DEEP LEARNING Assignment -3

Ramkiran – 17545, CDS

Project 3 – Natural Language Inference

Introduction

In this project SNLI data-set is used to form a deep learning model and a tfidf model for textual entailment. Textual entailment is a process in which we identify the relation between two statements namely premise and hypothesis. Premise is the input statement and hypothesis is the conclusion drawn from the statement. There are 3 different classes of output one can draw between these statements in a natural language inference namely contradiction, neutral, entailment.

Contradiction: When the hypothesis and premise contradict each other
Neutral: When the hypothesis and premise are unrelated
Entailment: When the hypothesis is the conclusion of premise

Data

SNLI Data-set and Usage:

This data-set contains 3 files for train, test and development (dev). In the current project train file is used for training and 'dev' file is used for validation. Test file is used for evaluating testing accuracy.

In each file (train,test,dev) columns 'sentence1', 'sentence2','gold_label' are taken for training. 'sentence1','sentence2' columns in the file are used for premise and hypothesis respectively. 'gold_label' column forms the output representation of the NLI problem.

Glove Vectors and Usage:

Glove vectors word embedding are used in these project for word representation. Glove vectors file are available with 6 Billion (6B) words. There are 4 different files each representing a word vector file of 6B with different vector length. These vector lengths comes in the size of 50,100,200,300.

Preprossessing:

Prepossessing includes 4 major operations in this project.

1. Tokenization: From the corpus of entire premise and hypothesis, vocabulary of the data set is formed. Each word in the corpus is tokenized..

2. Punctuation: All the punctuation are removed from the data set

3. Lemmatization: Lemmatization is the algorithmic process of determining the lemma of a word based on its intended meaning (Wikipedia). Words like 'good' and 'better' will be represented by the same word.

4. Stopword Removal: Words like the, a, etc. add very information while comparing premise and hypothesis and these words occur very frequently. These word are to be removed for proper processing.

After preprocessing 28863

TF-IDF Model

The tf–idf is the product of two statistics, *term frequency* and *inverse document frequency*. There are various ways for determining the exact values of both statistics. (source: Wiki)

Term frequency (source: Wiki)

In the case of the **term frequency** tf(*t*,*d*), the simplest choice is to use the *raw count* of a term in a document, i.e., the number of times that term *t* occurs in document *d*. If we denote the raw count by $f_{t,d}$, then the simplest tf scheme is tf(*t*,*d*) = $f_{t,d}$.

Inverse Document frequency (source: Wiki)

The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient)

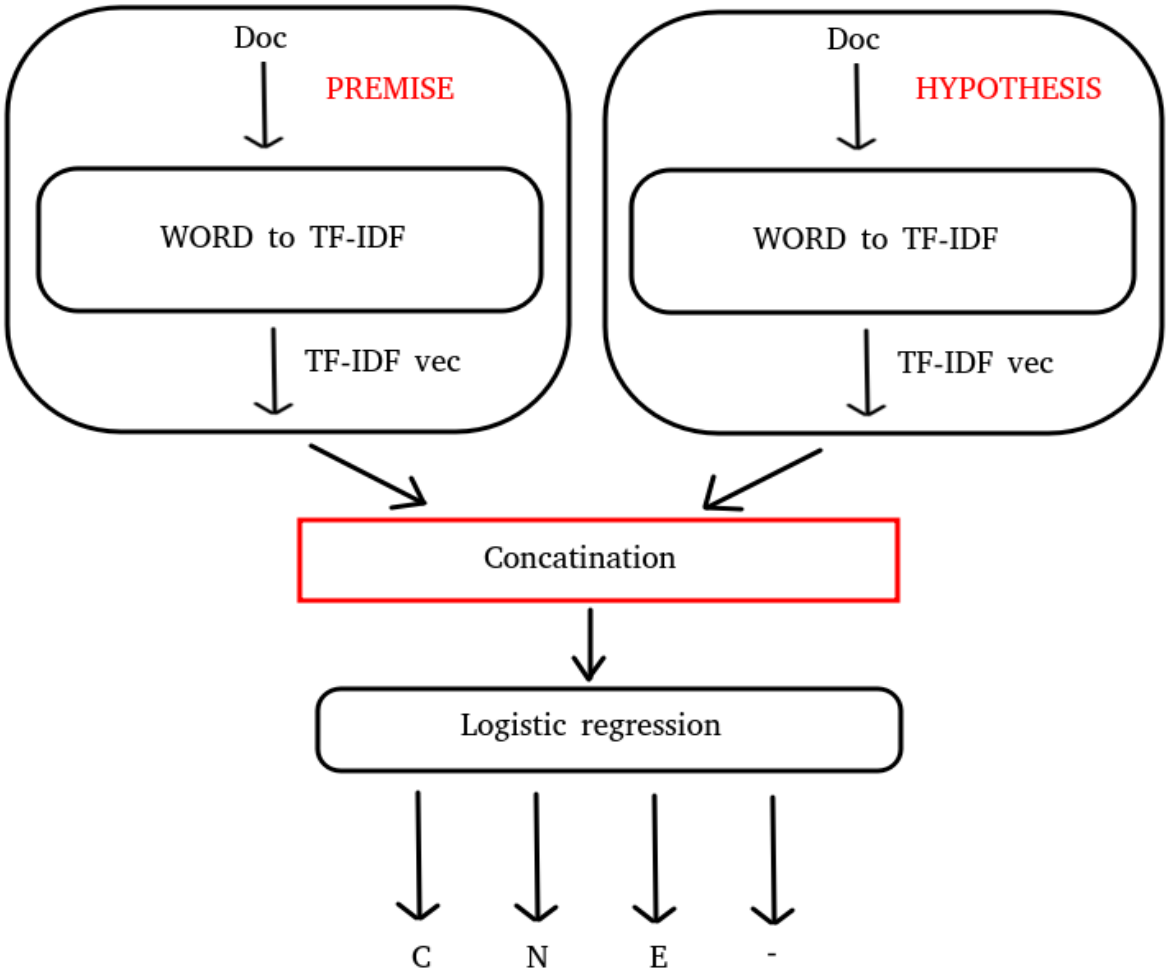

Figure 1: TF-IDF – Logistic Regression Architecture

Logistic Regression (With TF-IDF features)

From the corpus of premise and hypothesis, vocabulary for tf-ifd computation is formed. TF-IDF feature vector for each document is formed based this vocabulary. TF-IDF feature vector from prenise doc and hypothesis doc are concatenated and fed to multi-class logistic regression for training. The architecture of TF-IDF logistic regression is presented in figure 1. The results of the logistic regression are presented below.

	precision	recall	f1-score	support
-	0.00	0.00	0.00	176
contradiction	0.62	0.60	0.61	3237
entailment	0.62	0.69	0.65	3368
neutral	0.64	0.62	0.63	3219
accuracy			0.63	10000
macro avg	0.47	0.48	0.47	10000
weighted avg	0.62	0.63	0.62	10000

Accuracy of **63**% was reported using the Logistic Regression (With TF-IDF features).

Deep Learning Models

In deep learning models two different models with different hyper-parameters are modeled. These are some constraints while trying to model using deep learning model.

Constraints

1. **Memory**: When the entire data set (test) is used to train the model, because of huge data size and limited RAM capacity deep-model used to abruptly stop during a session. Hence to train the network a maximum of 1,50,000 of test sample are used to train the network. The size of test data is also dependent on the glove vector size used in the model. Also google colab is used to train the model as it has huge RAM.

2. **Time**: As the model complexity of the network increased time taken for each epoch has also increased resulting in a lesser number of experiments (8). Time taken for each epoch ~ 300 to 700 seconds.

3. **Colab**: Google colab closes the session if the session is ideal for some time. Program used to exit if some operation on the code is not performed for some time.

Deep – Learning Model 1:

Drawing inspiration from the above architecture a deep learning model was developed. Premise and hypothesis sentences are converted into a glove vector sequence and are padded with zeros to make the size of the sequence same for the RNN input. These glove vector sequences are concatenated and fed to a deep learning model (LSTM and Dense layer for training). Architecture of Model 1 is showed in the figure 2. Last output of Bi-LSTM is taken as input for dense nework.

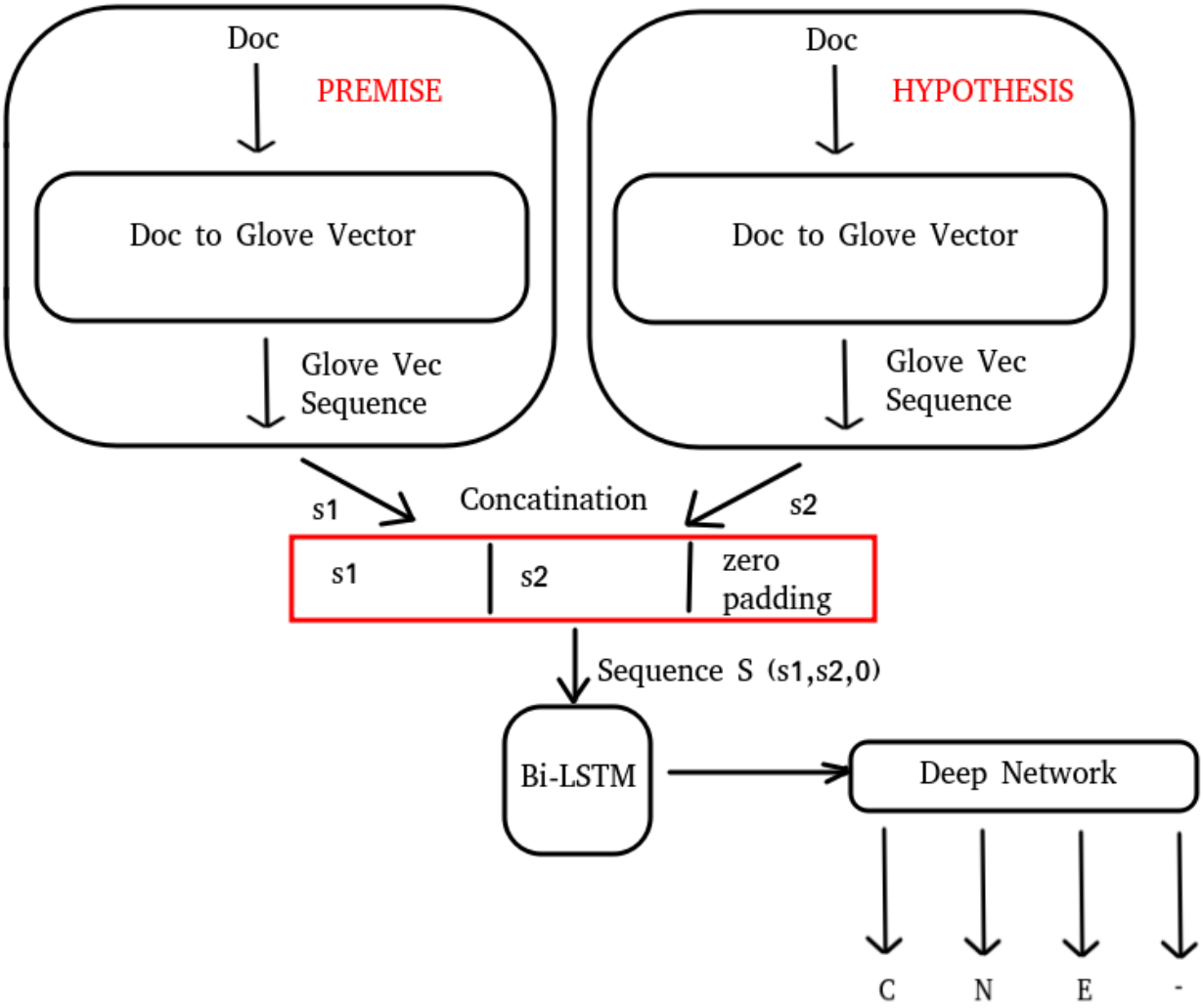

Figure 2: Deep Model -1 Architecture

With this model, after varying for different hyper parameters the maximum test accuracy obtained is **67%.** Figure 3 depicts the training process and compares training data against validation data.

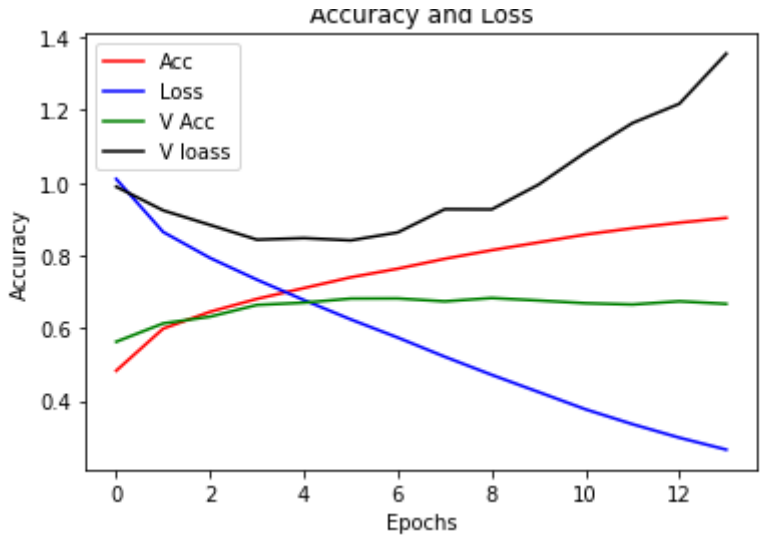

Figure 3: Train accuracy, loss vs Validation accuracy,loss. Early stopping is used to stop the validation error from further growing.

Model Description

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_2 (Bidirection multiple                  641600
_________________________________________________________________
dense_6 (Dense)              multiple                  12832
_________________________________________________________________
dense_7 (Dense)              multiple                  132
=================================================================
Total params: 654,564
Trainable params: 654,564
Non-trainable params: 0
```

Confusion Matrix

	precision	recall	f1-score	support
0	0.73	0.64	0.68	3237
1	0.67	0.77	0.71	3368
2	0.64	0.65	0.65	3219
3	0.00	0.00	0.00	176
accuracy			0.68	10000
macro avg	0.51	0.52	0.51	10000
weighted avg	0.67	0.68	0.67	10000

0 – Contradiction, 1 – Neutral, 2- Entailment, 3-None

There are two major problems in this network.

1. **Padding**: Padding done at the end of the LSTM Network is fed as input to the network which is wrong. To avoid this problem masking is introduced when there is padding.

2. **Concatenation**: Concatenation of both the vectors are done before passing it to LSTM. In the new architecture premise and hypothesis are passed to a LSTM layer to find their abstract representation and then concatenated to form the combined vector as presented in Figure 4.

Deep – Learning Model 2:

This model is inspired from the paper “A large annotated corpus for learning natural language inference”. Many variants are tried and two best performing models will be discussed.

Model 2.a: LSTM output at time instant of premise (or hypothesis) is summed up and then concatenated with hypothesis (or premise) LSTM output. The concatenated output fed to a dense network for classification.

Model 2.b: Last LSTM output of premise (or hypothesis) is concatenated with hypothesis (or premise) LSTM output. The concatenated output fed to a dense network for classification.

These models are described in the figure5.

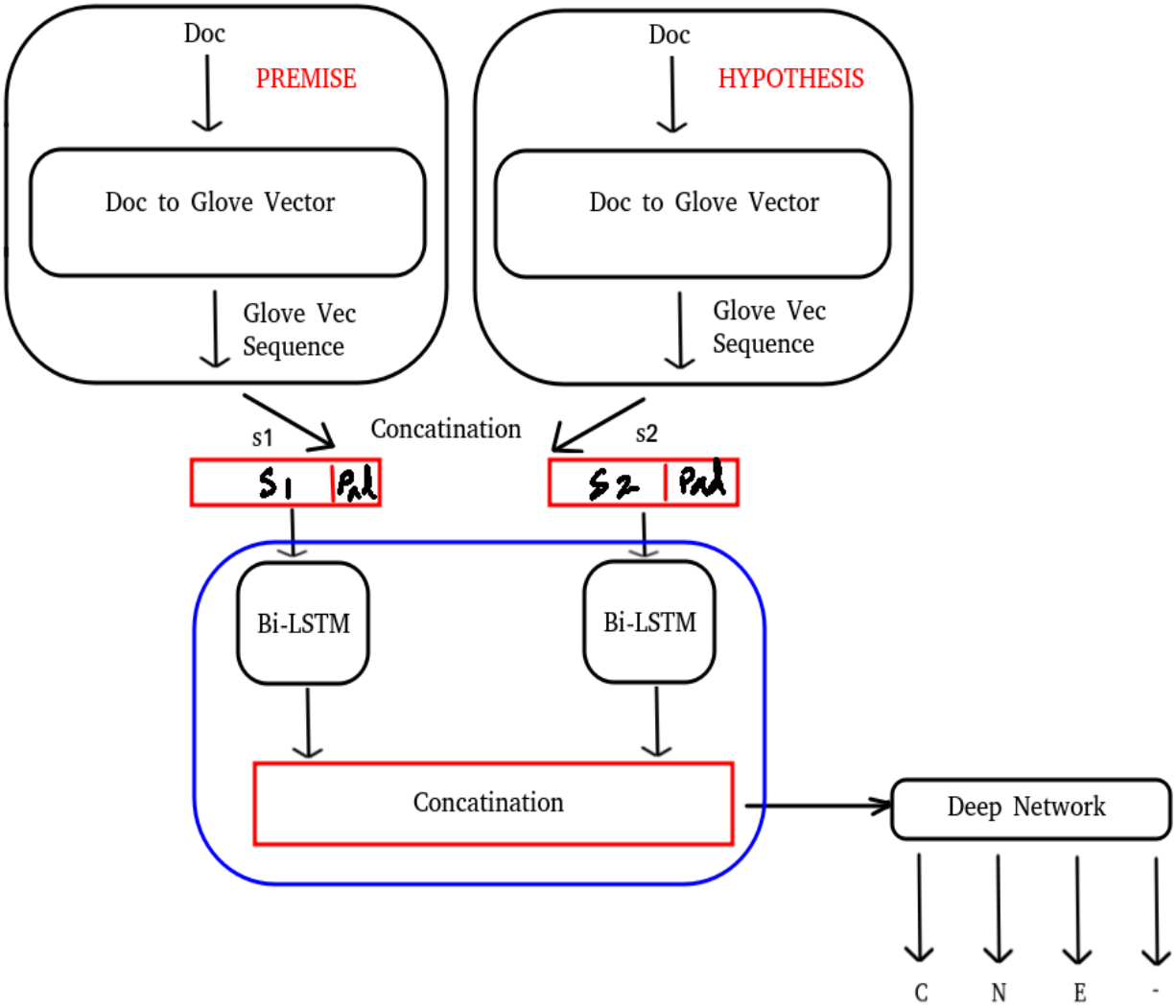

Figure 4: Deep Model -2 Architecture

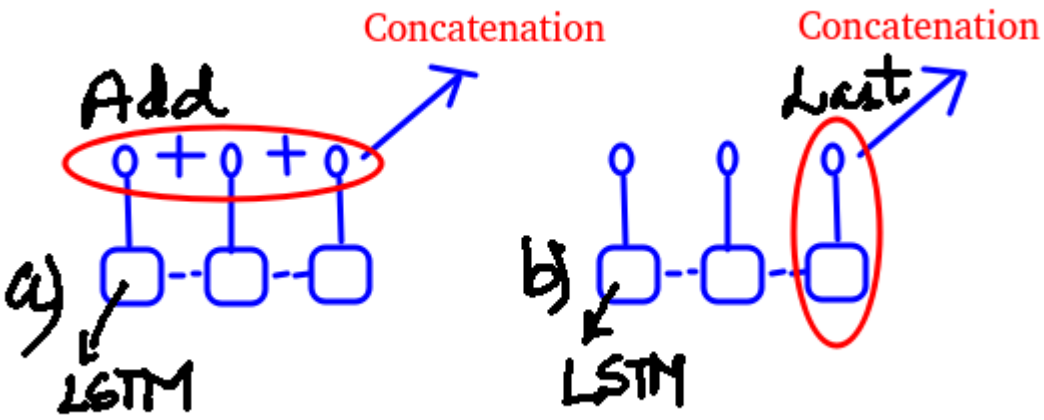

Figure 5: Two different types of LSTM Outputs that are tried before concatenation

Model For 2.b with glove vector size of 200

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 20, 200)]	0	
input_2 (InputLayer)	[(None, 20, 200)]	0	
masking (Masking)	(None, 20, 200)	0	input_1[0][0] input_2[0][0]
bidirectional (Bidirectional)	(None, 20, 512)	935936	masking[0][0] masking[1][0]
bidirectional_1 (Bidirectional)	(None, 20, 512)	1574912	bidirectional[0][0] bidirectional[1][0]
bidirectional_2 (Bidirectional)	(None, 512)	1574912	bidirectional_1[0][0] bidirectional_1[1][0]
concatenate (Concatenate)	(None, 1024)	0	bidirectional_2[0][0] bidirectional_2[1][0]
dense (Dense)	(None, 128)	131200	concatenate[0][0]
dropout (Dropout)	(None, 128)	0	dense[0][0] dense_1[0][0]
dense_1 (Dense)	(None, 128)	16512	dropout[0][0]
dense_2 (Dense)	(None, 4)	516	dropout[1][0]

Total params: 4,233,988
Trainable params: 4,233,988
Non-trainable params: 0

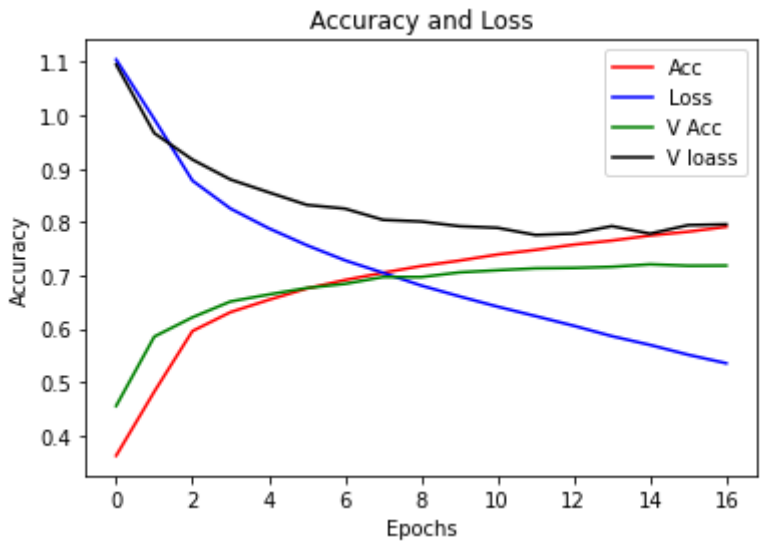

Figure 6: Train accuracy, loss vs Validation accuracy,loss. Early stopping is used to stop the validation error from further growing. Adam optimizer is used with Xavier uniform initialization.

Test Accuracy achieved with this model is **72%**. As per literature test accuracy achieved is close to 78%. Given sufficient memory this number can be achieved.

Confusion Matrix

	precision	recall	f1-score	support
0	0.75	0.71	0.73	3237
1	0.69	0.82	0.75	3368
2	0.71	0.65	0.68	3219
3	0.00	0.00	0.00	176
accuracy			0.71	10000
macro avg	0.54	0.54	0.54	10000
weighted avg	0.70	0.71	0.71	10000

Finally a table is presented of all the models that are tried.

Sl.no	Model	Glove Size	Accuracy(%)
1	TF-IDF Logistic regression	-	63
2	1-LSTM (200)-DNN(32-4) (Cancat-before)	200	68
3	1-RNN (200)-DNN(32-4) (Cancat-before)	200	65
4	1-RNN (200)-DNN(32-4) (Cancat-before)	300	66
5	Mask-1-LSTM(200)-DNN(32,4) (Model.a)	200	68
6	Mask-3-LSTM(256,256,256)-DNN(128,128,4) (Model.a)	200	70
7	Mask-3-LSTM(256,256,256)-DNN(128,128,4) (Model.b)	200	71