Sentiment Analysis of Movie Reviews

Nikhil Cheke (1611CS02)
Harsimran Bedi(1611CS03)

Problem Statement

- Automatic classification of subjectivity of Movie reviews

- Binary classification task

- Two classes: Positive and Negative

Methodology

Dataset:

- IMDB movie review dataset
- Keras has built in IMDB data set
- 50,000 reviews evenly split into train and test
- Positive and negative class has equal number of reviews

Methodology

Word Embeddings:

- Need of numeric representation
- Word embeddings of input data created
- One word converted to a vector of numbers
- Similarity between words is similarity of its vectors

Experiments

The different Neural network techniques:

- Multi layer perceptron model
- 1-D CNN
- LSTM
- LSTM with CNN

Observations

Sr. No	Neural Network	Parameters	Accuracy(%)
1.	Multi layer Perceptron	hidden layer=1 epochs=2	87.37
2.	LSTM	memory units=100 epochs=3 dropout=0.2	85.56
3.	LSTM and CNN	memory units=100 epochs=3	86.15
4.	1-D CNN	hidden layer=1 epochs=2 dropout=0.0 strides=2	87.53
5.	1-D CNN	hidden layer=1 epochs=2 dropout=0.2 strides=2	88.70
6.	1-D CNN	hidden layer=1 epochs=2 dropout=0.4 strides=2	88.92
7.	1-D CNN	hidden layer=1 epochs=2 dropout=0.4 strides=1	89.16

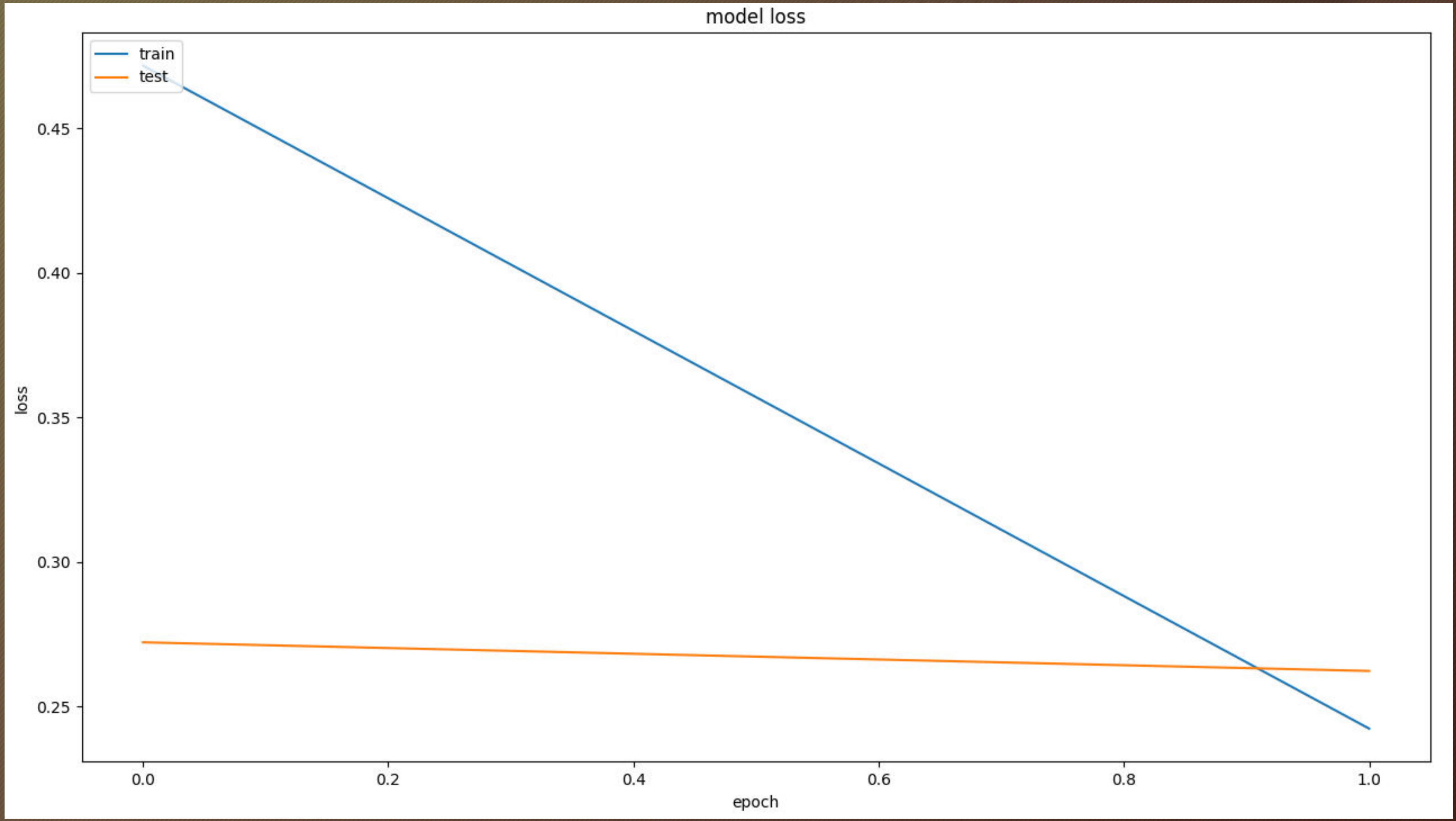
Code snippet

```
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)

# pad dataset to a maximum review length in words
max_words = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_words)
X_test = sequence.pad_sequences(X_test, maxlen=max_words)

# create the model
model = Sequential()
model.add(Embedding(top_words, 32, input_length=max_words))
model.add(Dropout(0.4))
model.add(Conv1D(filters=32, strides=1, kernel_size=3, padding='SAME', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(250, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())

# Fit the model
history=model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=2, batch_size=128, verbose=2)

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))

plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
```

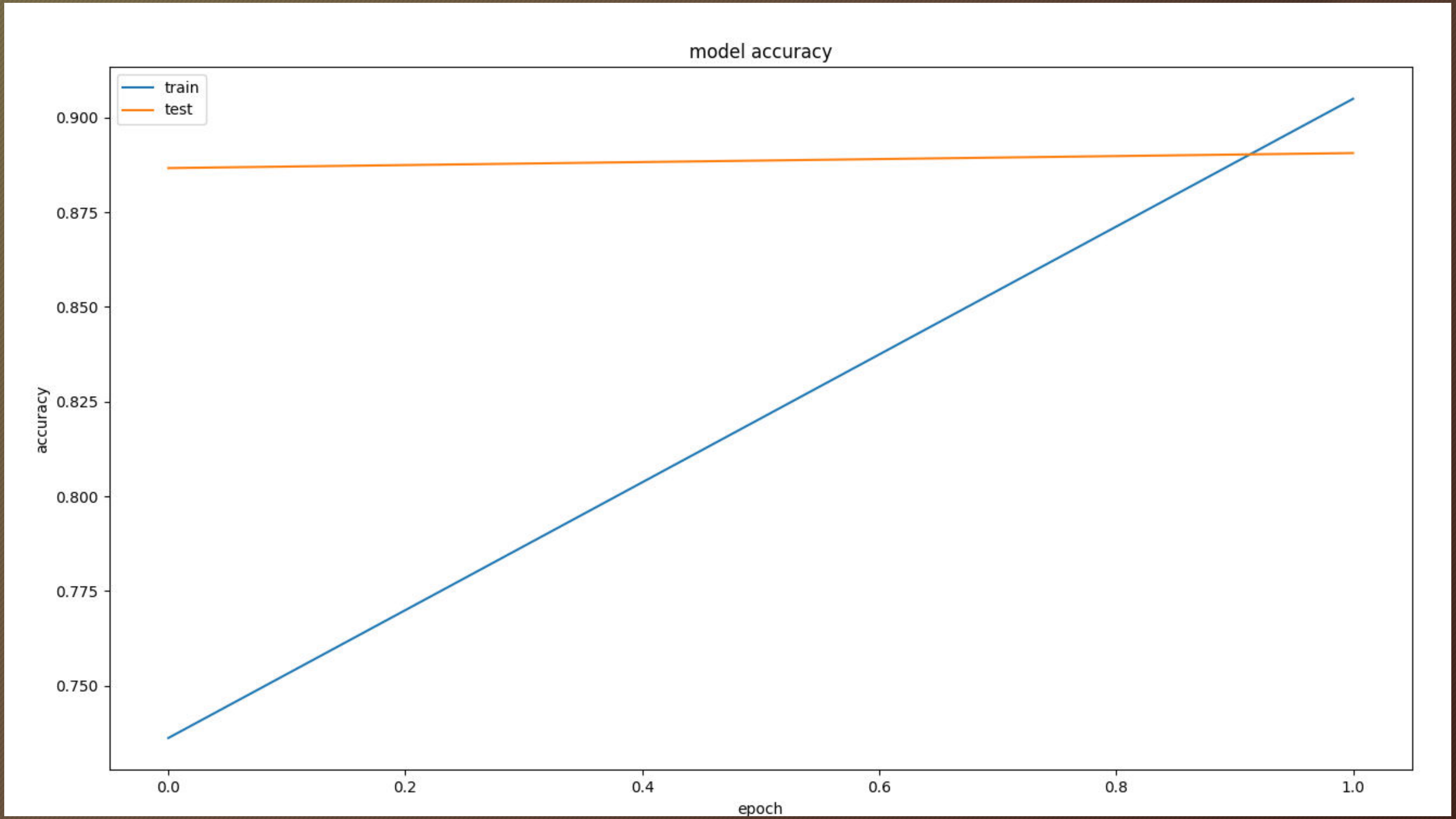
Output

```
nikhil@nikhil-SVE14A15FNW: ~/Desktop/keras-master
nikhil@nikhil-SVE14A15FNW:~$ cd Desktop
nikhil@nikhil-SVE14A15FNW:~/Desktop$ cd keras-master
nikhil@nikhil-SVE14A15FNW:~/Desktop/keras-master$ sudo python conv.py
[sudo] password for nikhil:
Using TensorFlow backend.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 32)           160000
_________________________________________________________________
dropout_1 (Dropout)          (None, 500, 32)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 500, 32)           3104
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 250, 32)           0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8000)              0
_________________________________________________________________
dense_1 (Dense)              (None, 250)               2000250
_________________________________________________________________
dropout_2 (Dropout)          (None, 250)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 251
=================================================================
Total params: 2,163,605
Trainable params: 2,163,605
Non-trainable params: 0
_________________________________________________________________
None
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE3 instructions, but these are available on your machine and could spe
ed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available on your machine and could s
peed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these are available on your machine and could s
peed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but these are available on your machine and could spee
d up CPU computations.
76s - loss: 0.4685 - acc: 0.7402 - val_loss: 0.2727 - val_acc: 0.8878
Epoch 2/2
72s - loss: 0.2426 - acc: 0.9047 - val_loss: 0.2617 - val_acc: 0.8916
Accuracy: 89.16%
nikhil@nikhil-SVE14A15FNW:~/Desktop/keras-master$
```

Epoch Vs Error



Epoch Vs Accuracy



Future work

- Sarcasm Detection with our model
- Humor Detection with our model

References

- Dave, S. Lawrence, D. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings WWW 2003, 2003.
- Ye Yuan, You Zhou.Twitter Sentiment Analysis with Recursive Neural Networks.
- Cicero Nogueira dos Santos,Maira gatti.Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts
- Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T. and Huang, Dan and Ng, Andrew Y. and Potts, Christopher. Learning Word Vectors for Sentiment Analysis. Association for Computational Linguistics 2011.