Hyper-parallel Machine Learning and Big Data Mining Project

2019.5

Hongyi Guo

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Contents

Contents

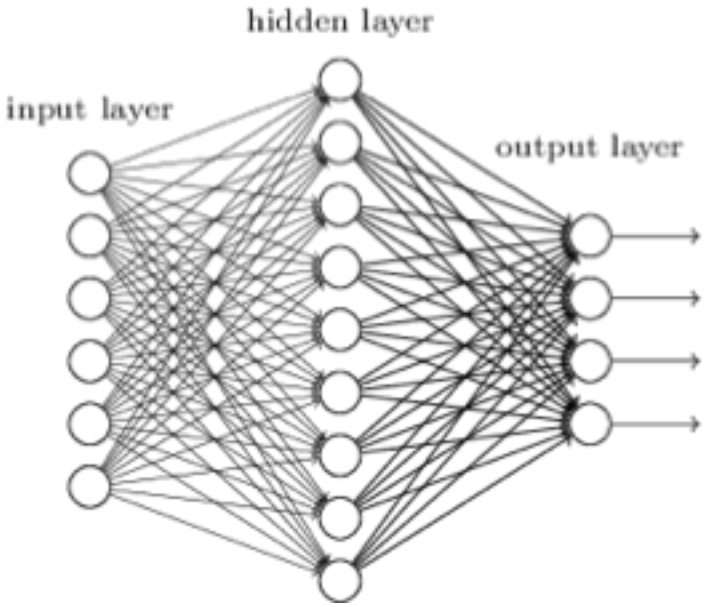
Introduction

- This project aims to solve a 4-class classification problem. To be more specific, we tried to use EEG data to predict which emotion the subject is having.
- The dataset is a subset of SEED-4 dataset, where the feature is differential entropy with 62 leads, and the feature dimension is 62 x 5 = 310. Labels are 0, 1, 2, 3, corresponding to four kinds of emotions: neutral, sad, fear and happy.
- In the following parts, first, I am going to introduce some required baseline models and their experiments, including FC network, Min-Max Modular Network. Then I will introduce the Inductive Bias which is very important for my later model selection. Finally, I will show my CNN and RNN implementation and experiments.

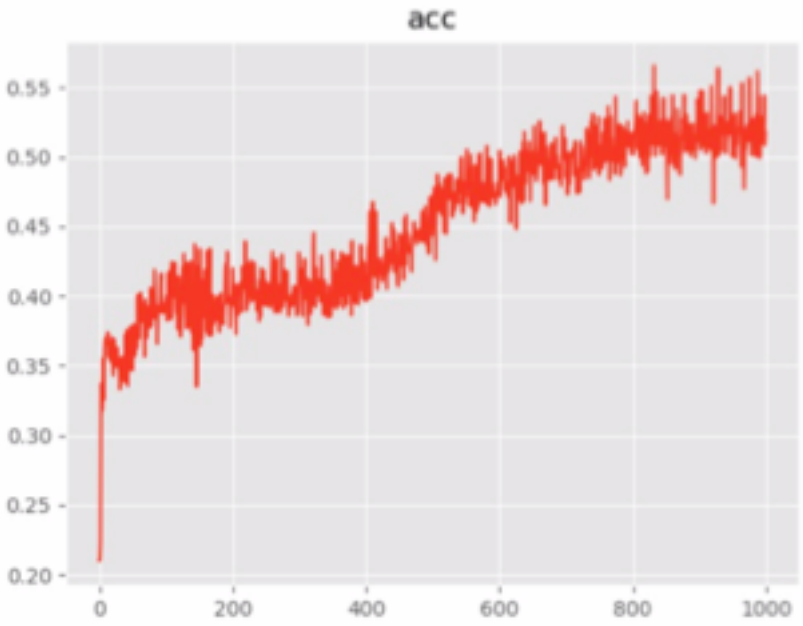
Contents

Fully-connected Network

- In a fully connected layer each neuron is connected to every neuron in the previous layer, and each connection has it's own weight.
- In my implementation, *hidden layer* has 128 units, with *relu* activation function
- *Output layer* has 4 units, with *sigmoid* activation function, representing the probability of being classified into corresponding 4 classes.



Fully-connected Network



(a) Accuracy Curve of Fully-Connected Network

	neutral	sad	fear	happy
neutral	563	108	99	153
sad	107	721	70	350
fear	265	200	162	166
happy	72	113	24	597

Table 1: Confusion Matrix of Fully-Connected Network

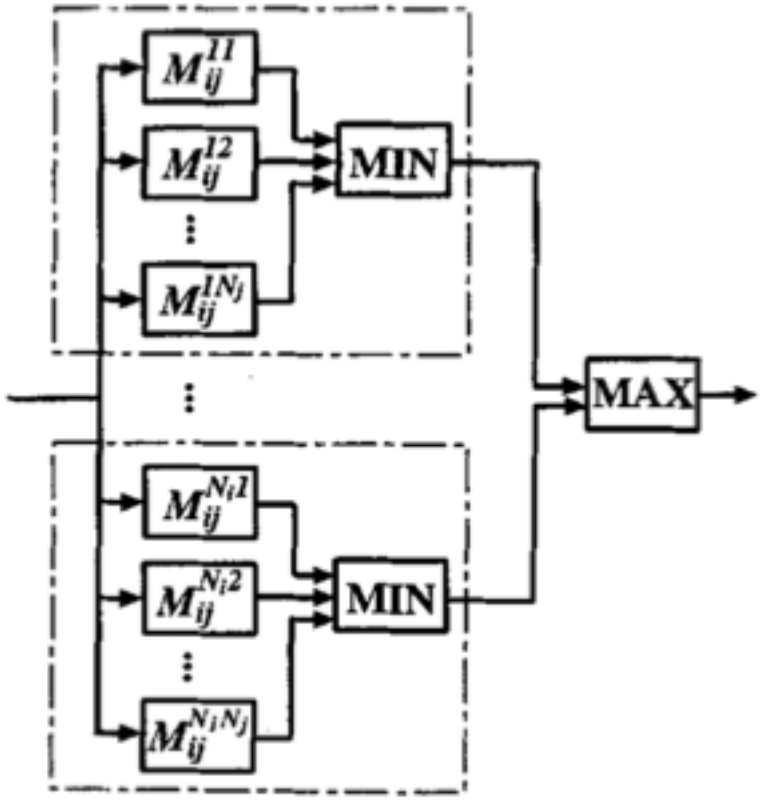
	neutral	sad	fear	happy
F1	0.58	0.60	0.28	0.58

Table 2: F1 Score of Fully-Connected Network

Contents

Min-Max Modular Network

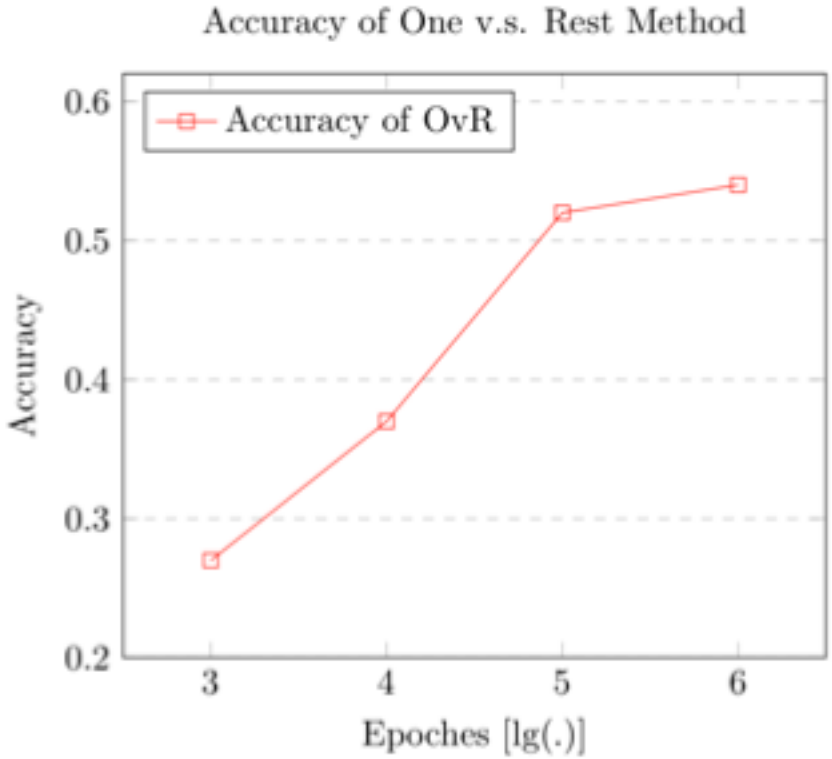
- For a 2-class classification problem
- Divide data from these two classes (i,j) into N parts.
- Then we train models $\mathrm{M}_{\mathrm{ij}}^{mn}$, $m,n \in 1..N, m \leq n$ (The positive samples are from class i`s $m$-th part and negative samples are from class $j$`s n-th part) and take the min values as the probability of being classified into class i.
- Finally, we take max of these **MIN** values. If the **MAX** value is larger than 50%, we predict class i, otherwise, class j.



Min-Max Modular Network (1)

- Multi-class classification random decomposition
  - Assuming there are n classes. For each class, we treat training data from other classes as negative samples. So we transform original problem into n 2-class classification sub-problems. Note that this is similar to the **One-v.s.-Rest** decomposition.
  - For the n outputs of these sub-problems, we let them vote on the final prediction. That is, we predict test data is from class i if the $\mathbf{MAX}$ value of the i-th model is the biggest.
  - So, we believe in the model that is the most self-confident.

Min-Max Modular Network (1)



	neutral	sad	fear	happy
neutral	678	60	28	157
sad	481	673	21	73
fear	278	139	91	285
happy	240	13	0	553

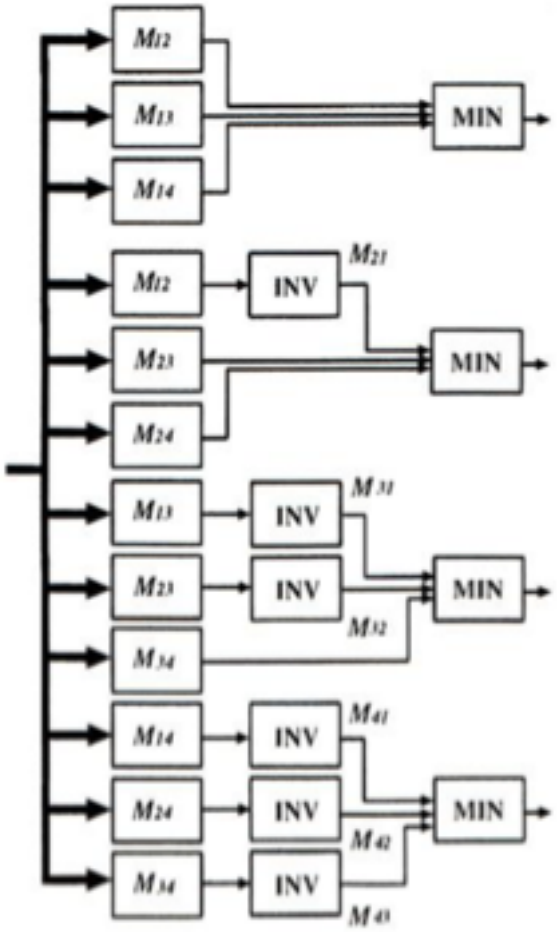
Table 3: Confusion Matrix of Min-Max OvR Method

	neutral	sad	fear	happy
F1	0.52	0.63	0.20	0.59

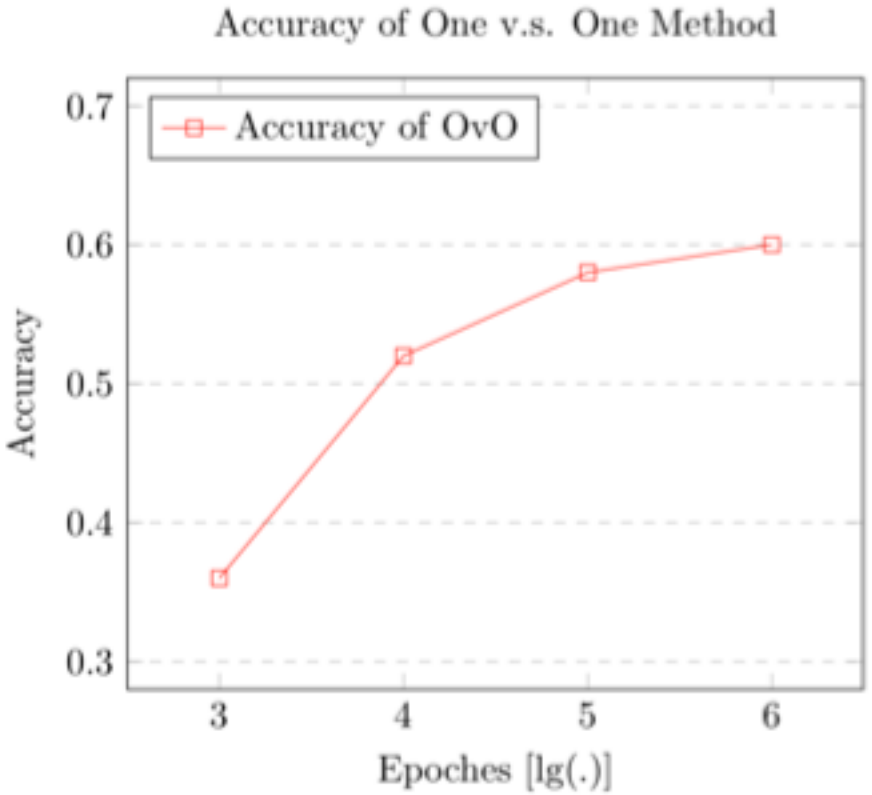
Table 4: F1 Score of Min-Max OvR Method

Min-Max Modular Network (2)

- Multi-class classification prior-based decomposition
  - Assuming there are n classes. We train a classifier for each pair of different classes i, j with Min-Max Modular Network. So we get $\frac{n*(n-1)}{2}$ models in total. And then we vote on the final prediction following the diagram shown on the right.
  - Note that I changed the final MIN operation to AVERAGE, which gives better accuracy in practice.



Min-Max Modular Network (2)



	neutral	sad	fear	happy
neutral	370	105	71	377
sad	133	931	86	98
fear	105	157	180	351
happy	11	62	34	699

Table 5: Confusion Matrix of Min-Max OvO Method

	neutral	sad	fear	happy
F1	0.48	0.74	0.31	0.60

Table 6: F1 Score of Min-Max OvO Method

Contents

Inductive Bias

- Here I want to discuss the difference between nowadays deep neural networks and traditional shadow models from the perspective of the inductive bias.
- The inductive bias (also known as learning bias) of a learning algorithm is **the set of assumptions** that the learner uses to predict outputs given inputs that it has not encountered.
- The kind of necessary assumptions about **the nature of the target function** are subsumed in the phrase inductive bias.

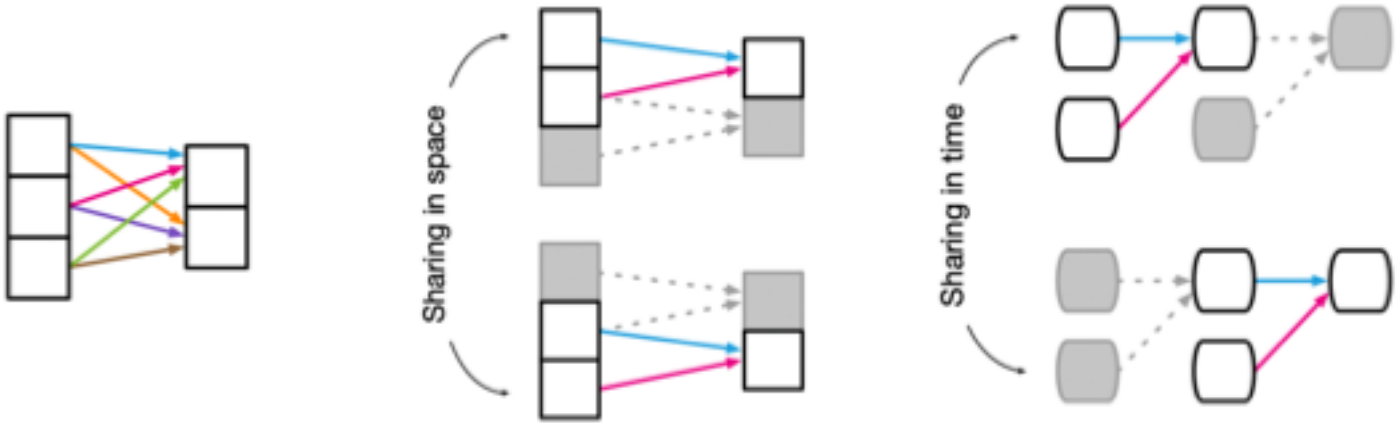
Inductive Bias

- *Maximum margin:*
  - When drawing a boundary between two classes, attempt to **maximize the width of the boundary**. This is the bias used in **support vector machines**. The assumption is that distinct classes tend to be separated by wide boundaries.
- *Nearest neighbors:*
  - Assume that most of the cases in a small neighborhood in feature space belong to the same class. This is the bias used in the **k-nearest neighbors algorithm**. The assumption is that cases that are near each other tend to belong to the same class.
- *All to all:*
  - For fully connected layers, the inductive bias assumes the relations are all-to-all (all units in layer i can affect all units in layer j). The implicit relational inductive bias in a fully connected layer is thus very weak: all input units can interact to determine any output units value, independently across outputs.

Inductive Bias

- *Convolutional layers in CNN*
  - For convolutional layers in CNN, the inductive bias is **locality** and **translation invariance**.
  - Locality reflects that the arguments to the relational rule are those entities in close proximity with one another in the input signals coordinate space, isolated from distal entities. Translation invariance reflects reuse of the same rule across localities in the input.
- *Recurrent layers in RNN*
  - We can view the inputs and hidden states at each processing step as the entities, and the **Markov dependence of one steps hidden state on the previous hidden state and the current input, as the relations**. The rule for combining the entities takes a steps inputs and hidden state as arguments to update the hidden state. The rule is reused over each step, which reflects the relational inductive bias of **temporal invariance** (similar to a CNNs translational invariance in space).
  - RNNs also carry a bias for locality in the sequence via their Markovian structure.

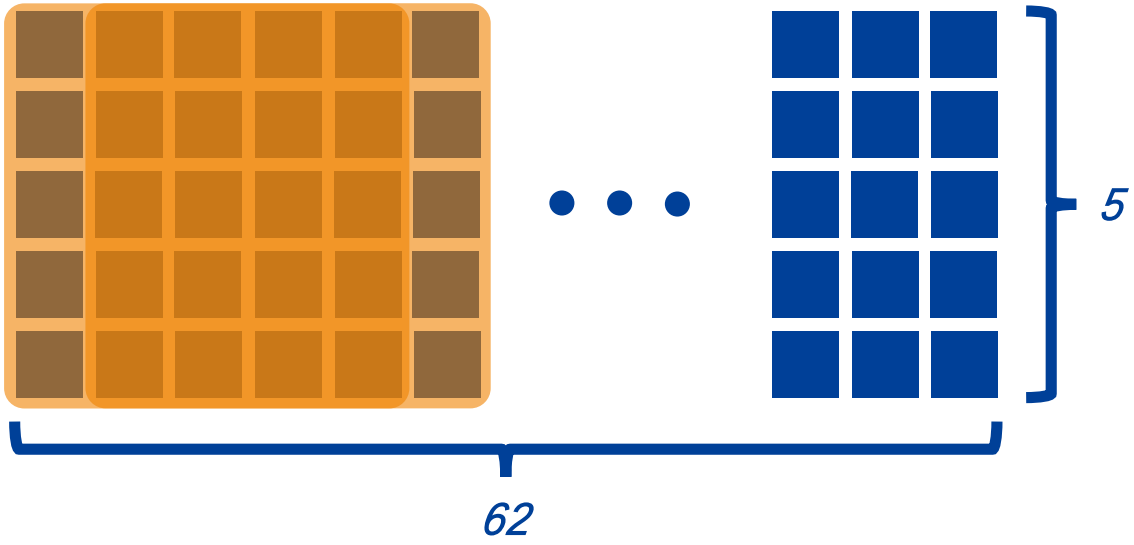
Inductive Bias



Compare the Inductive bias of FC, CNN, RNN

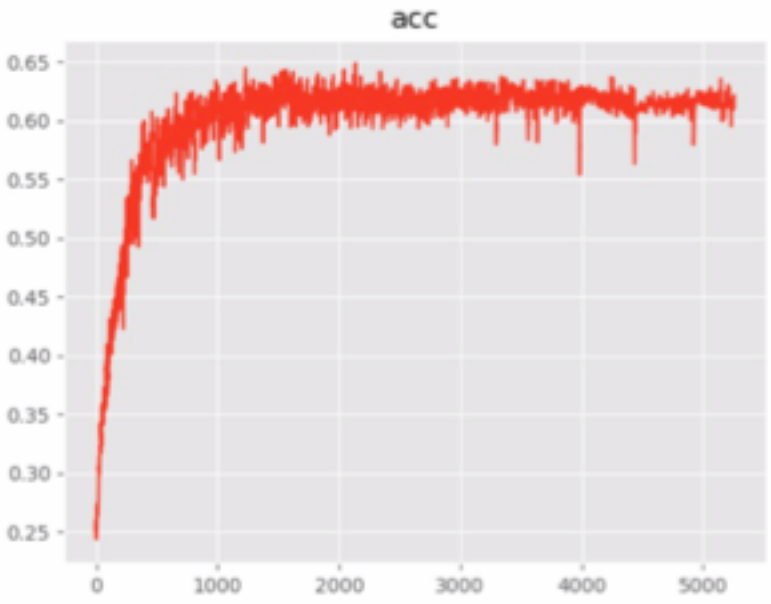
Contents

Apply CNN

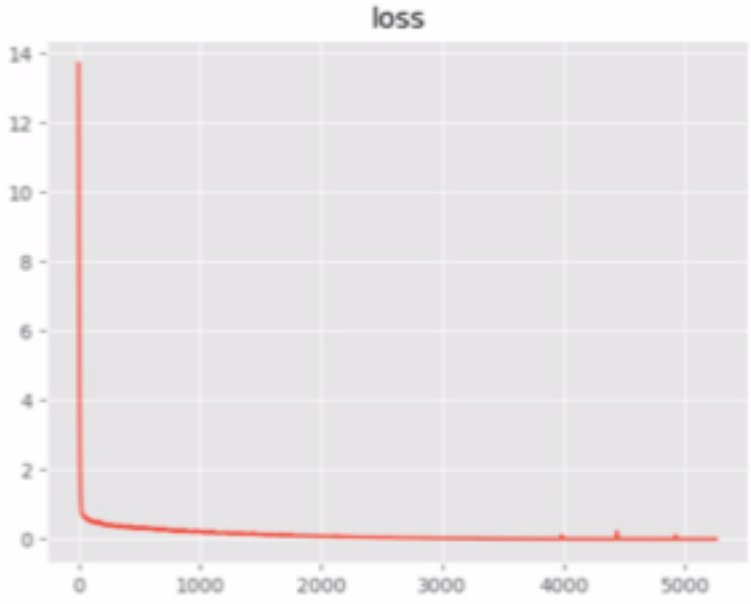


- *Implementation:*
- Conv1: kernel_size=(5,5), filters=16, stride=1, ac=relu
- Conv2: kernel_size=(3,3), filters=32, stride=1, ac=relu
- Fully-connected Layer: units=4, ac=sigmoid
- Pool1: pool_size=(2,2), stride=2
- Pool2: pool_size=(3,3), stride=2

CNN experiments

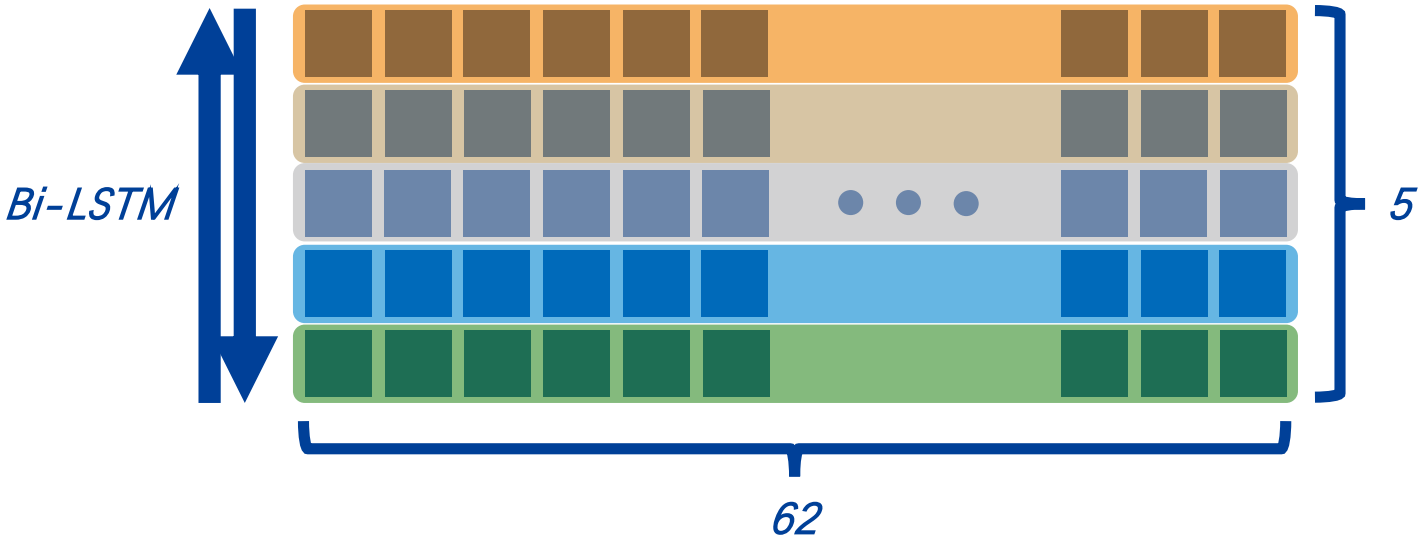


(a) Accuracy Curve of CNN



(b) Loss Curve of CNN

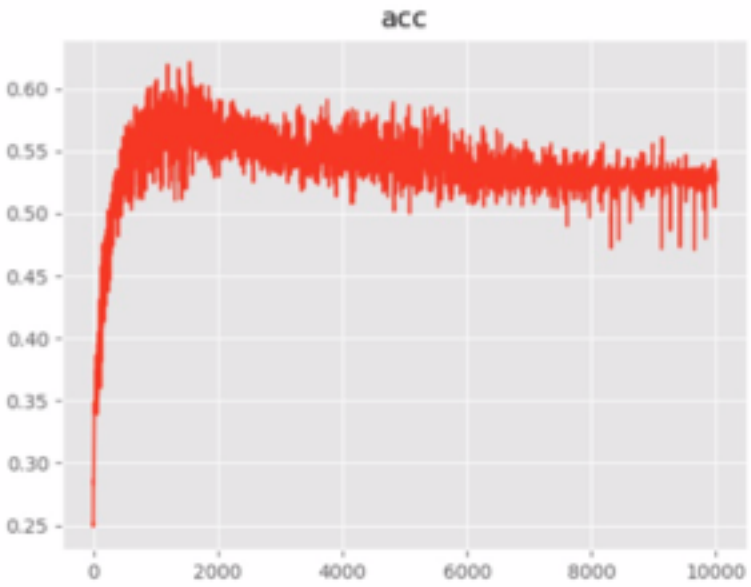
Apply RNN



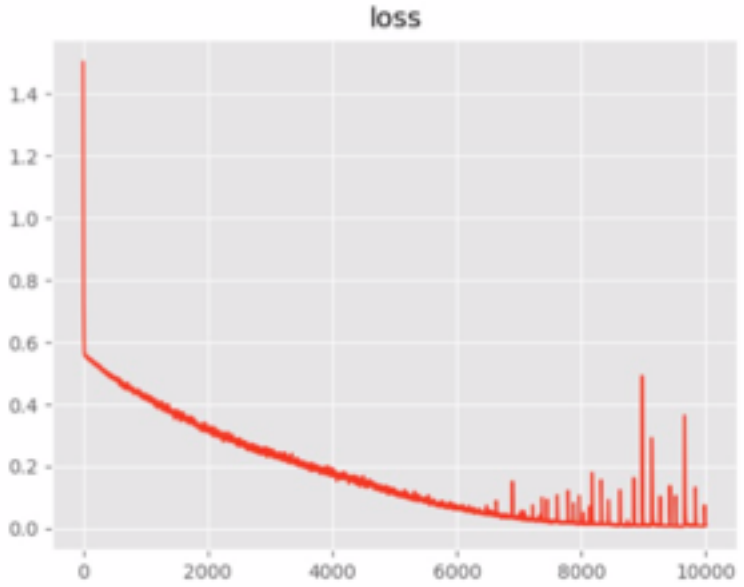
- *Implementation:*

- The latent size of the bidirectional LSTM cell is 32

RNN experiments



(a) Accuracy Curve of Bi-LSTM



(b) Loss Curve of Bi-LSTM

Figure 5: Training Curves of Bi-LSTM

Thank you!