Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

Assignment 3

Table of Contents

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

Introduction

The purpose of this assignment is to create a recurrent neural network that can generate lyrics of a song based on provided melody and an initial word. We chose to use LSTM network that receives a sequence of word embeddings and melody features (using Pretty Midi package). The network learns to predict the embedding of the subsequent word for each word in the input. The use of word embeddings in both ends of the network allows the network to learn and produce words which are not part of the training set.

Dataset

The dataset used in this assignment includes 605 songs, 600 are the train set and 5 are the test set. For each song, the dataset provides the song name, artist, title, lyrics, and the melody of the song. There are 592 songs with a matching valid midi file of the train set and 5 songs in the test set. Therefore, the total train set size is 592 songs, and the test set size is 5 songs.

A validation set is created as 10% of the train set to better evaluate the model performances. Thus, the actual train set size is 533 songs, and the validation size is 59 songs.

Preprocess

The lyrics of every song were transformed into embeddings vectors of size 300 using GloVe word embeddings. Therefore, each song is represented by an array of size (lyrics length, 300). All the songs representations were padded to the max song lyrics length, which is 1481, for future process which reshaped all matrices into size (1481,300). Then, sequences of size sequence_length (=5) of word embeddings are created, resulting in a shape of (135694, 5, 300). In addition to the input matrices generation process there is another process of generating the “labels” matrices in the same manner which resulting matrices of shape (135694, 5, 300).

* 135694 is the number of words that was generated by the sequence generation process.

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

Midi features

Using Pretty_Midi package, every midi file is converted to a python object which is then used to extract specific features for each song. The midi features extracted are:

- Beat – estimation of the time the first beat starts (size=(1))
- Tempo – empirical estimate of the global tempo (size=(1))
- Semitone - relative amount of each semitone across the entire song (size=(12))
- Piano rolls – piano roll flattened across instruments (size=(128,fs*song_len[s])), fs is the sampling frequency (we used 8)

As requested, we implemented two methods of including these features as part of the model training.

First method

In the first method we used the first three features (Beat, Tempo, Semitone) constructed as a matrix of 14 features for every word in the sequence, resulting in matrix of shape (sequence_length, 14) that were added to the input. The midi features are the same for every word in the song. The purpose of this method is to summarize the melody and provide the model with the general characteristics of it.

Second method

In the second method we used all the extracted features, the first three used the same as in the first method and the piano roll is also added to the input. For every song, the piano roll was segmented by the song lyrics length. This resulted in segments of the piano roll which are assigned to the words in the song lyrics according to the word's position in the song. As part of the implementation the average piano roll segment is used to pad or trim every piano roll segments.

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

Network's architecture

The Network's architecture we used is based on LSTM, fully connected (dense), batch normalization and dropout layers. There are several inputs to the model, the lyrics, midi features and the piano rolls. Furthermore, the architecture of the model and the inputs change according to the midi feature inclusion method (mentioned above) used. The exact architecture of both models is presented in appendix A.

The lyrics input is in the shape of (batch_size(=128), sequence_length, word_embeddings_length). This input is then fed to an LSTM layer with sequence_length*(2^7) [=640] hidden units and an activation function of ReLu. This layer is followed by a fully connected layer. The midi features input is in the shape of (batch_size(=128), sequence_length, 14). Which is also followed by a fully connected layer. Both paths are joined using Concatenation layer into a single path.

For the second method there is another input in the shape of (batch_size(=128), sequence_length, 128, piano_roll_mean). This last 2 dimensions of the input are flattened, and the input is fed to another LSTM layer, now with 128 hidden units and an activation function of ReLu. This layer is also followed by a fully connected layer. In this method this path is joined with the above concatenation of the previous two paths. This concatenates all the inputs into one path.

In both methods the joined path is fed into a series of fully connected -> batch normalization -> dropout layers (this block repeated 3 times, fully connected size are 512, 512, 256). The last layer is again fully connected in the size of the word embeddings (300). The output activation is linear as the embeddings are real numerical values.

During the training phase we discovered that the models that were being generated overfitted the training data. While trying to overcome this issue we added Batch Normalization (BN) layers and a Dropout layer (rate=0.2). This addition prevented some of the overfitting, but the model still overfits the data to a certain extent and have difficulty generalizing to the data. The values for the regularization layers that we added (BNN and Dropout) were chosen because they seemed to be the most beneficial based on a few preliminary experiments we conducted.

Another way that we combat the overfitting of the model to the training data is by predicting the next word’s embedding. The use of word embedding in both ends of the network allows the network to learn and produce words which are not part of the training set. Not limiting the model to the range of the previously seen words allows it to better generalize and predict based on the ‘essence’ of the words.

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

Prediction Process

The model is generating word embeddings predictions (vector) of size 300. These predictions are the representation of the next word based on the model learned knowledge. To produce words from those embeddings a conversion must be applied. We utilize the cosine similarity measure and a form of KNN selection to accomplish that.

The cosine similarity is measured between every word embedding in the GloVe dictionary to the prediction vector. A fixed size min heap (sorted by the similarity) is used to store the top most similar words to the prediction vector. Doing so enables us to maintain a short list of the top candidates for selection. The size of the candidate heap is set to 30 as it produced diversified and focused words selections.

The candidates and their similarities are used in a subsequent selection process to product a single word prediction. The similarities are transformed to the range [0,1] using softmax. The chosen word is selected using probabilistic method based on the transformed values. Thus, words with higher probability have higher chance to be selected.

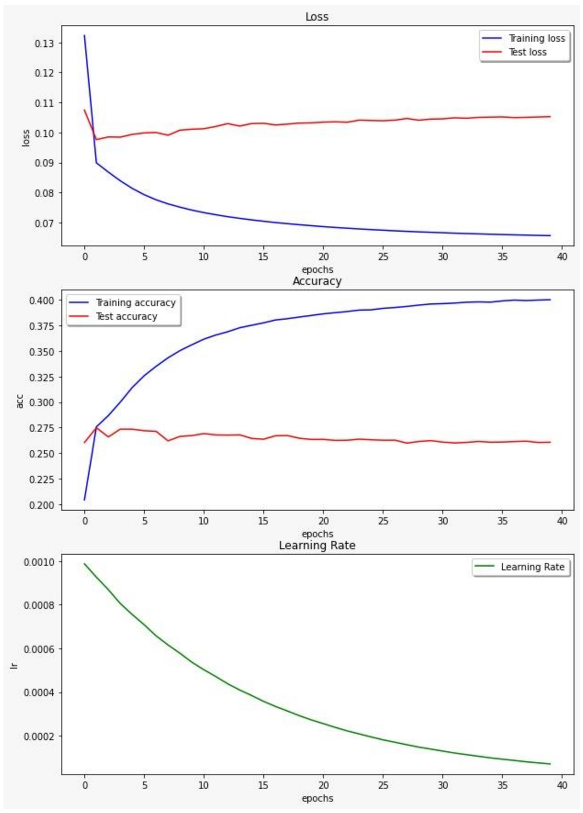
In the beginning of each sequence a single word is inserted to model with its midi features. The word is converted to its embedded form and zero padded to match the model's expected sequence length. Each predicted word is pushed into a queue of size (sequence_length) which is fed back to the model to generate the next prediction. Additionally, the predicted words are inserted to a list that represents the predicted song lyrics.

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

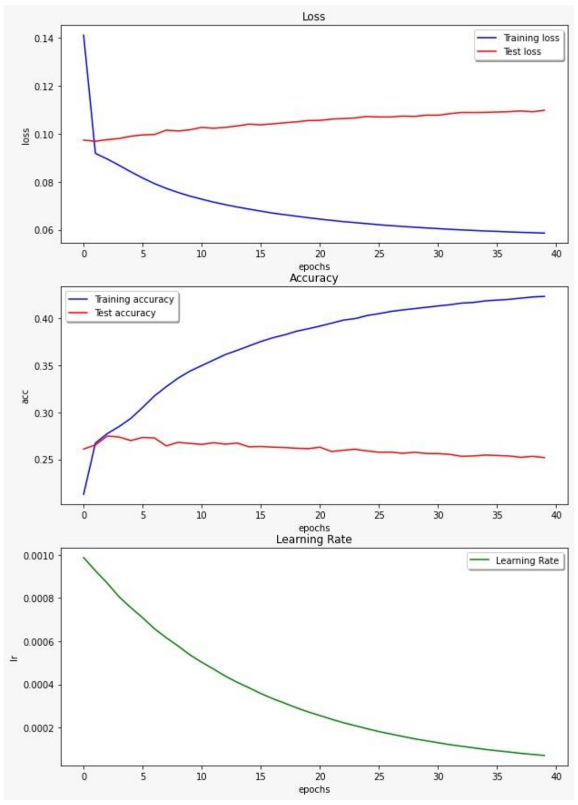
Performance

The model is fitted on the training data over 40 epochs, we used exponential learning rate decay with starting value of 0.001, number of decay steps 1000 and decay rate of 0.95. As mentioned above the model is overfitted to the training data as represented in the graphs below:

<u>Method 1</u>



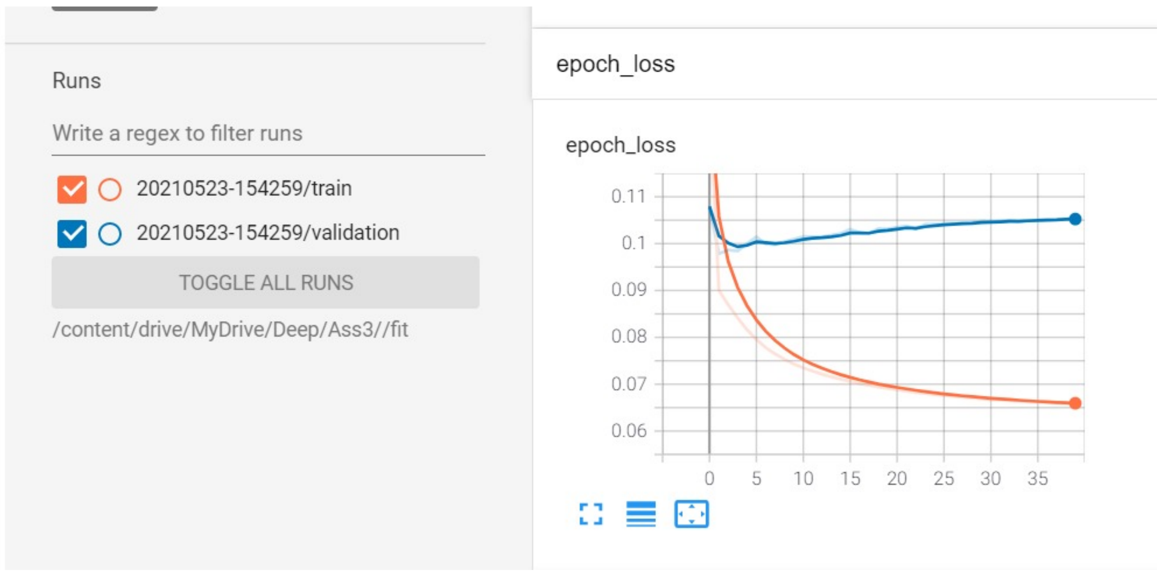
<u>Method 2</u>



The meaning of the overfitting is that the model is able to recognize the connections between the words in the training set but is limited in generalizing them.

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

Tensor Board



Runs

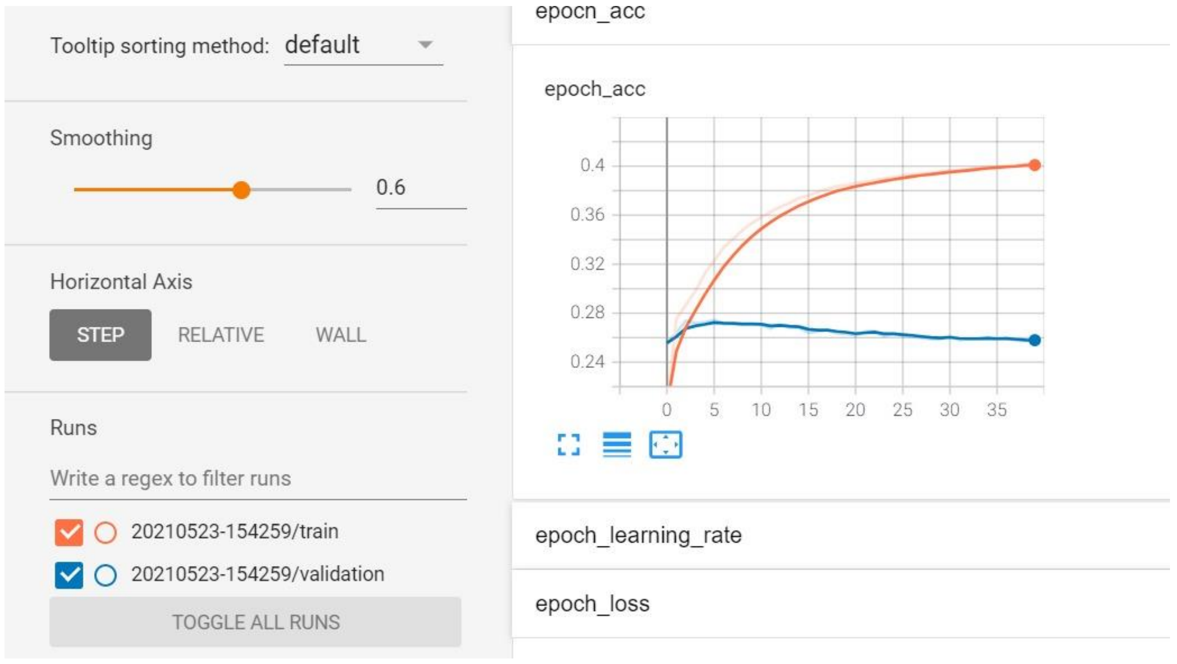
Write a regex to filter runs

☑ 20210523-154259/train
☑ 20210523-154259/validation

TOGGLE ALL RUNS

/content/drive/MyDrive/Deep/Ass3//fit

epoch_loss



Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

[✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

☑ 20210523-154259/train
☑ 20210523-154259/validation

TOGGLE ALL RUNS

epocn_acc

epoch_learning_rate

epoch_loss

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

Results

We conducted 10 different tests, 2 tests for each song in the test set using the two different architectures. The first word of each song is fed to the model with its corresponding midi features (with/without the piano roll). A limit to the length of the generated song was set as the average length of the training set songs lyrics divided by 10 (for computational resources restrictions).

As part of our experiments the first sentence that was generated is the following amazing lyrics:

Banana that you n't

The results are as follows:

Song name	Method 1 lyrics	Method 2 lyrics
Eternal flame	close could what though think as did let that one we must one once some one at be think where because then that you'll it again 're can if n't you	close will but you so too really n't might think just wish as what 're really thought it we but you it say there undinal going up of going only well
Honesty	if sure where they so what when 'm again obviosuly they was much they just do as but it could we if it along also were should does n't something someone	if if they should that let just take that still it has still it because say want give if could 'll could just able it there still it never still but
Lovefool	dear there are would 'll should my when again so very very far we though what so want going could not would sure thought if because what nothing well 'm put	dear there still though even so so something you get n't though know how because there there actually don39t be because because i might would make i just let when you
Barbie girl	hiya .... but always which so gonna need your think will that just there even going take could would it want both just her well but which one want to there	hiya why why going not even that it there even though when where did because what n't n't will have what me say can 'll everybody really go down well it
All the small things	all but could even should would might say not what even what they if can got that him nothing well even if	all have we but 'll really just couldnât only another n't always even when know just that what probably not not

	if i'm sittin' them does some indeed even nothing	though would put after some wonder even get would once

For the second stage of our experiments, we selected 3 initial words: ‘banana’, ‘world’, ‘air’. These words were fed to the model with the test set melodies.

Song name	Method 1 lyrics	Method 2 lyrics
Eternal flame	Banana so up place that there actually he someone they did it that ddont knowing it think so i just fly.i them 've i n't dont that they you won?t tell	Banana so though but you think just remember did so i did want want them off where need be can there still thought amarillo's want to because i?m never going to
Eternal flame	World just you can them it 't want help do 're just but when never they sure still do could do even never but sure does so might we n't ya	World thing because never they though thought n't would been but if they got would sure though could not we never need but so sure also but if 'm could we
Eternal flame	Air that thing now i'm actually n't say why would but really actually he's when because could but interested.i while there never did so perhaps really n't wont to must it	Air there i might sure want go they sure should because because even thing they gonna sure how can really you let him take something let too but maybe know but
Honesty	Banana though could there have well should well actually what n't it something but just so though would them need you still should not anyone make that worry okay you'er actually	Banana it thought it so really know what though n't could want you you turned take just out you once i so want should because i you just could n't think
Honesty	World when only you n't again know one because want we they do can they not though but despair things so might did would we have when get they they you	World because that what want has never because it never sure should know because they come there we when could it get but n't when 've have something we be another's
Honesty	Air when want so make n't tell them if 'll 'll say always think same could you can	Air they we so 'll they but say what we i 'll dont might would so could must have

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

	just put n't you it going up when if have it same there	did when if really might should should i 'll do even they
Lovefool	Banana well what the way n't need not sure you though what if really just they actually dudududuudu we'll will would not them them one same i don.t think it that	Banana could what actually even just this same what be did they we still going down though need will know guess but would must always they let me would 'll n't
Lovefool	World then so could should because i tickin' as not even would him there going we already you that aint going here think where would want so too much only but	World 'll even well did when maybe it just the we i it like!i only just though could because you could do it could but even n't not because they did
Lovefool	Air could need this never actually aunt's what when going put if because that thing as same could because it 'dro but up back could not i we able take would	Air even want way out not it even still what because also probably have that feel where just want we get one too way know going n't just really n't because
Barbie girl	Banana being which enough still n't there not something it he preachin' there as even it it have even well we so never just really guess sure i would actually think	Banana sure it even think they really want which have n't because think there been know one because it actually think just something not well you came both if when 'd
Barbie girl	World can they tell my could just n't them even could should because that iõm you come way that actually ddont it actually did there well though actually n’t get there	World not when did you so say can want that though n't not so way as thought only thought there even you even they there was first time even just another
Barbie girl	Air would but then again still it not only that 'll n't again need it sure could because so even that if could when have but it even but we can	Air there that not another take make having so n't able be have what sure what n't have should should should they would that even sure will you go them that

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

All the small things	Banana really because did get both as even we even up still put it well have too quite because can always get let it they because actually awesome.if fly.i sure it	Banana up some love out just going while when but but when sure could want there they came we thought deffinantly you why hyave i knew think sure but how that
All the small things	World when we so thing you want because sure make n't we them would that me never you we there well even that is nothing you so 'll might would sure	World know n't just if even just this one would but 're really never want bring still think if when got that because that defanitly going when it’s even that.i every
All the small things	Air the they can when up like my most way then make n't make get could so they should have though could should it sure only when had there just that	Air if that 'll it going way but äôm n't really could have even but same so what if gotta must could have knew i that really but if we might

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

Initial Word Effect

After conducting the experiments above, the results show the effect of the first word fed to the model. The model is struggling with correctly identifying the meaning of the input words and so is struggling to predict the next words. Therefore, the model generates somewhat random patterns of words. This is due to the word embedding predictions aren't good enough, and so the similarity measure calculated is not effective enough in producing good word selections.

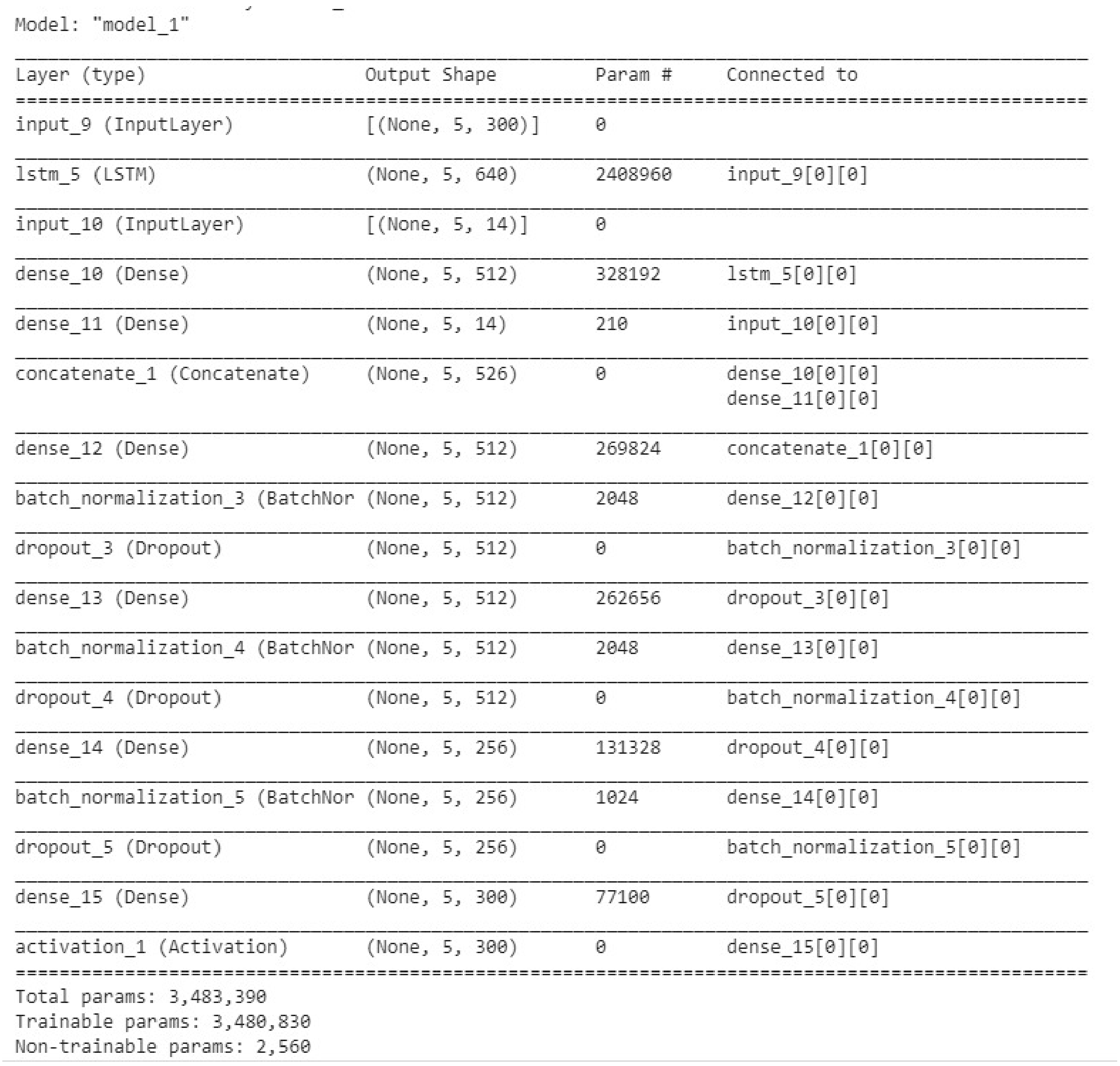
Possible improvements

To combat the overfitting and misclassification we would suggest more testing of various network architectures. Possibly differently sized networks with additional Regularization techniques. Moreover, we suggest a more training data to combat the generalization challenge.

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

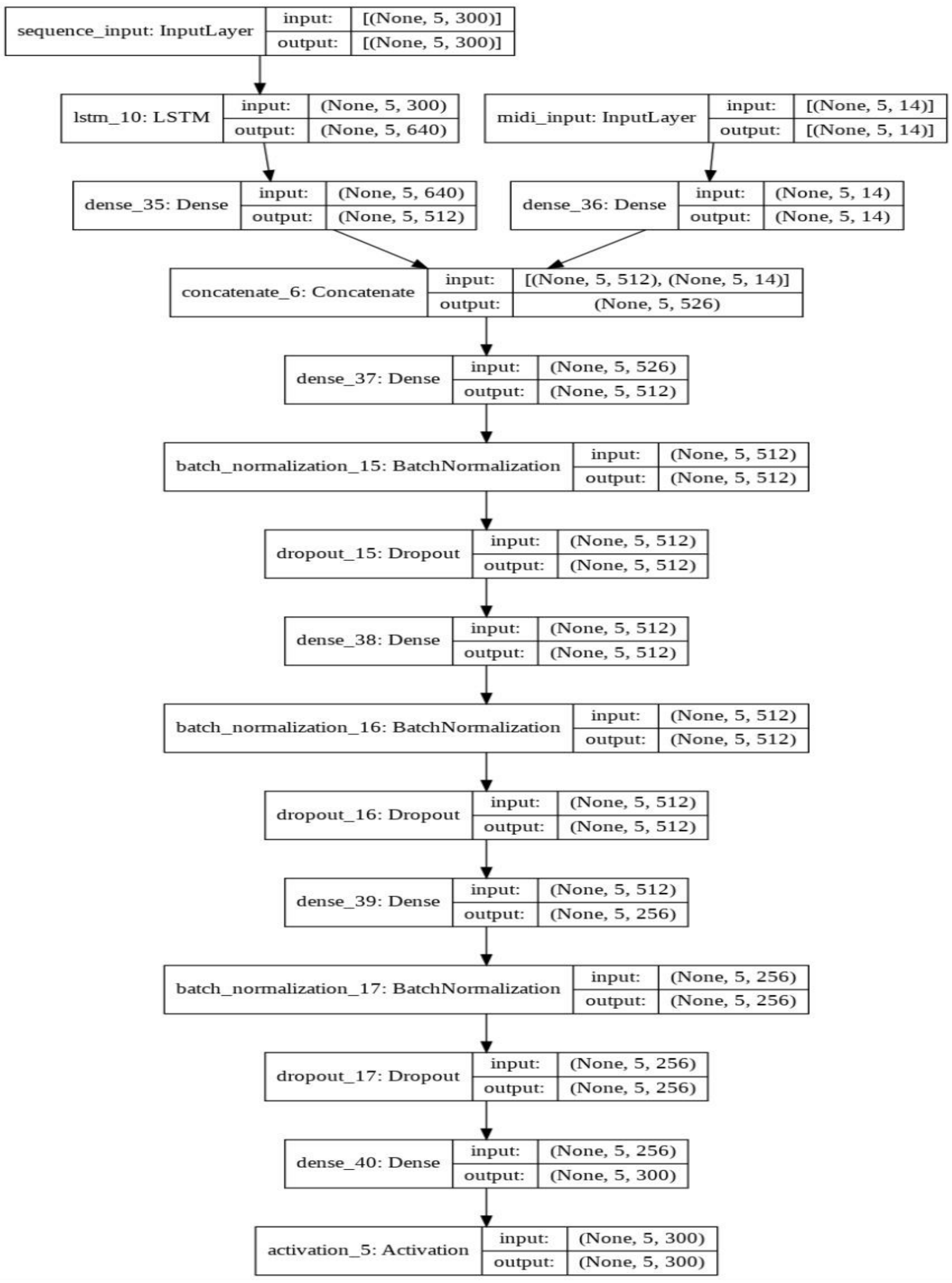
Appendix A

Architecture method 1



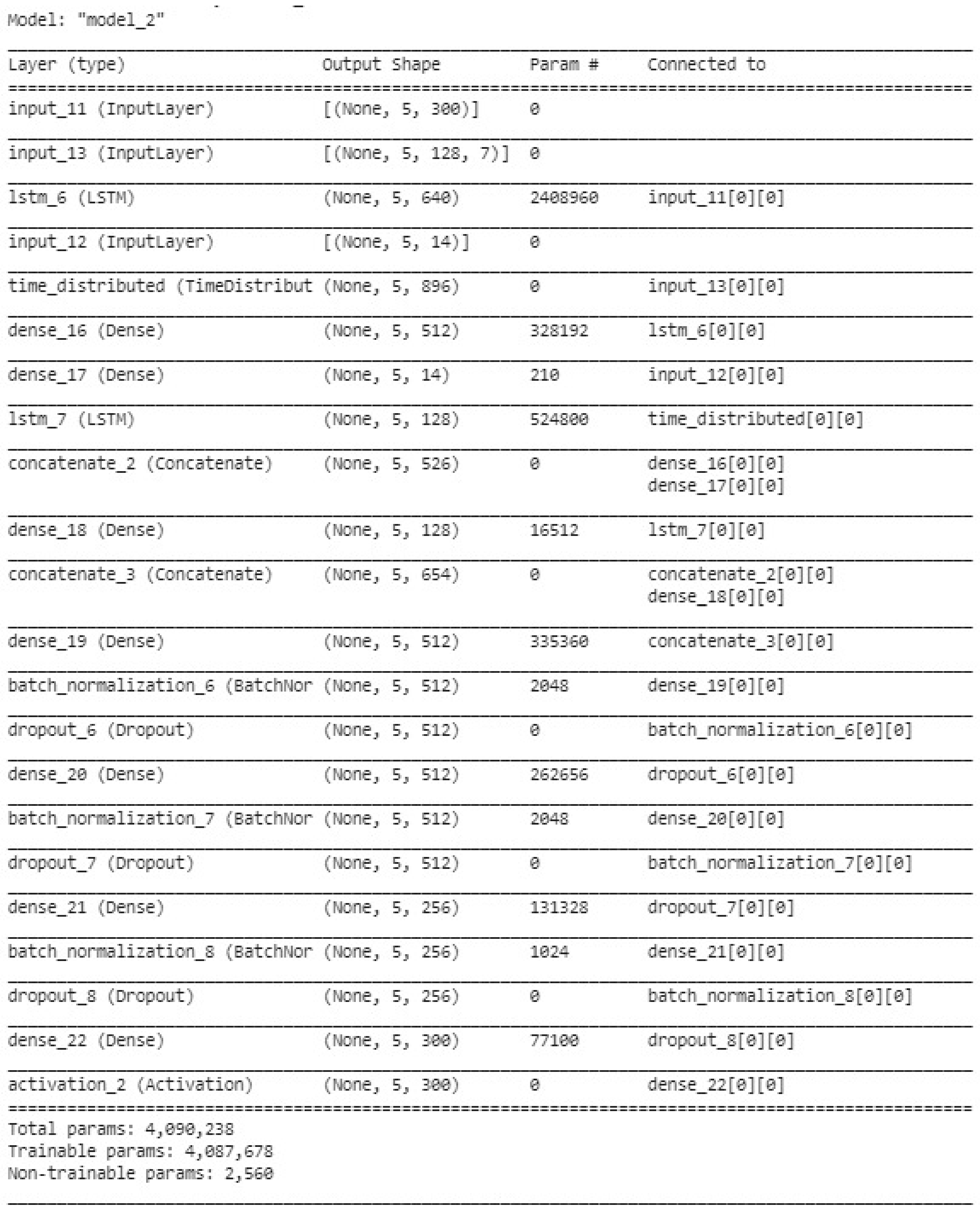
```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_9 (InputLayer)            [(None, 5, 300)]     0
__________________________________________________________________________________________________
lstm_5 (LSTM)                   (None, 5, 640)       2408960     input_9[0][0]
__________________________________________________________________________________________________
input_10 (InputLayer)           [(None, 5, 14)]      0
__________________________________________________________________________________________________
dense_10 (Dense)                (None, 5, 512)       328192      lstm_5[0][0]
__________________________________________________________________________________________________
dense_11 (Dense)                (None, 5, 14)        210         input_10[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 5, 526)       0           dense_10[0][0]
                                                                 dense_11[0][0]
__________________________________________________________________________________________________
dense_12 (Dense)                (None, 5, 512)       269824      concatenate_1[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 5, 512)       2048        dense_12[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 5, 512)       0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
dense_13 (Dense)                (None, 5, 512)       262656      dropout_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 5, 512)       2048        dense_13[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 5, 512)       0           batch_normalization_4[0][0]
__________________________________________________________________________________________________
dense_14 (Dense)                (None, 5, 256)       131328      dropout_4[0][0]
__________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 5, 256)       1024        dense_14[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 5, 256)       0           batch_normalization_5[0][0]
__________________________________________________________________________________________________
dense_15 (Dense)                (None, 5, 300)       77100       dropout_5[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 5, 300)       0           dense_15[0][0]
==================================================================================================
Total params: 3,483,390
Trainable params: 3,480,830
Non-trainable params: 2,560
```

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268



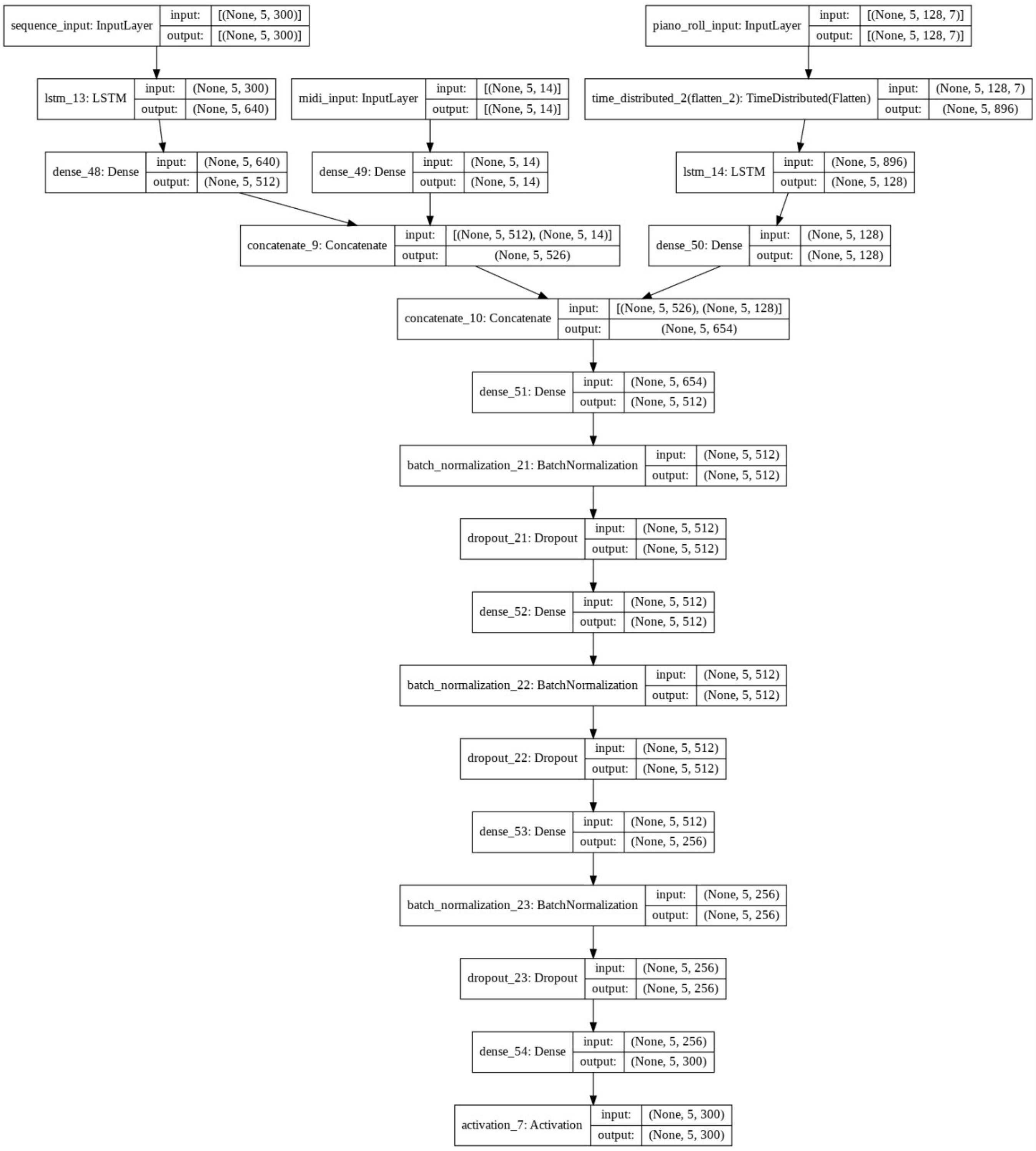
Yiftach Savransky – 312141369
Gal Rosenthal – 312585268

Architecture method 2



```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_11 (InputLayer)           [(None, 5, 300)]     0
__________________________________________________________________________________________________
input_13 (InputLayer)           [(None, 5, 128, 7)]  0
__________________________________________________________________________________________________
lstm_6 (LSTM)                   (None, 5, 640)       2408960     input_11[0][0]
__________________________________________________________________________________________________
input_12 (InputLayer)           [(None, 5, 14)]      0
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, 5, 896)       0           input_13[0][0]
__________________________________________________________________________________________________
dense_16 (Dense)                (None, 5, 512)       328192      lstm_6[0][0]
__________________________________________________________________________________________________
dense_17 (Dense)                (None, 5, 14)        210         input_12[0][0]
__________________________________________________________________________________________________
lstm_7 (LSTM)                   (None, 5, 128)       524800      time_distributed[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 5, 526)       0           dense_16[0][0]
                                                                 dense_17[0][0]
__________________________________________________________________________________________________
dense_18 (Dense)                (None, 5, 128)       16512       lstm_7[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 5, 654)       0           concatenate_2[0][0]
                                                                 dense_18[0][0]
__________________________________________________________________________________________________
dense_19 (Dense)                (None, 5, 512)       335360      concatenate_3[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 5, 512)       2048        dense_19[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 5, 512)       0           batch_normalization_6[0][0]
__________________________________________________________________________________________________
dense_20 (Dense)                (None, 5, 512)       262656      dropout_6[0][0]
__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 5, 512)       2048        dense_20[0][0]
__________________________________________________________________________________________________
dropout_7 (Dropout)             (None, 5, 512)       0           batch_normalization_7[0][0]
__________________________________________________________________________________________________
dense_21 (Dense)                (None, 5, 256)       131328      dropout_7[0][0]
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 5, 256)       1024        dense_21[0][0]
__________________________________________________________________________________________________
dropout_8 (Dropout)             (None, 5, 256)       0           batch_normalization_8[0][0]
__________________________________________________________________________________________________
dense_22 (Dense)                (None, 5, 300)       77100       dropout_8[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 5, 300)       0           dense_22[0][0]
==================================================================================================
Total params: 4,090,238
Trainable params: 4,087,678
Non-trainable params: 2,560
__________________________________________________________________________________________________
```

Yiftach Savransky – 312141369
Gal Rosenthal – 312585268


sequence_input: InputLayer | input: [(None, 5, 300)] | output: [(None, 5, 300)]
piano_roll_input: InputLayer | input: [(None, 5, 128, 7)] | output: [(None, 5, 128, 7)]
lstm_13: LSTM | input: (None, 5, 300) | output: (None, 5, 640)
midi_input: InputLayer | input: [(None, 5, 14)] | output: [(None, 5, 14)]
time_distributed_2(flatten_2): TimeDistributed(Flatten) | input: (None, 5, 128, 7) | output: (None, 5, 896)
dense_48: Dense | input: (None, 5, 640) | output: (None, 5, 512)
dense_49: Dense | input: (None, 5, 14) | output: (None, 5, 14)
lstm_14: LSTM | input: (None, 5, 896) | output: (None, 5, 128)
concatenate_9: Concatenate | input: [(None, 5, 512), (None, 5, 14)] | output: (None, 5, 526)
dense_50: Dense | input: (None, 5, 128) | output: (None, 5, 128)
concatenate_10: Concatenate | input: [(None, 5, 526), (None, 5, 128)] | output: (None, 5, 654)
dense_51: Dense | input: (None, 5, 654) | output: (None, 5, 512)
batch_normalization_21: BatchNormalization | input: (None, 5, 512) | output: (None, 5, 512)
dropout_21: Dropout | input: (None, 5, 512) | output: (None, 5, 512)
dense_52: Dense | input: (None, 5, 512) | output: (None, 5, 512)
batch_normalization_22: BatchNormalization | input: (None, 5, 512) | output: (None, 5, 512)
dropout_22: Dropout | input: (None, 5, 512) | output: (None, 5, 512)
dense_53: Dense | input: (None, 5, 512) | output: (None, 5, 256)
batch_normalization_23: BatchNormalization | input: (None, 5, 256) | output: (None, 5, 256)
dropout_23: Dropout | input: (None, 5, 256) | output: (None, 5, 256)
dense_54: Dense | input: (None, 5, 256) | output: (None, 5, 300)
activation_7: Activation | input: (None, 5, 300) | output: (None, 5, 300)