# AI Project 2: Multi-label Classification on YouTube-8M


**KAR CHUN TEONG**
518030990014
Department of Computer Science
Shanghai Jiao Tong University
teongkarchun@sjtu.edu.cn

**MATSUNAGA TAKEHIRO**
518030990028
Department of Computer Science
Shanghai Jiao Tong University
matsunagatakehiro@sjtu.edu.cn

**EDUARDO WANG ZHENG**
518030990025
Department of Computer Science
Shanghai Jiao Tong University
eduardowangzheng@sjtu.edu.cn


## Abstract


The authors develop two models, which are Deep Fully Connected Neural Networks with Merge-and-Run Mappings(DMRNets) and long short-term memory(LSTM), to achieve multi-label classification on the YouTube-8M dataset.


## 1 Introduction

This project is the group project of CS410 Artificial Intelligence 2020 Fall in Shanghai Jiao Tong University named Multi-label Classification on YouTube-8M. The main purpose is to use a neural network model taking two video-level features(mean_rgb and mean_audio) as input, to output the prediction of labels of videos from the YouTube-8M dataset. The authors develop two different neural network models, which are Deep Fully Connected Neural Networks with Merge-and-Run Mappings(DMRNets) and LSTM in their attempts to achieve this goal.

## 2 Methods

### 2.1 Deep Fully Connected Neural Networks with Merge-and-Run Mappings(DMRNets)

#### 2.1.1 Model Overview

One of the neural network architecture we consider is the deep fully connected neural networks with merge-and-run mappings, this is actually a variation of the deep convolutional neural networks with merge-and-run mappings, which coincidentally has the same short-form as DMRNets, to differentiate these two architectures, we would use DMRNets(FCN) as the short-form of Deep fully connected neural networks with Merge-and-Run Mappings and DMRNets(CNN) as the short-form of Deep convolutional neural networks with Merge-and-Run Mappings. DMRNets(CNN) itself is a variation of the residual neural network (ResNet).

ResNet originated from Deep Convolutional Neural Networks(CNN), researchers have been pondering at the question of, does stacking more convolutional layers (which is increasing the depth) leads to better performance? They found out that stacking layers leads to a degradation problem, as they increase the depth, accuracy get saturated then degrades rapidly, and this is not caused by overfitting, the more layers they stack, the higher the training error. Some proposed a solution which is residual mapping, the basic building block of residual mapping is shown in Figure 1. Denote the desired underlying mapping as $H(x) = F(x) + x$, instead of directly fitting to $H(x)$, fit the layer to $F(x)$, and fit another layer to the identity mapping which is $x$, at the end merge the two layers to get the desired mapping $H(x)$. The authors of the article argued and proved that this modification significantly reduced the degradation problem while increasing the depth of the model, for more information please refer to the original article[1].

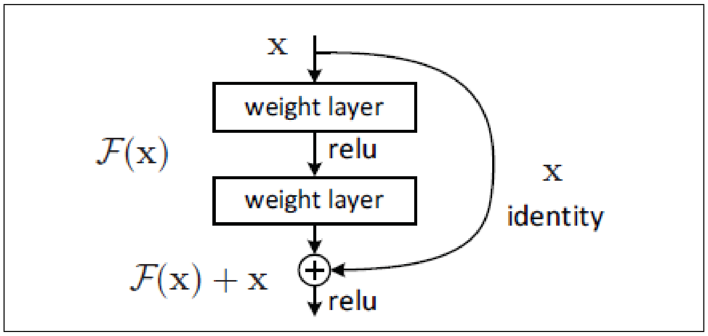


Figure 1: Residual Learning: a building block.

Next, we move on to the variation based on ResNet, the DMRNets(CNN). The idea behind this architecture is to assemble the residual branches in parallel through a merge-and run mapping. Merge means to average the inputs of these residual branches, and by Run it means to add the average to the output of each residual branch as the input of the subsequent residual branch. To further illustrate the model, please refer to Figure 2. Figure 2(a) shows 2 residual branches assembled sequentially, Figure 2(b) shows 2 residual branches assembled in parallel and with identity mappings, and finally Figure 2(c) shows 2 residual branches assembled in parallel and with the proposed merge-and-run mappings. The authors of the article argued and proved that this modification further reduced the training difficulty of ResNet and retained the advantages of ResNet, for more information please refer to the original article[2].

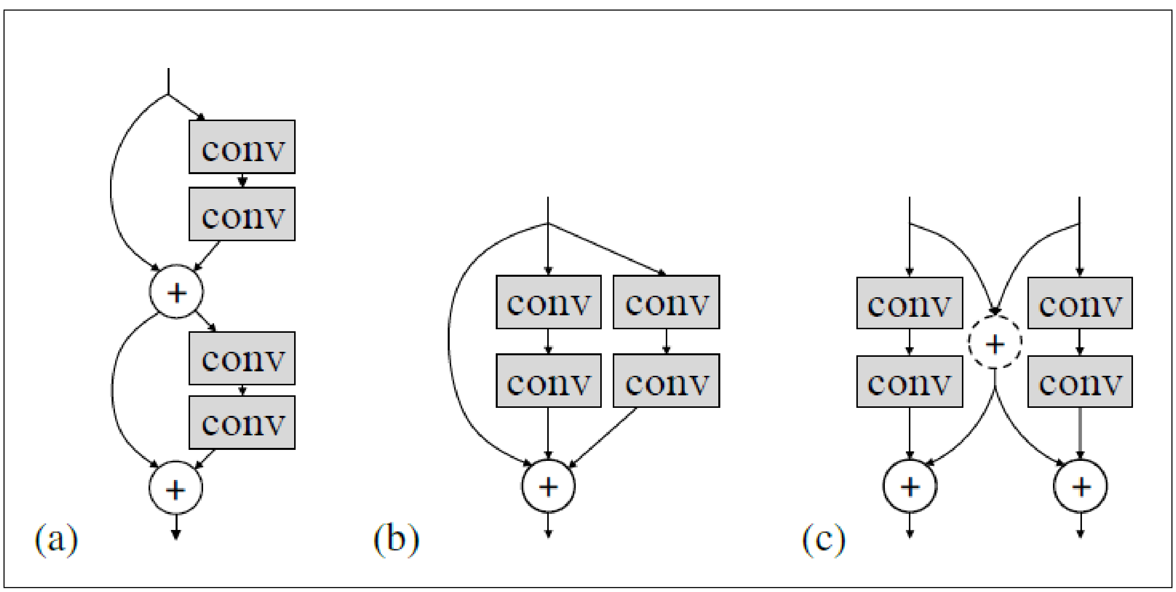


Figure 2: Comparison between different assembly blocks.

In this project, we adopt and do some variations on the DMRNets(CNN) by using the fully connected layers instead of convolutional layers as the basic layers to form the DMRNets(FCN). Part of the model structure is shown in Figure 3. First we define a fc_block which consists of dense layer, batch normalization layer, leaky ReLU layer, and dropout layer. Then we define the merge-and-run function, which separate input into 2 branches, branch a pass through the fc_block while branch b pass through identity mapping, which is essentially mapping to itself, then we pass both branches into an average layer, then the output passes through another leaky ReLU layer. In Figure 3, we can see both input first go through a fc_block, then into a merge-and-run mappings. The model is actually so deep that the rest of the structure is too large to be shown in the report, but it's essentially the same process, repeatedly pass through fc_block and merge-and-run mappings and finally merged into a single output, and the full picture is included in the submission folder.

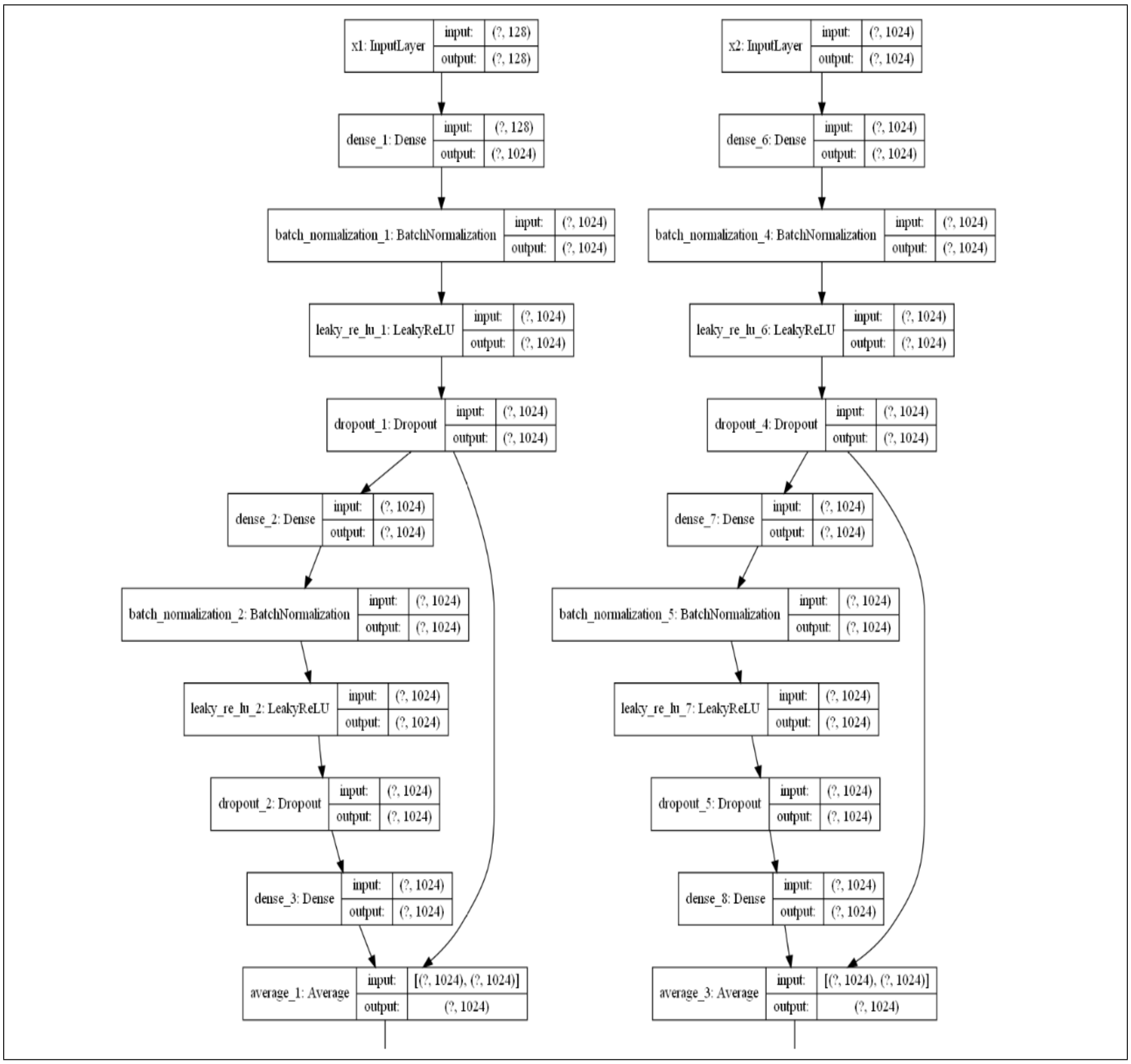


Figure 3: Part of the structure of DMRNets(FCN).

### 2.1.2 Performance

Small datasets MAP@10: Whole datasets MAP@10:

### 2.2 Long short-term memory (LSTM)

#### 2.2.1 Model Overview

LSTM is a type of recurrent neural network (RNN) architecture. A common LSTM unit is composed of a cell and 3 gates. The cell stores values over time intervals as cell states, and the 3 gates' main job is to regulate the flow of information into and out of the cell. Specifically, forget gate decides which information to forget, input gate decides to update which value of cell states and update cell states, and at last, the output gate output filtered cell states. To put it in an abstract high-level view, what LSTM do is using past context as reference to predict the output, for more information please refer to the original article[3]. The model structure is shown in Figure 4.

#### 2.2.2 Performance

Small datasets MAP@10: Whole datasets MAP@10:

### 2.3 Bidirectional Long short-term memory (biLSTM)

#### 2.3.1 Model Overview

biLSTM is a variant of LSTM, biLSTM train 2 instead of 1 LSTMs on 1 input sequence, the first on the input sequence, and the second on a reversed copy of the input sequence. In other words, it connects two hidden layers of opposite directions to the same output, this generally provide additional context to the network, thus results in better training results. for more information please refer to the original article[4]. The model structure is shown in Figure 5.

#### 2.3.2 Performance

Small datasets MAP@10: Whole datasets MAP@10:

## 3 Experiments

### 3.1 Training

We wanted to use cloud platforms to train our model, but most of them needs payment, and we don't have the fund for that, Google Cloud Platform offers 1 month free trial, but to get the free trial we need to register with credit card, which unfortunately we don't possess, at the end of the day we need to train the model using our local machines. Limited by the performance of our local machines, we could only use a small subset of the datasets for training, in this case, we use the first 200 training shards (train0000.tfrecord train0199.tfrecord) and the first 100 validation shards (validate0000.tfrecord validate0099.tfrecord) as our training datasets. Figure 6 and 7 shows the graph of training curve of the LSTM and biLSTM models. As shown in the figures, both models converged at around 0.75, with biLSTM having a slightly better performance.

### 3.2 Comparison and Analysis of Performance

Table 1: Performance of different models

| Model | 200 train MAP@10 | 100 validation MAP@10 | All datasets MAP@10 |
|---|---|---|---|
| DMRNets | 0 | 0 | 0 |
| LSTM | 0 | 0 | 0.759 |
| biLSTM | 0 | 0 | 0.763 |

The performances of 3 models are listed in Table 1, as we could see, the MAP@10 for the first 200 training shards and 100 validation shards are always higher than the MAP@10 for the whole datasets, the reason is obvious as we use the first 200 training shards and 100 validation shards as training datasets which leads to possible overfitting. However, we consider that the LSTM and biLSTM's performance of around 0.75 to 0.76 for the whole datasets as satisfactory, since we only use an

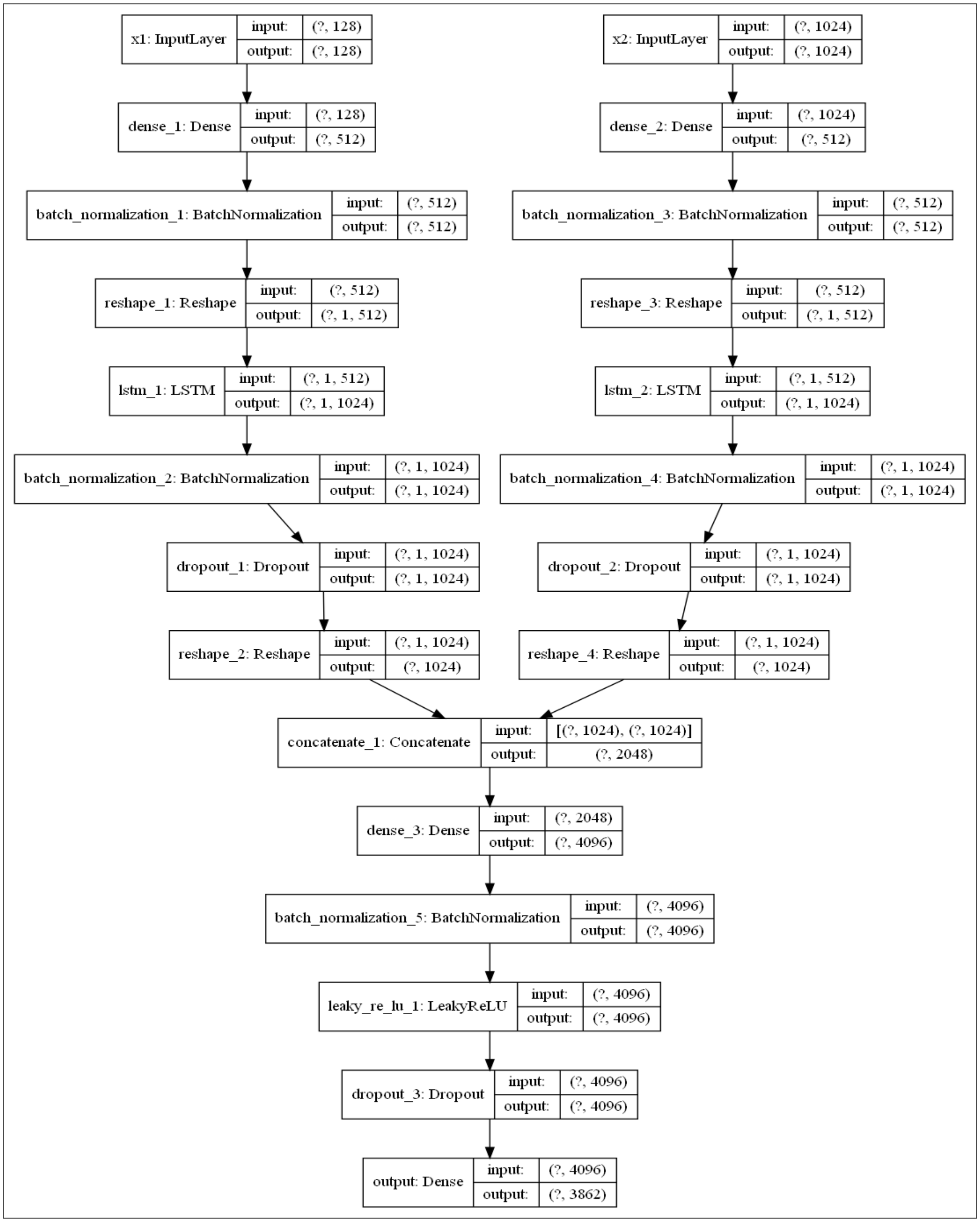


Figure 4: Structure of LSTM

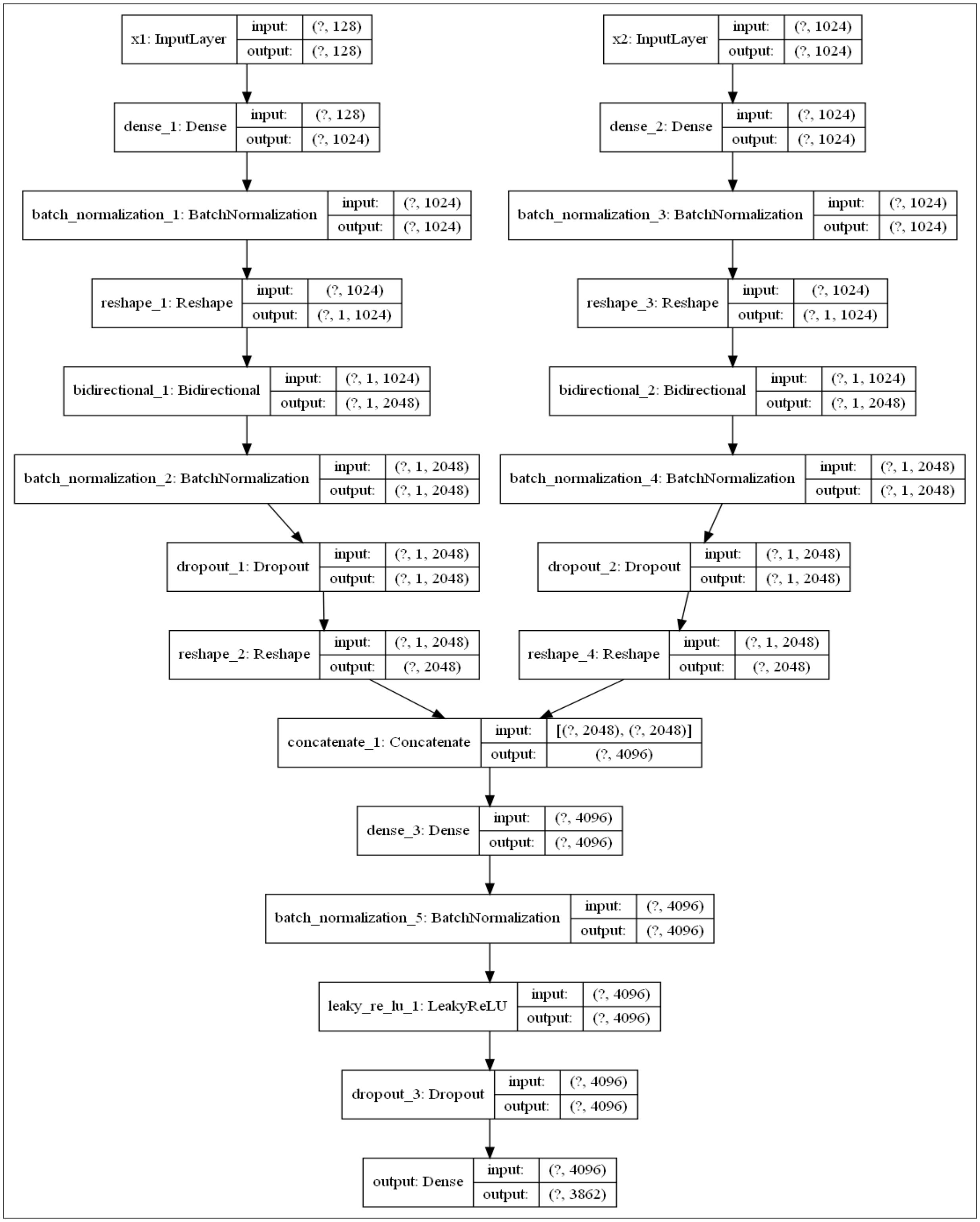


Figure 5: Structure of biLSTM

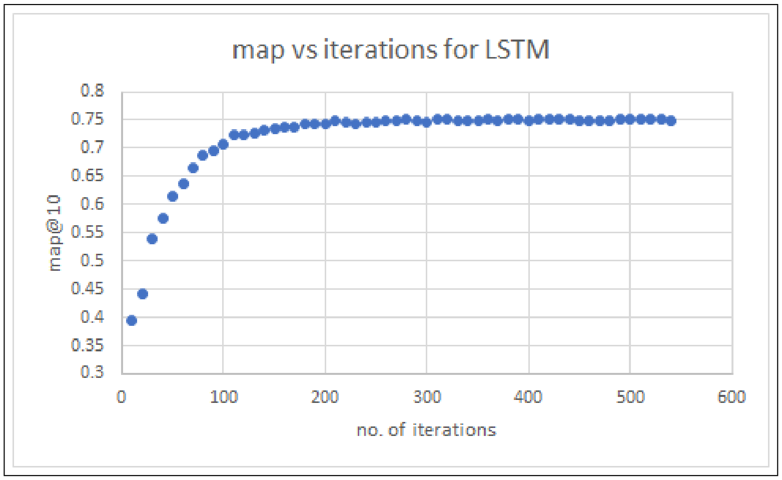


Figure 6: LSTM training graph

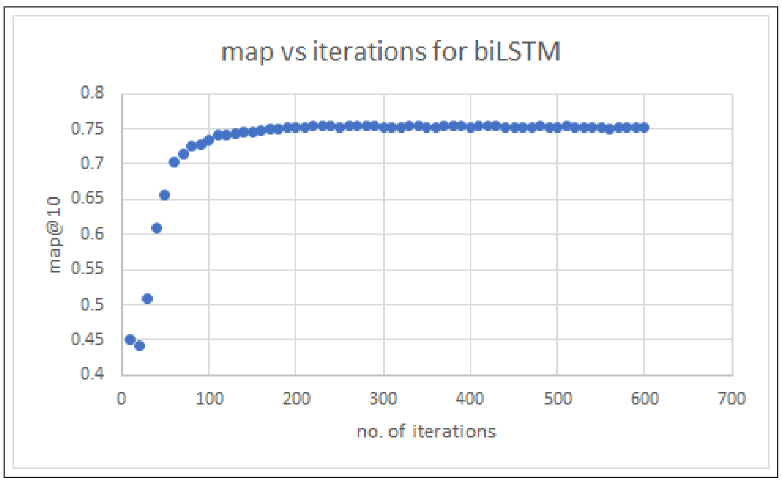


Figure 7: biLSTM training graph

extremely small subset to train them. We hypothesize that if we use a larger datasets for training, the model would lead to more satisfactory results, too bad that we don't have the computing power to prove this hypothesis.

## 4 Conclusion

Based on the experiments, we conclude that the xxx model has a superior performance over the other models we used.

## 5 Contributions

- KAR CHUN TEONG:
  - Production management(organization and distribution of workload, arrangement of meeting schedule, etc.)
  - Analysis and implementation of DMRNets model
  - Design and compilation of final report and PPT
- MATSUNAGA TAKEHIRO
  - Analysis and implementation of LSTM model

    - Training and testing of LSTM model
- EDUARDO WANG ZHENG
    - Implementation of DMRNets model
    - Training and testing of DMRNets model

## References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. **Note that the Reference section does not count towards the eight pages of content that are allowed.**

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition.

[2] Liming Zhao, Mingjie Li, Depu Meng, Xi Li, Zhaoxiang Zhang. Deep Convolutional Neural Networks with Merge-and-Run Mappings.

[3] Sepp Hochreiter. Long Short-Term Memory.

[4] Alex Graves, Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures.