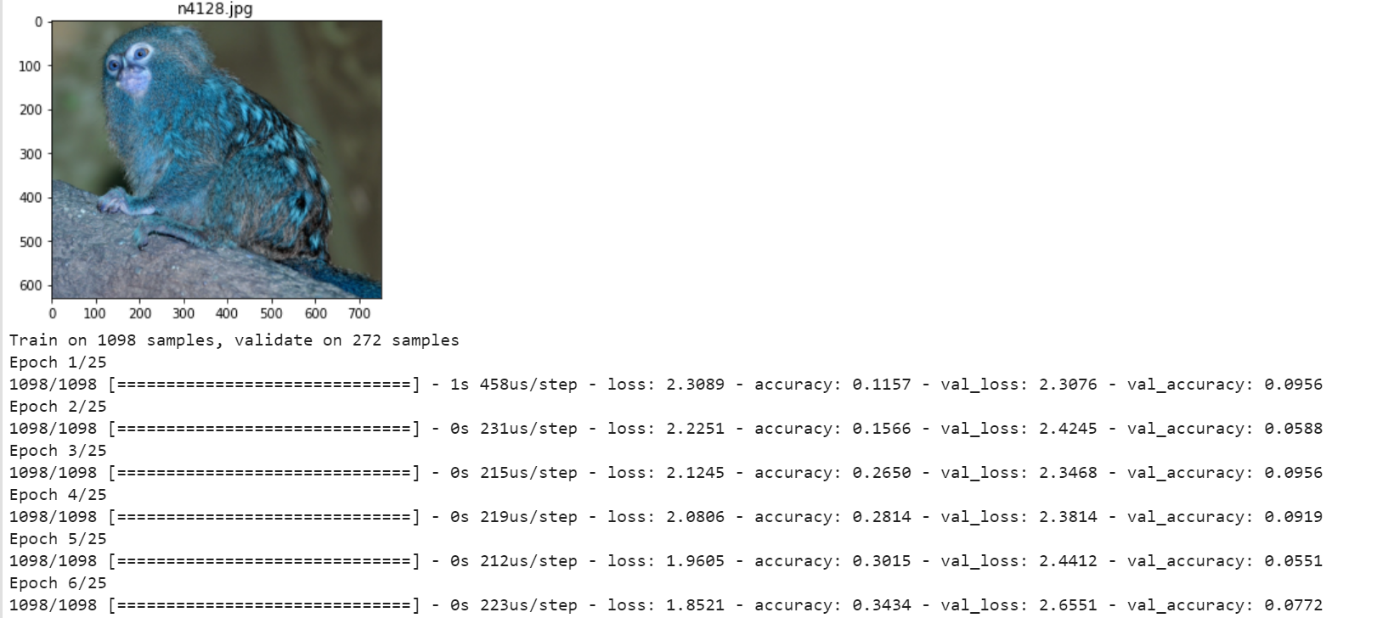


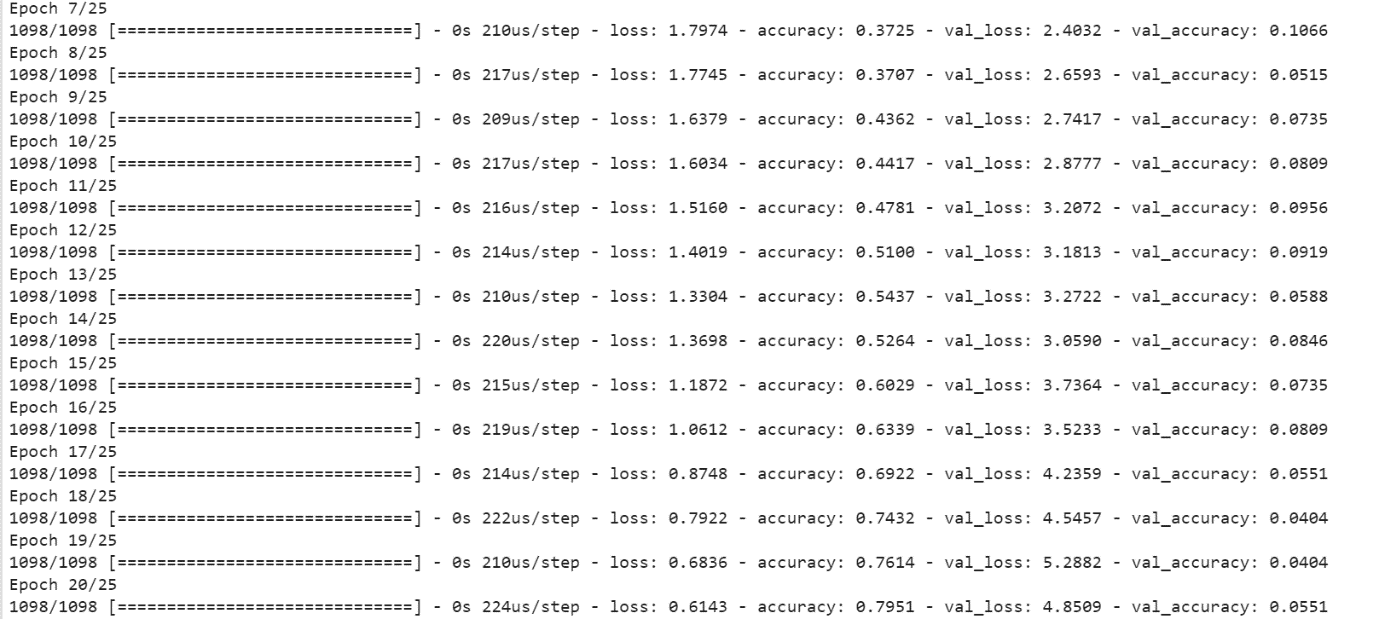


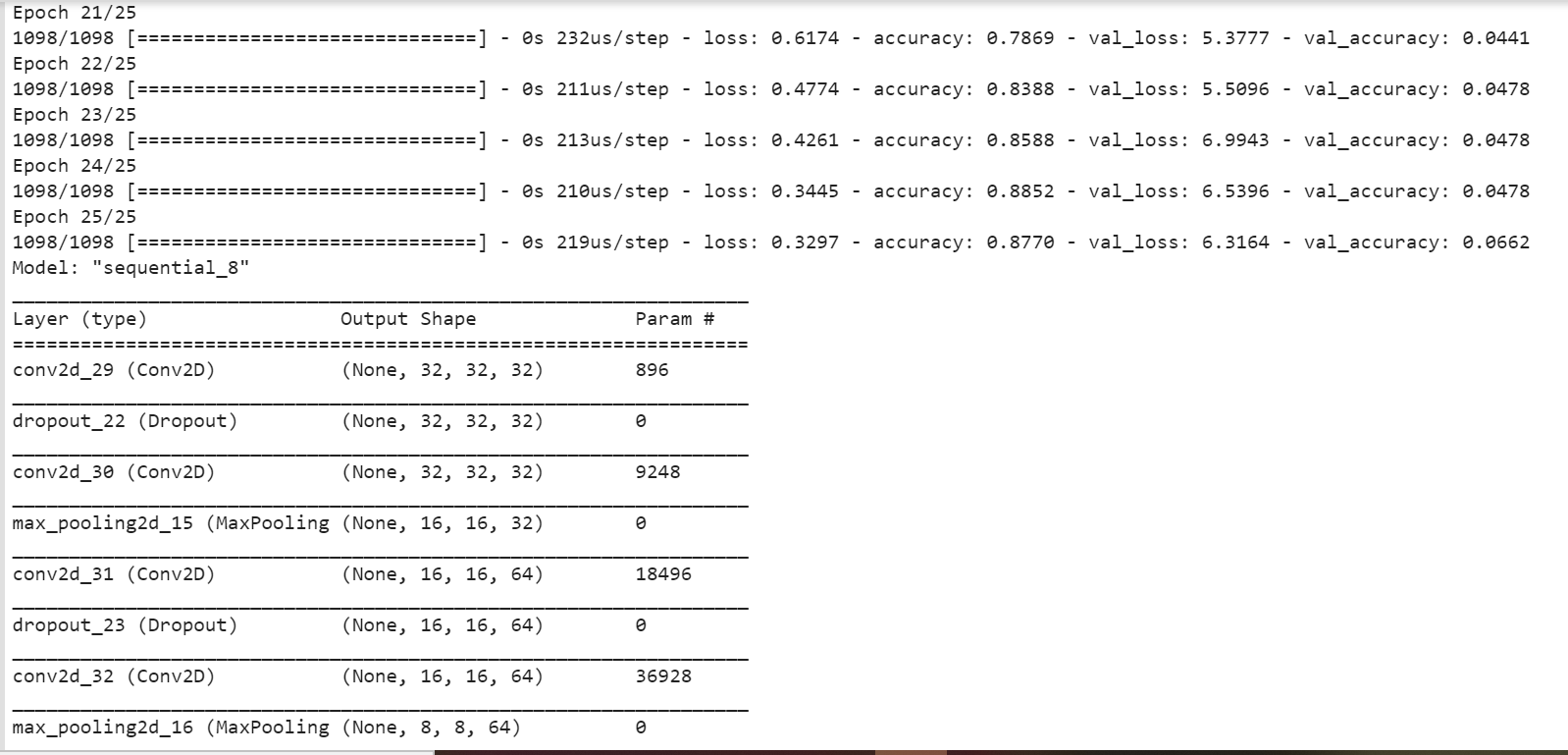
**Output images are shown below:**

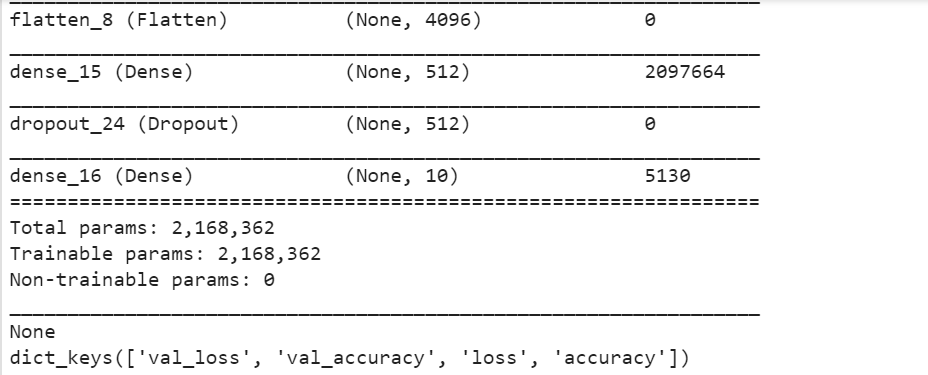


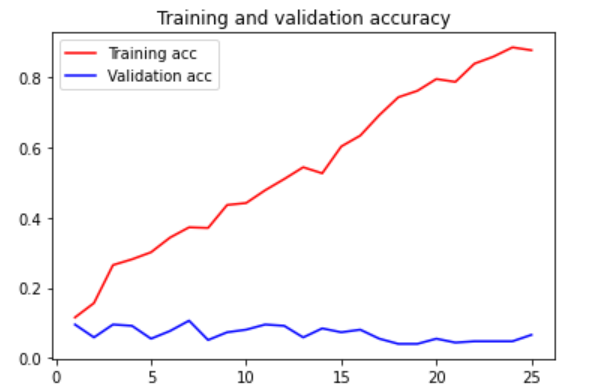


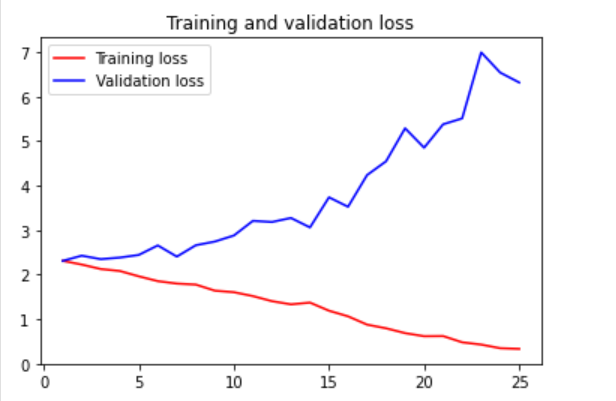


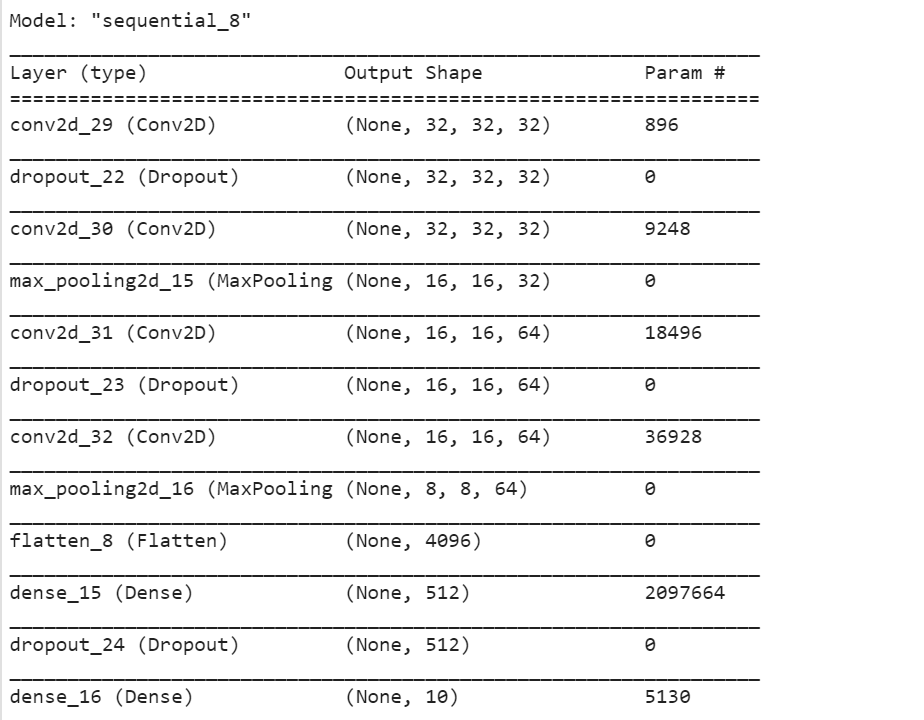


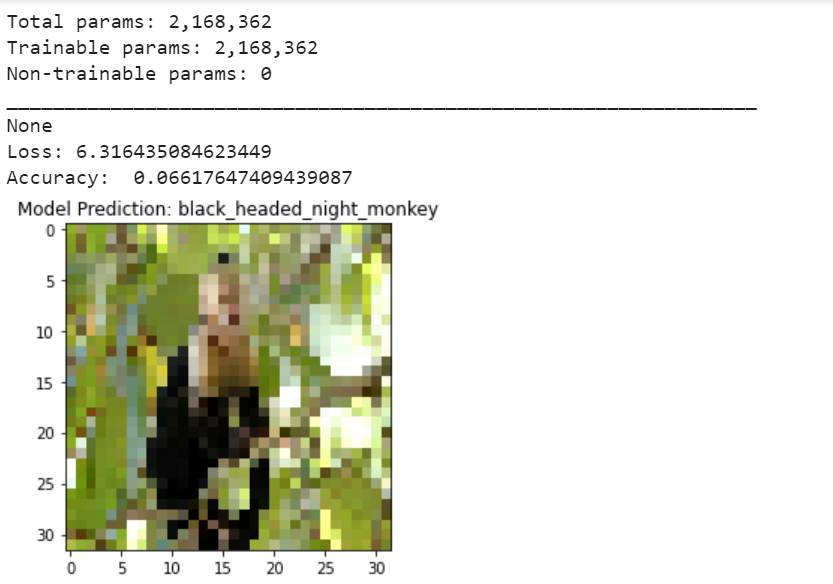


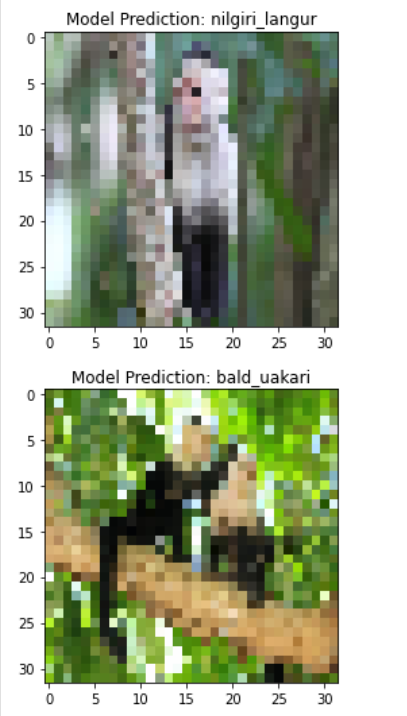


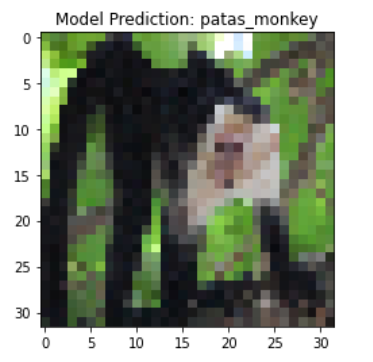




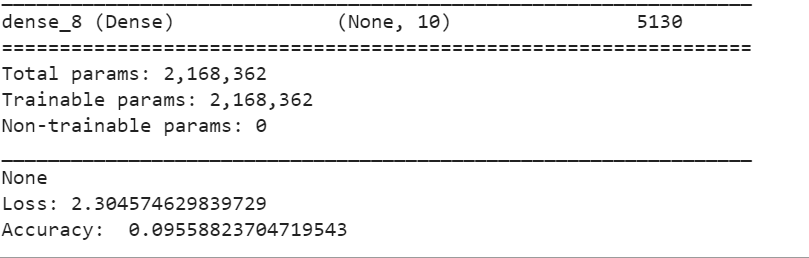


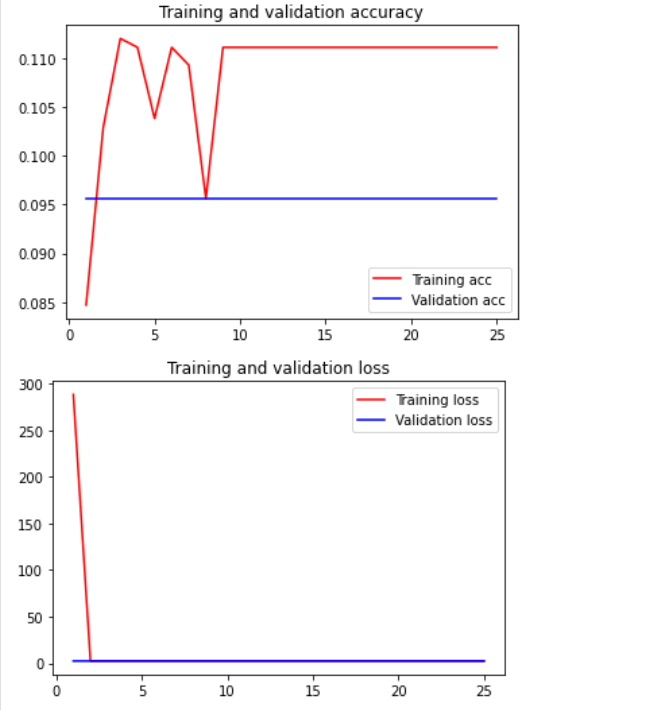






**Output of model without scaling is shown below:**





**Task 4:**

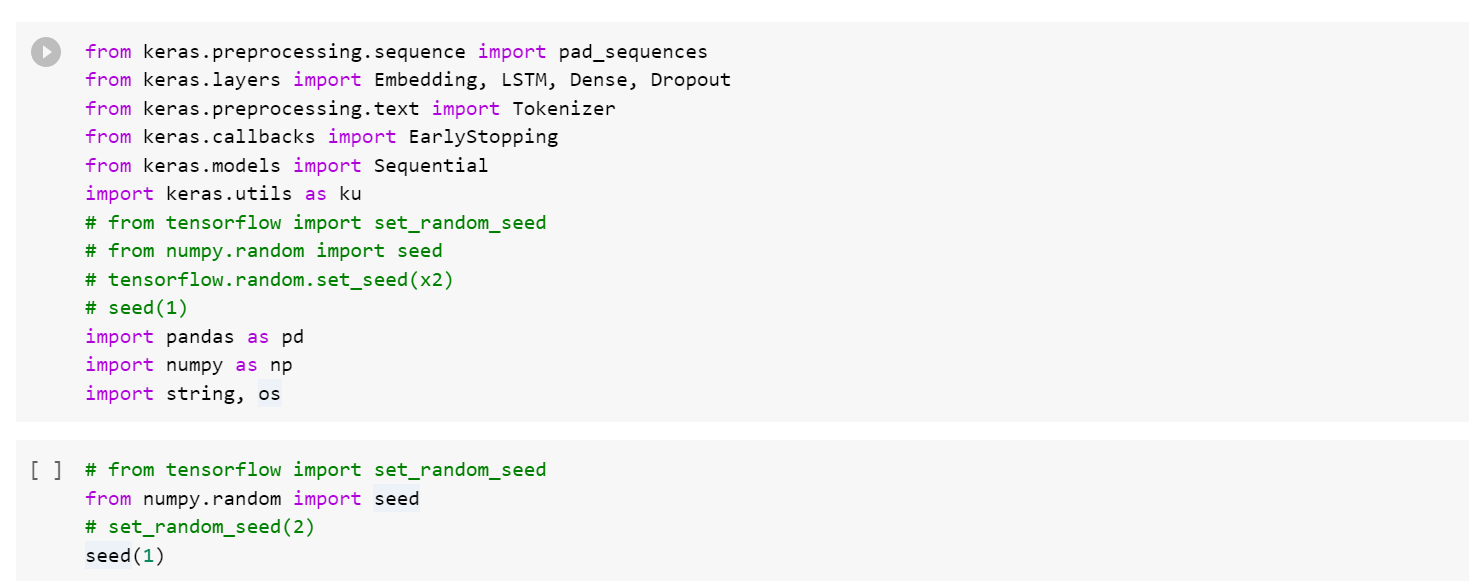
The purpose of this question is to learn about text generation. Use New York Times Comments and Headlines to train a text generation language model which can be used to generate News Headlines

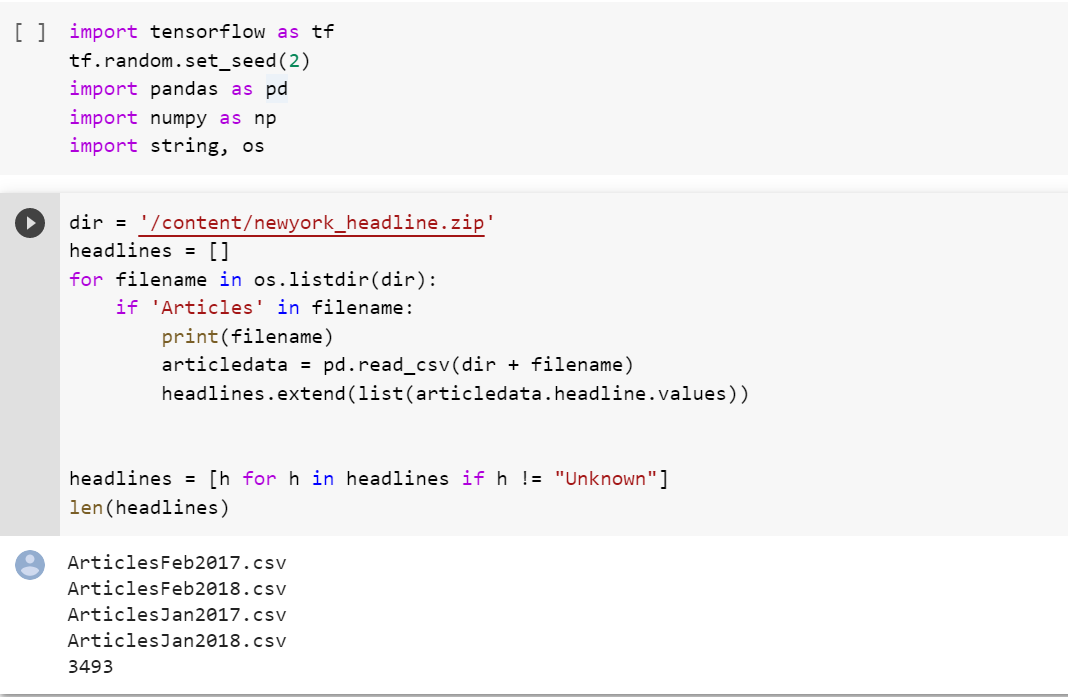
a. Pass a sample headline or word to the model and predict the generated headline

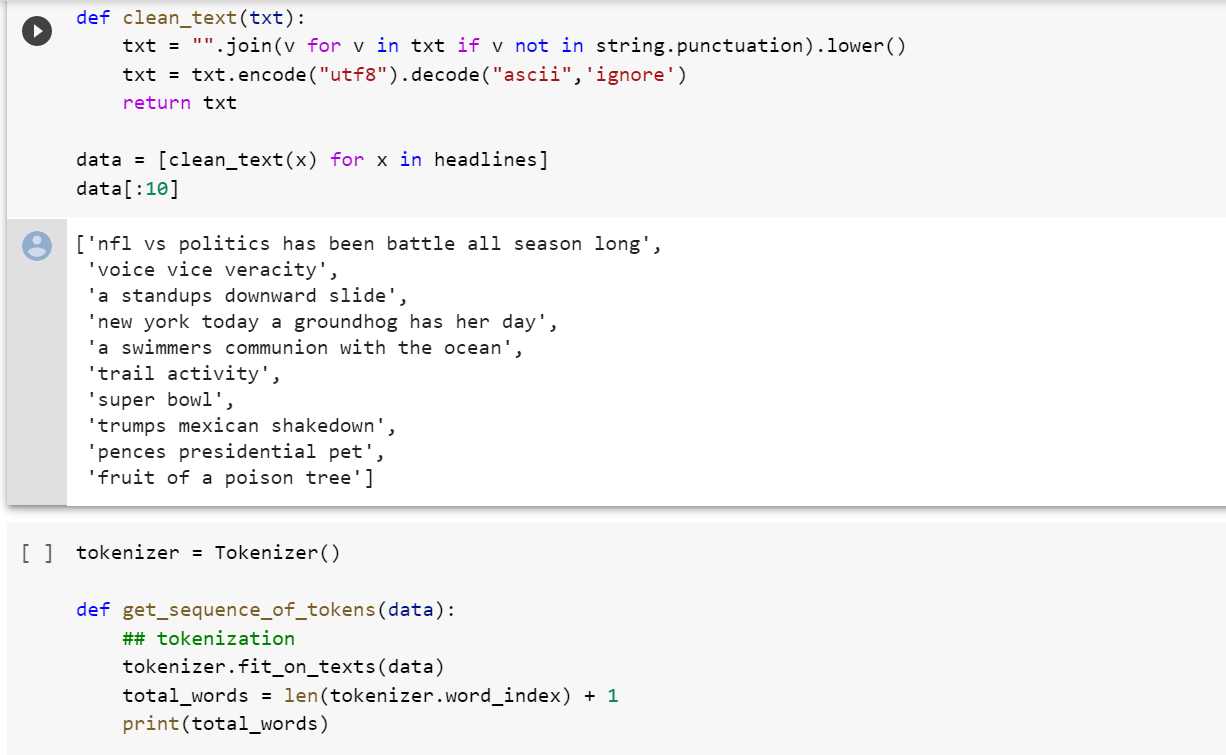
**Code:**

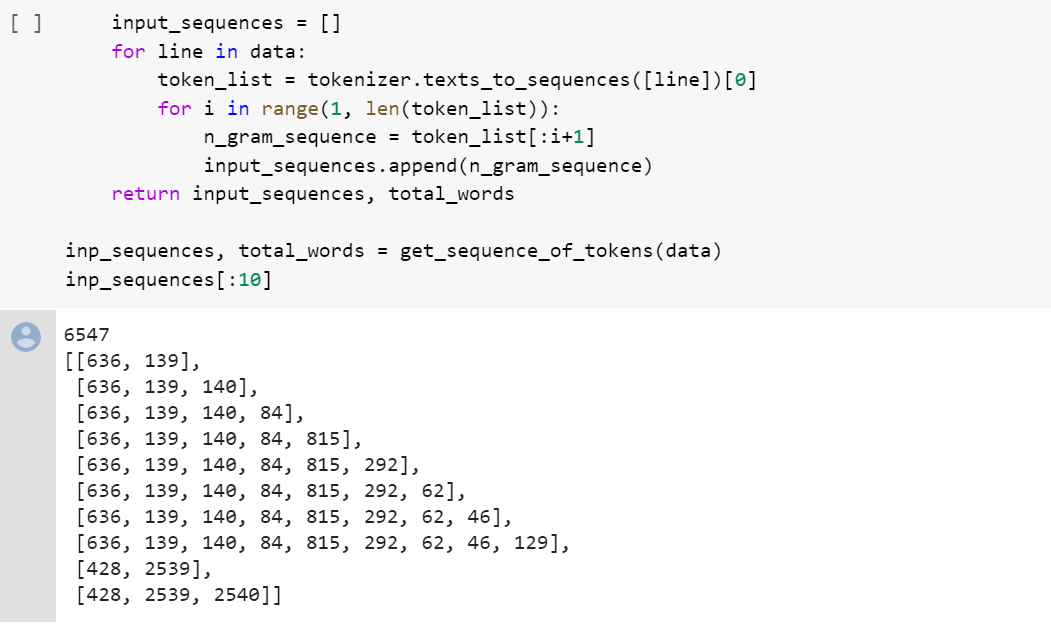
In this task, we have used the dataset of New York Times Comments and Headlines to train a text generation language model which can be used to generate News Headlines.

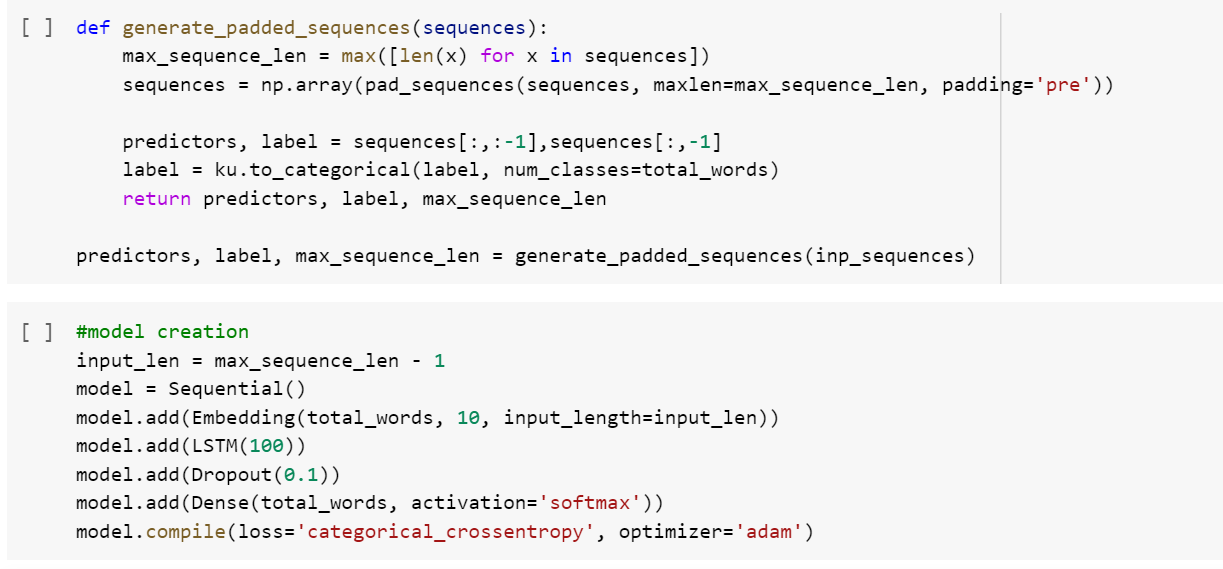
1. As the first step, we need to import the required libraries
2. Load the dataset of news headlines
3. In dataset preparation step, we will first perform text cleaning of the data which includes removal of punctuations and lower casing all the words.
4. The next step is Tokenization. Tokenization is a process of extracting tokens (terms / words) from a corpus. Python’s library Keras has inbuilt model for tokenization which can be used to obtain the tokens and their index in the corpus. After this step, every text document in the dataset is converted into sequence of tokens.
5. Now that we have generated a data-set which contains sequence of tokens, it is possible that different sequences have different lengths. Before starting training the model, we need to pad the sequences and make their lengths equal. We can use pad\_sequence function of Kears for this purpose. To input this data into a learning model, we need to create predictors and label. We will create N-grams sequence as predictors and the next word of the N-gram as label.
6. Now we can obtain the input vector X and the label vector Y which can be used for the training purposes. We will run this model for total 100 epochs. Train the model.
7. Write the function to predict the next word based on the input words (or seed text). We will first tokenize the seed text, pad the sequences and pass into the trained model to get predicted word. The multiple predicted words can be appended together to get predicted sequence.
8. Below code shows the step by step output.

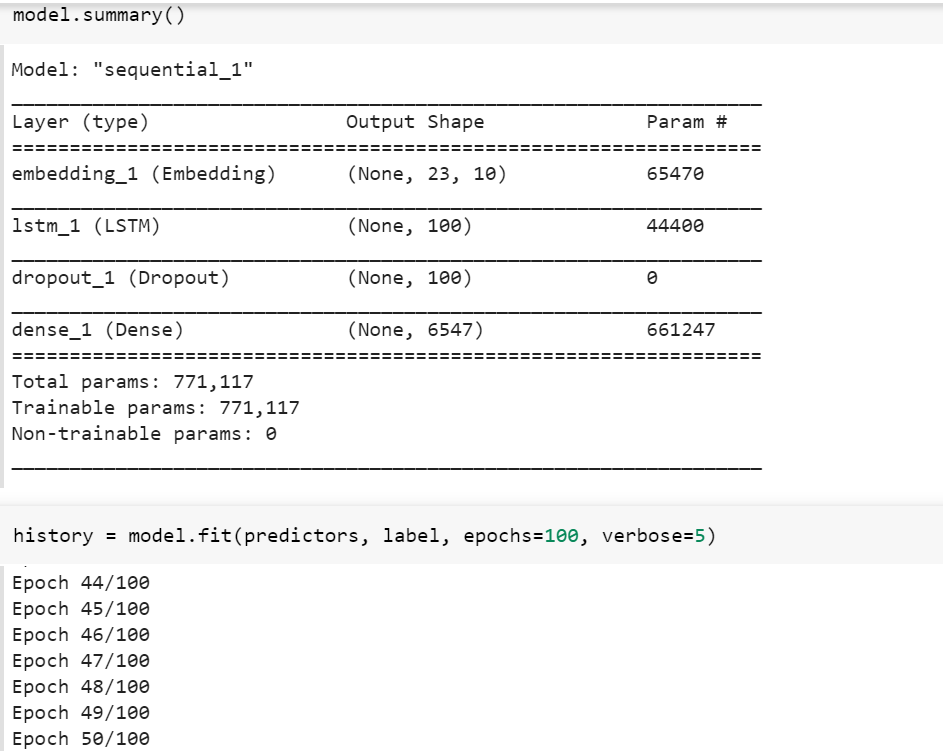




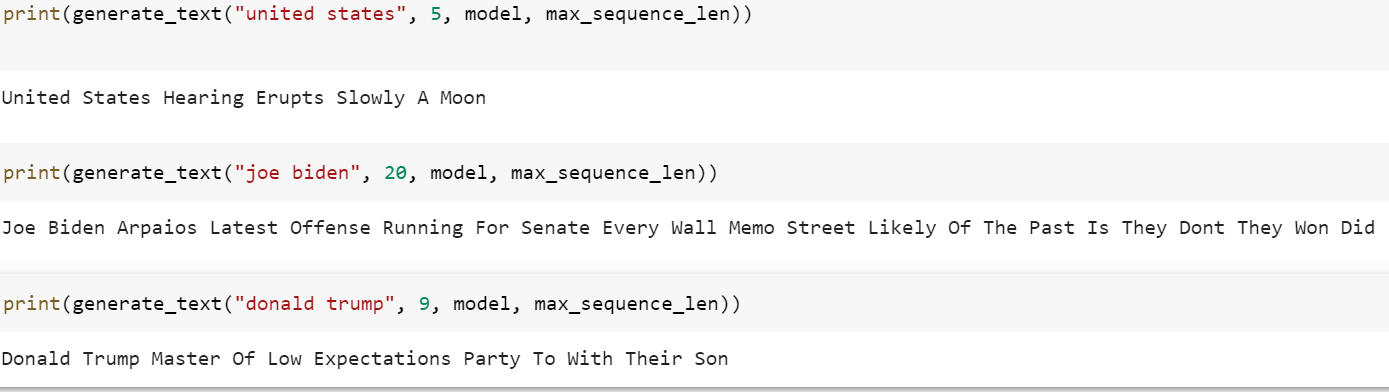












Thus, the text prediction is done.

**Task 5:**

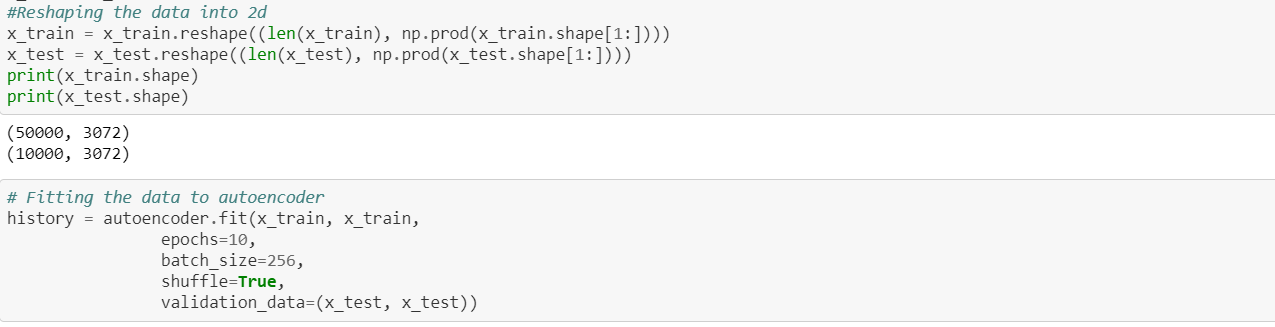
Apply Autoencoder on the Cifar\_10dataset and then pass the result of Autoencoder to CNN or LSTM or three layers model to classify data.

1. Repeat the same thing with PCA (apply PCA on the dataset and then pass the result to CNN or LSTM or three layers model)
2. Compare the result of both approach

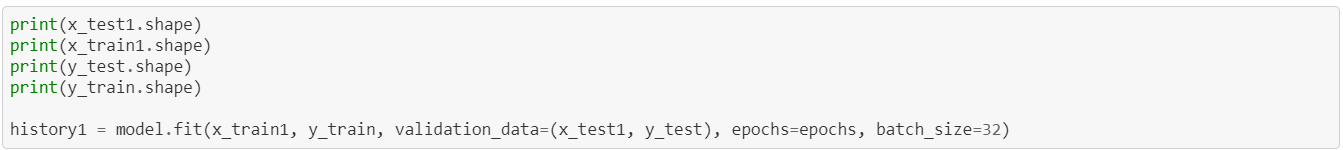
**Code:**

**For First part**



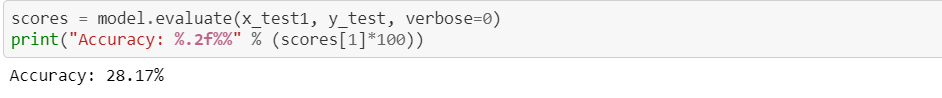




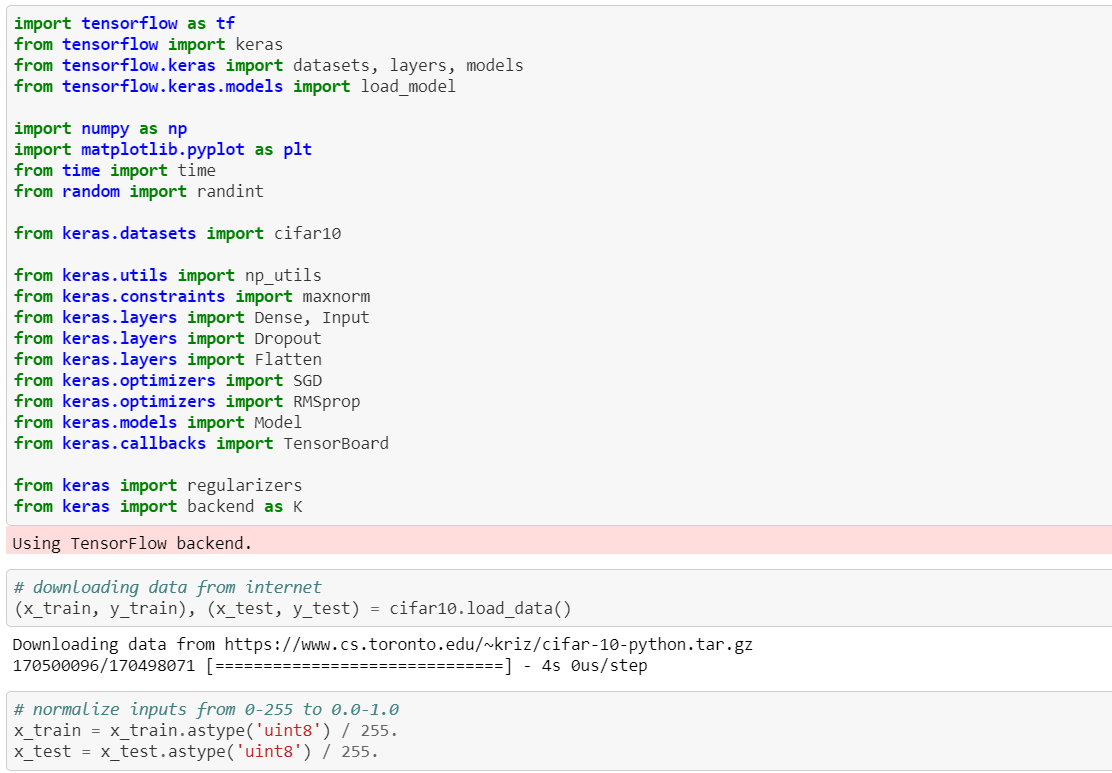




**Output for first part:**



**Code for Second part:**









**Output of Second Part:**



**c)** We can clearly see that when CNN is applied to output of autoencoder then the accuracy is 28.17. Whereas when we apply PCA to CIFAR10 dataset and then apply CNN the accuracy is almost double i.e.. 49.50. So we can clearly say that second scenario is good.

**Data Sets**

The data sets used for performing the above tasks are provided in the below links.

Task 1 Data Set: <https://umkc.app.box.com/s/nfaji3a8c86yfy5f4f9pzidmurvj82dz>

Task 2 Data Set:<https://umkc.app.box.com/s/qhtpjz7hdb8vdzmwscp7alvs6ilt0zf2>

Task 3 Data Set: <https://www.kaggle.com/slothkong/10-monkey-species>

Task 4 Data Set: <https://umkc.app.box.com/s/v9d6l40pmqsu4x4h6edhvgxx8yn8h47d>

Task 5 Data Set: Cifar-10 data set

**Parameters:**

There were not any strict parameters for these tasks. We have followed the instructions as stated in the tasks and performed accordingly.

**Evaluations and Discussions:**

After discussions with team members, tasks are shared among us for execution. After final execution of all the programs, we discussed about the approaches and methods followed in developing the code and getting the necessary output. Each task execution is briefed in the below video link.

<https://youtu.be/fph51y8bCOc>

**Conclusion:**

As stated in the objectives, we have successfully implemented all the tasks using the required methods. We have deployed the techniques which are learnt in the deep learning classes. Verified the results for each task and shared the lessons learnt among the team members.