Assignment – 3

Task: Analyze text and Sequence data using Recurrent Neural networks.

As per the instructions provided, using the below conditions,

1. Cutoff reviews after 150 words

2. Restrict training samples to 100

3. Validate on 10,000 samples

4. Consider only the top 10,000 words

5. Consider both an embedding layer, and a pretrained word embedding.

The following task is classified into three categories,

1. Using custom designed RNN and apply the implementation on IMDB data (downloaded from web)
2. Using pretrained embedding layer (Glove or Word2Vec) both are embedded techniques; in this task I'm using Glove implementation.
3. Finally, the implementation uses LSTMs to verify the performance (considering the fact LSTMS work good in most of the cases)

Custom Designed RNN:

The architecture implemented here is simple once and we used an embedding layer followed by a flatten layer.

```
!pip install keras_preprocessing
from keras.datasets import imdb
from keras import preprocessing

from keras_preprocessing.sequence import pad_sequences

max_features = 10000 # nr of words to consider as features
maxlen = 150 # cuts off the text after this nr of words among the most common words, i.e. 'max_features'
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words =max_features) # loads the data as list of integers


x_train = x_train[:100]
y_train = y_train[:100]

x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)
```

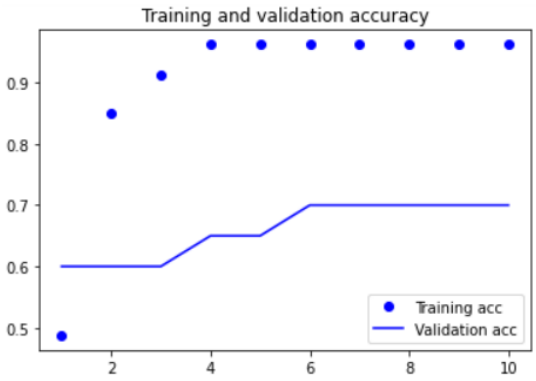
Using keras implementation, I designed the layer as per the above. Clearly, it is sequential model with embedding, flattening and dense layers.

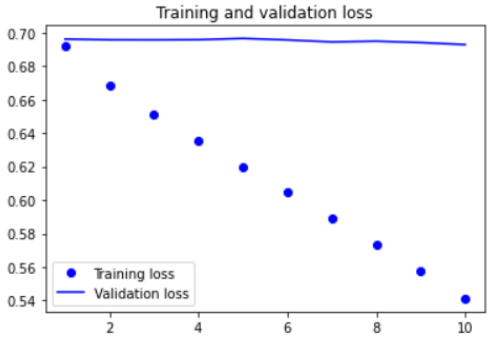
I used rmsprop optimizer, loss as binary cross entropy and metric is accuracy.

After running to 10 epochs, I got the following result.

```
Epoch 10/10
3/3 [==============================] - 0s 12ms/step - loss: 0.5412 - acc: 0.9625 - val_loss: 0.6928 - val_acc: 0.7000
```

Below are the plots for the above task,





By observing the plots, we can notice the model works better on the training data and when we notice the validation accuracy performs constant results from epoch 6 to epoch 10, it's a decent result.

Using Glove Embedding:

Brief information above glove:

GloVe aims to capture the semantic relationships between words in a corpus of text by representing each word as a vector in a high-dimensional space. The GloVe algorithm uses co-occurrence statistics to learn the vector representations of words. It calculates the co-occurrence probabilities between each pair of words in a corpus and then uses a technique called matrix factorization to map these probabilities into a low-dimensional space. The resulting vectors capture the underlying relationships between words in the corpus.

I downloaded glove file from the web and imported in my code

```
embedding_dim = 100 # GloVe contains 100-dimensional embedding vectors for 400.000 words

embedding_matrix = np.zeros((max_words, embedding_dim)) # embedding_matrix.shape (10000, 100)
for word, i in word_index.items():
    if i < max_words:
        embedding_vector = embeddings_index.get(word) # embedding_vector.shape (100,)
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector # Words not found in the mebedding index will all be
```

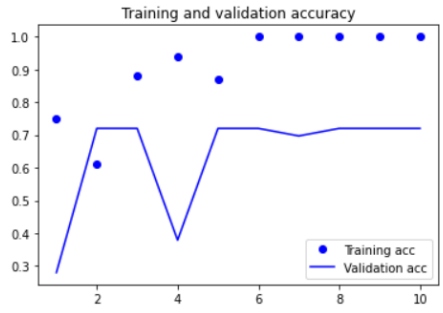
Model Definition

```
from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense

model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length = maxlen))
model.add(Flatten())
model.add(Dense(32, activation = "relu"))
model.add(Dense(1, activation="sigmoid"))
model.summary()
```

And the results are below,

```
Epoch 10/10
4/4 [==============================] - 1s 188ms/step - loss: 0.0118 - acc: 1.0000 - val_loss: 1.0974 - val_acc: 0.7199
```



The results were less when compared to the custom designed architecture. May be if we improve the structure or enhance the design the result would better.

Technique	**Training Accuracy**	**Validation Accuracy**	**Training Loss**	**Validation Loss**
Custom designed RNN	0.9625	0.7000	0.5412	0.6928
Using Glove Embedding	1.0000	0.7199	0.0118	1.0974

From the above comparison, it is evident that the Glove embedding model has slightly better validation accuracy than the customized model.

In the next step, I increased the training sample to 200 and implemented the same process for both embedded and pre-trained embedded models. The following table shows the results of both models

Technique	**Training Accuracy**	**Validation Accuracy**	**Training Loss**	**Validation Loss**
Custom designed RNN	1.0000	0.6500	0.5422	0.6977
Using Glove Embedding	1.0000	0.7068	0.0173	0.6352

When the training sample has increased to 200 the validation accuracy of both the customized model and Glove embedding model has significantly dropped compared to 100 training models.

Using LSTMs:

Considering the important fact about LSTMs, they are designed to solve the vanishing gradient problem in traditional RNNs, which occurs when gradients become too small during backpropagation, making it difficult to learn long-term dependencies.

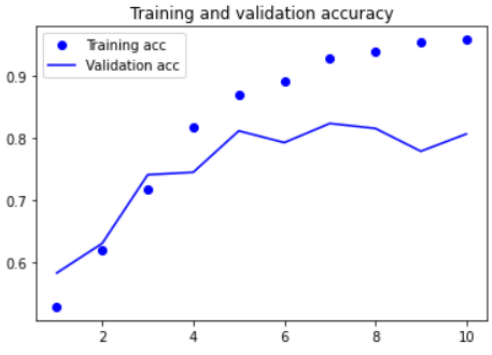
```
from keras.models import Sequential
from keras.layers import Embedding
from keras.layers import LSTM
from keras.layers import Flatten, Dense

model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation = "sigmoid"))

model.compile(optimizer = "rmsprop",
              loss = "binary_crossentropy",
              metrics = ["acc"])
history = model.fit(input_train, y_train, epochs=10, batch_size=128, validation_split=0.2)
```

I added LSTM in the design, and we can observe the accuracy of the model below,

Technique	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
Custom designed RNN	96.29	86.70	0.1090	0.345



Conclusion:

A training sample of 100 has produced better accuracy for both the customized model and Glove embedding model than that of 200 samples. This implies that the model is overfitting when the training sample is increased. Comparatively Glove embedding model has better performance than the customized model. This is because the Glove embedding is a pre-trained model and it is trained on high-quality data. At the end, I have also tried LSTM (long short-term memory networks). This model has produced the best results compared to other models. There can be many reasons for that like the training sample is different, and the way a model functions is also different from others.