# Machine Learning - deep and structured Final Project

B00902006 何主恩
B00902053 馮　硯
B00902064 宋昊恩
R03922163 姜佳昀

July 2, 2015

## 1 Introduction

The main target of this final project is to pipeline all of our previous works (from Hw1 to Hw3). Though it seems quite easy at the first sight, we need to take care of the problem of error propogation. In our following works, we decide to rebuild and improve all of our works in the past in order to have a better final result.
The following sections include *Improvement from Hw1*, *Improvement from Hw2* and *Improvement from Hw3*. We will show the details of new algorithms and their experiment results.

## 2 Improvement from Hw1

We try three different methods here. They are: *Improved Self-implemented DNN*, *DNN with Caffe Package* and *CNN with Caffe Package*.

### 2.1 Improved Self-implemented DNN

#### 2.1.1 Implementation

We improve our previous DNN work in Hw1 as followings.

- Replace *Sigmoid* unit with *ReLU* to shorten our training time significantly
- Implement *dropout* regularization with rate 0.5 to avoid overfitting
- Implement *l2-normalization* to avoid overfitting
- Reduce *learnining rate* from 0.01 to 0.001
- Reduce *batch size* from 1024 to 128 to enhance accuracy

#### 2.1.2 Experiment Result

Enhance on-board performance from 0.7 to 0.733.

### 2.2 Caffe Package

#### 2.2.1 Brief description

To enhance the performance of Hw1, we use *Caffe* package with various kinds of layers to implement different model design. We run our experiments on *Tesla K40* to accelerate.

#### 2.2.2 DNN - Experiment Result

- Gerneral Setting:
  - number of features: $69 \times 9$
  - learning rate: 0.001
  - momentum: 0.95
  - dropout rate: 0.2
  - number of epoch: 35
  - batch size: 128
- Table Result (Accuracy):

| layers | off-board (valid) | on-board (public) | on-board (private) |
|---|---|---|---|
| 1024-512-256-48 | 0.658 | 0.70003 | 0.69956 |

#### 2.2.3 CNN - Experiment Result

- Gerneral Setting:
  - number of features: $69 \times 9$
  - learning rate: 0.0001
  - momentum: 0.95
  - dropout rate: 0.2
  - number of epoch: 22
  - batch size: 128
- Table Result (Accuracy):

| layers | off-board (valid) | on-board (public) | on-board (private) |
|---|---|---|---|
| 30, 3x3-30, 3x3-30, 3x3-2048-2048-48 | 0.707 | 0.73318 | 0.73383 |
| 30, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 0.707 | 0.73859 | 0.73828 |
| 64, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 0.707 | 0.73845 | 0.73815 |

## 3 Improvement from Hw2

There are three possible ways to fulfill this target. They are: *naive stitching*, *hidden markov model*, and *nbest-rnnlm*.

### 3.1 Naive stitching

#### 3.1.1 Brief description

It is an esay but also well-performed baseline algorithm. We stitch the transformed phone sequence and then eliminate the continued replicate phone characters. Besides that we remove the phone characters with too few appearances. Our experiments are as follows.

#### 3.1.2 DNN - Experiment Result

- Gerneral Setting:
  - number of features: $69 \times 9$
  - learning rate: 0.001
  - momentum: 0.95
  - dropout rate: 0.2
  - number of epoch: 35
  - batch size: 128
- Table Result (edit distance):

| layers | appearance (>) | on-board (public) | on-board (private) |
|---|---|---|---|
| 1024-512-256-48 | 0 | 29.40541 | 27.14189 |
| 1024-512-256-48 | 1 | 17.24324 | 16.06419 |
| 1024-512-256-48 | 2 | 12.79392 | 11.85135 |
| 1024-512-256-48 | 3 | 13.18243 | 12.51689 |

#### 3.1.3 CNN - Experiment Result

- Gerneral Setting:
  - number of features: $69 \times 9$
  - learning rate: 0.0001
  - momentum: 0.95
  - dropout rate: 0.2
  - number of epoch: 22
  - batch size: 128
- Table Result (edit distance):

| layers | appearance (>) | on-board (public) | on-board (private) |
|---|---|---|---|
| 30, 3x3-30, 3x3-30, 3x3-2048-2048-48 | 0 | 30.40203 | 28.94257 |
| 30, 3x3-30, 3x3-30, 3x3-2048-2048-48 | 1 | 18.75676 | 17.82432 |
| 30, 3x3-30, 3x3-30, 3x3-2048-2048-48 | 2 | 14.20270 | 13.53041 |
| 30, 3x3-30, 3x3-30, 3x3-2048-2048-48 | 3 | 14.38851 | 13.52027 |
| 30, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 0 | 28.79730 | 26.84797 |
| 30, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 1 | 16.72973 | 15.67230 |
| 30, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 2 | 12.46959 | 11.52703 |
| 30, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 3 | 13.02365 | 12.32770 |
| 64, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 0 | 28.33108 | 26.76689 |
| 64, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 1 | 16.61824 | 15.56419 |
| 64, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 2 | 12.42568 | 11.40541 |
| 64, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 3 | 12.67568 | 12.13514 |

### 3.2 Hidden markov model

#### 3.2.1 Brief description

*Hidden markov model* is the most common solution for this problem. Initially, we try to use the package HMMLib in *C++* language, however, we finally abandon this method. The main reason is that it is hard to decide the number of hidden states, $k$, and the varied length of observed sequence (phone sequemces from Hw1) will be largely affected by this parameter.
Instead, we decide to implement basic *Viterbi Algorithm* by ourselves. In this way, we can avoid to decide the number of states, but also increase our on-board score.

#### 3.2.2 Experiment Result

Improve our origin best score 11.92 to 10.61.

### 3.3 Nbest-rnnlm

#### 3.3.1 Brief description

For a sequence s, define

$$P(s) = \prod_i p(\text{i-th phone}|\text{i-th position})$$

where $p$(i-th phone|i-th position) is the same as *DNN soft-max* output.

Use Viterbi to find best $n$ sequences in terms of their $P$. Then rescore by *RNNLM* which trained on training data' s per frame phone sequences.

#### 3.3.2 Experiment Result

- Table Result (edit distance):

| Nbest | on-board (public) |
|---|---|
| 5 | 41.54392 |
| 100 | 38.68919 |
| 500 | 35.375 |

## 4 Improvement from Hw3

### 4.1 RNNLM

#### 4.1.1 Brief description

After using *Lexicon WFST* tool, we can generate n-best sequence for each sentence. Then, we directly use *RNNLM* package to find out the most likely sequence with maximum probability. Finally, we transfer the most likely sequence into required output with script privided by TA.
There are two available corpus that we can make use of. One is the previous dataset for Hw3, and another one is the given training set for final project.

#### 4.1.2 DNN - Experiment Result

- Gerneral Setting:
  - number of features: $69 \times 9$
  - learning rate: 0.001
  - momentum: 0.95
  - dropout rate: 0.2
  - number of epoch: 35
  - batch size: 128
- Table Result (edit distance):

| layers | NBest | on-board (public) |
|---|---|---|
| 1024-512-256-48 | 1 | 29.40541 |
| 1024-512-256-48 | 100 | 8.70037 |

#### 4.1.3 CNN - Experiment Result

- Gerneral Setting:
  - number of features: $69 \times 9$
  - learning rate: 0.0001
  - momentum: 0.95
  - dropout rate: 0.2
  - number of epoch: 22
  - batch size: 128
- Table Result (edit distance):

| layers | NBest | on-board (public) |
|---|---|---|
| 30, 3x3-30, 3x3-30, 3x3-2048-2048-48 | 1 | 9.29588 |
| 30, 3x3-30, 3x3-30, 3x3-2048-2048-48 | 100 | 8.89513 |
| 30, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 1 | 9.07116 |
| 30, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 100 | 8.77903 |
| 64, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 1 | 9.07116 |
| 64, 3x3-30, 3x3-30, 3x3-1024-1024-48 | 100 | 8.86517 |

#### 4.1.4 Viterbi - Experiment

- Without *RNNLM*: 7.34082
- With NBest *RNNLM*: 7.14981

## 5 Division of Teamwork

- 何主恩：
  - *Improved Self-implemented DNN* [Hw1]
  - *RNNLM* [Hw2]
- 宋昊恩：
  - *Hidden Markov Model, Viterbi Algorithm Implementation* [Hw2]
  - *RNNLM* [Hw3]
  - Report Edition
- 姜佳昀：
  - *DNN with Caffe Package* [Hw1]
  - *CNN with Caffe Package* [Hw1]