DS-GA-1011 Assignment 2

Ruofan Wang

1 Pipeline and Code

In this assignment, we aim at solving the SNLI challenge using CNN and RNN. The whole process includes the following parts:

- Data loading and pre-processing
- Tokenize and load pre-trained embedding
- Train Model: Hyper parameter searching using Validation set
- Evaluate on the MNLI dataset and fine tuning

Here is the link to the github repo: https://github.com/RuofanW/NLPclass

2 Preprocessing and Load Embedding

We use NLTK tokenizer to tokenize the given sentences. After tokenizing, we load the pre-trained fast-text work 300-d embedding. We also use token '<pad>' and '<unk>' for padding and unknown words.

Then we build the dataloader for training and validation set. In the dataloader, every sample contains sentence 1, sentence 2 and label. There are 3 classes in total.

3 Modelling and Parameter Tuning

3.1 RNN Models and Hyperparameter Search

3.1.1 Baseline RNN Architecture

We use an RNN encoder to map each string of text (hypothesis and premise) to a fixed-dimension vector representation. The RNN here is a bi-directional GRU with default dropout rate 0.2.

Then we interact the two hidden representations and output a 3-class softmax. We simply concatenate the two representations, and feed them through a network of 2 fully-connected layers.

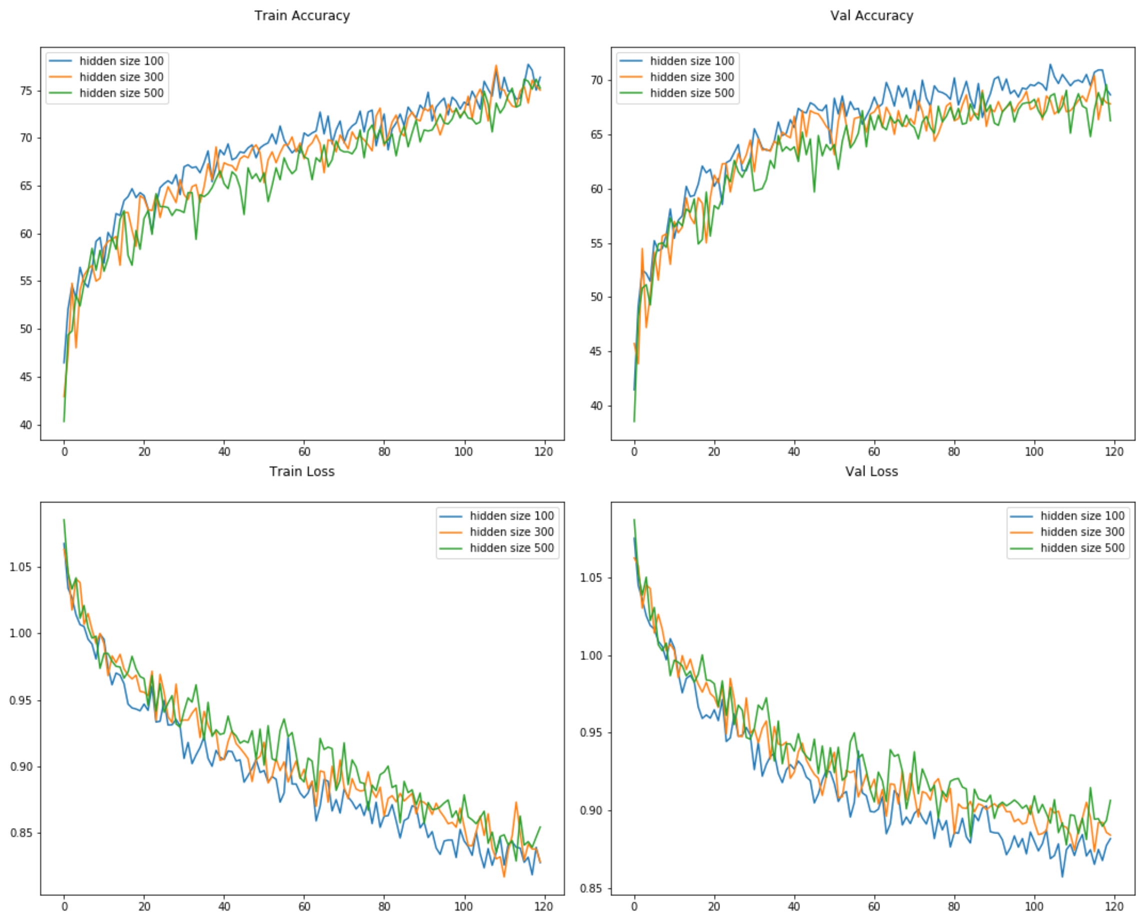
After building the baseline architecture, we conduct hyperparameter search using training and validation set.

3.1.2 Hidden Size

For hidden size of RNN, we try three different values and the results are plotted as follows. As we can see, the best hidden size is 100. So we choose 100 as hidden size in our furture work.

The number of trained parameters:

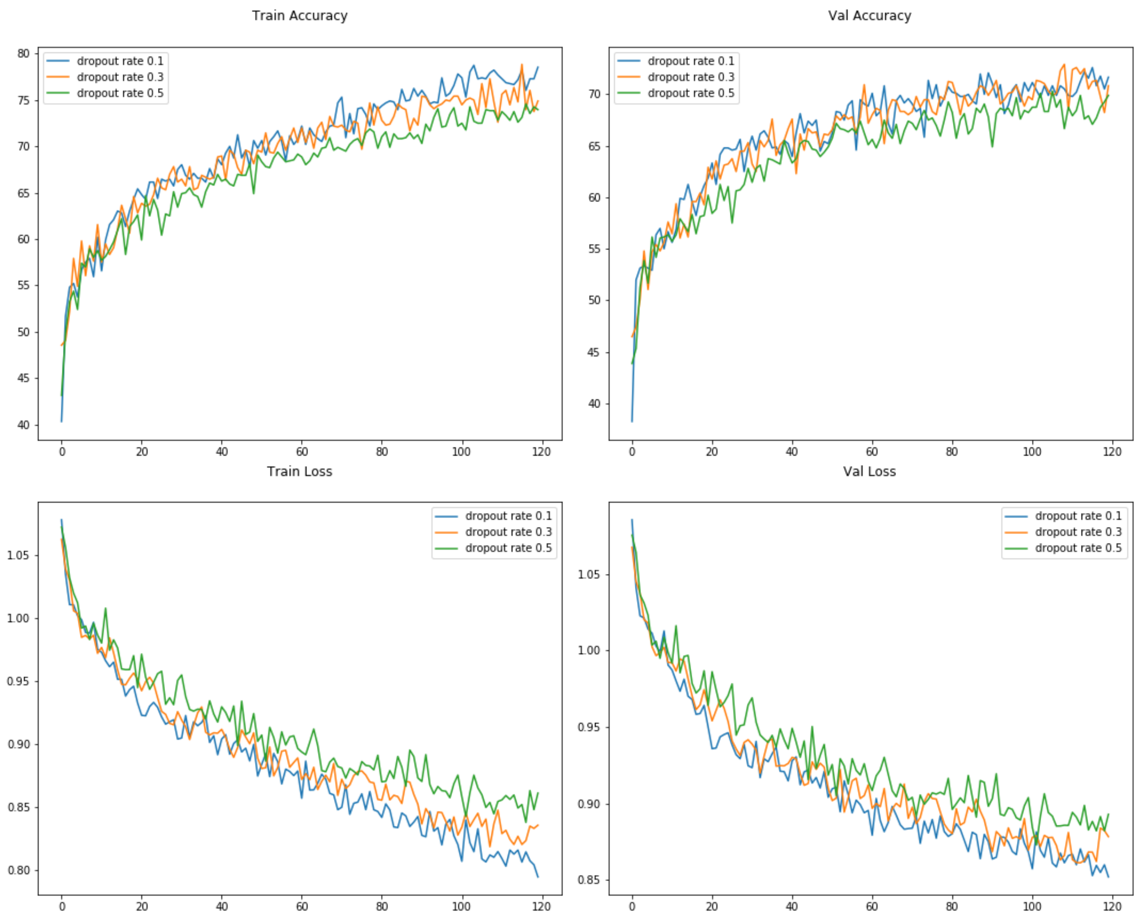
hidden size	# params
100	644003
300	2648803
500	5613603



3.1.3 Dropout

We can also add dropout to do regularization, which is an effective method to avoid overfitting. Dropout (definition from wiki) is a regularization technique for reducing overfitting by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. For dropout rates, we try 0.1, 0.3 and 0.5. And the results are also plotted below. From this figure, we can see the optimal dropout rate should be 0.1.

Changing dropout rate does not affect the number of trained parameters, so here in this case, all the models have 644003 trained parameters.



3.2 CNN Models and Hyperparameter Search

3.2.1 Baseline CNN Architecture

This time we use an CNN encoder to map each string of text (hypothesis and premise) to a fixed-dimension vector representation.

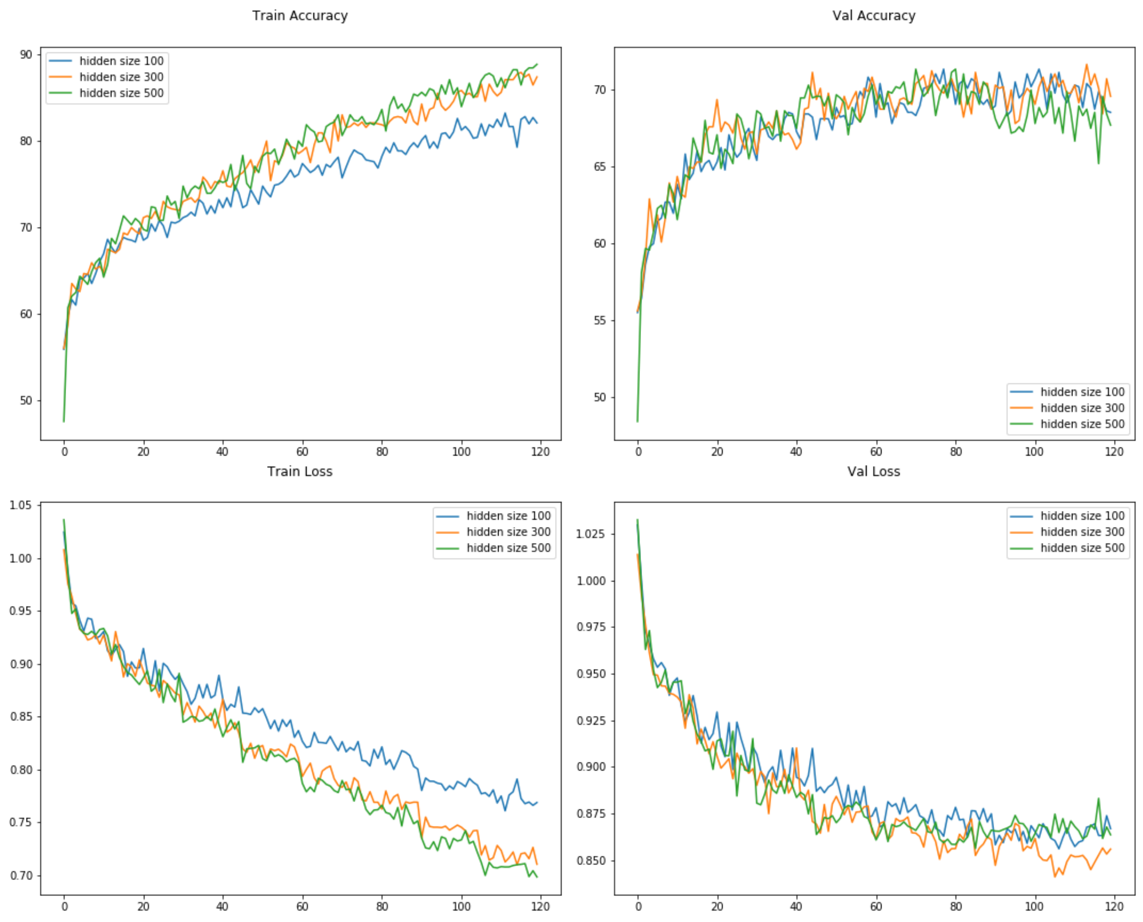
Then we interact the two hidden representations and output a 3-class softmax. We simply concatenate the two representations, and feed them through a network of 2 fully-connected layers.

After building the baseline architecture, we conduct hyperparameter search using training and validation set.

3.2.2 Hidden Size

For hidden size of CNN, we try three different values and the results are plotted as follows. As we can see, the best hidden size is 300, since when hidden size is 300, our model reaches the highest validation accuracy. The number of trained parameters:

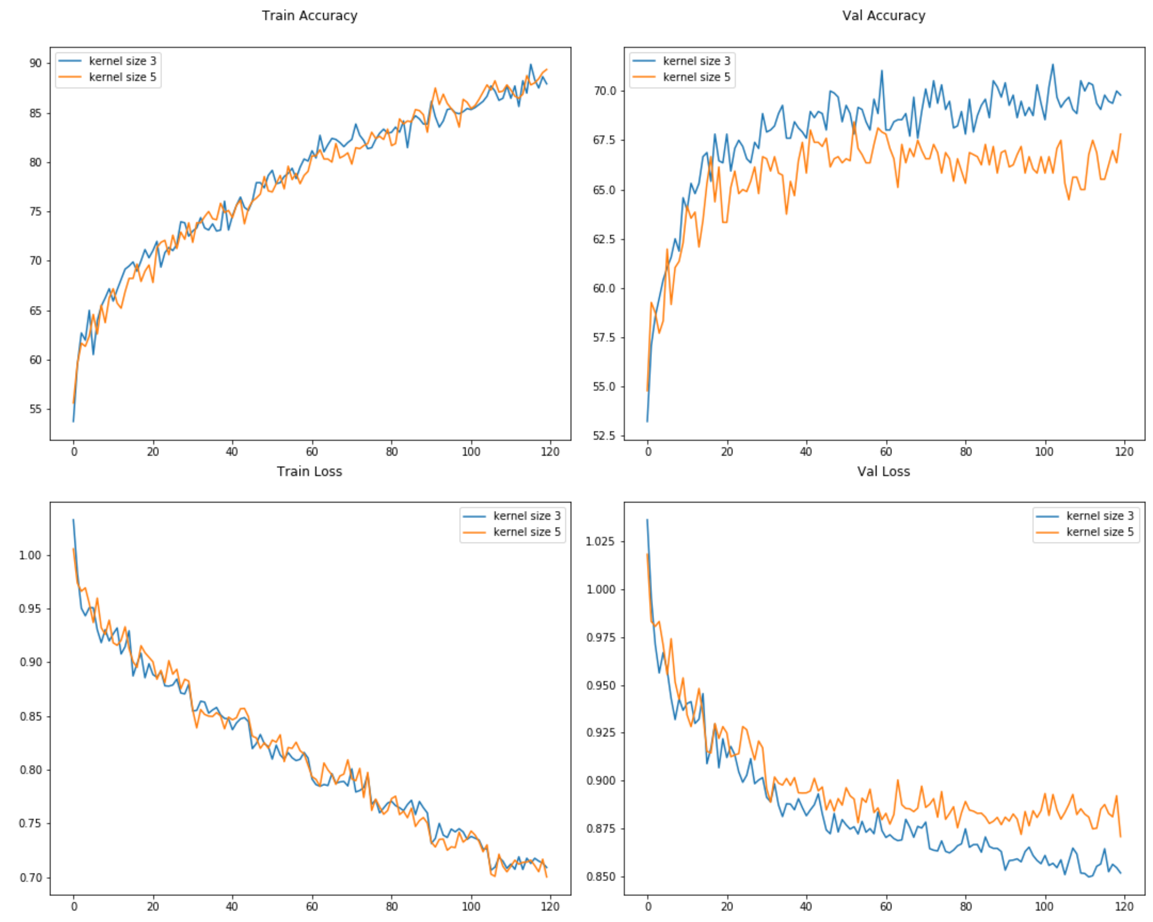
hidden size	# params
100	161003
300	903003
500	2205003



3.2.3 Kernel Size

In the convolutional network, kernel is one of the most important components. Kernel is a small receptive field which can capture local information. Here we try different kernel size, 3, 5. Their accuracies are plot as below. From this figure it is easy to tell that kernel size 3 is the optimal. In this way we can finalize our CNN model.

kernel size	# params
3	903003
5	1263003

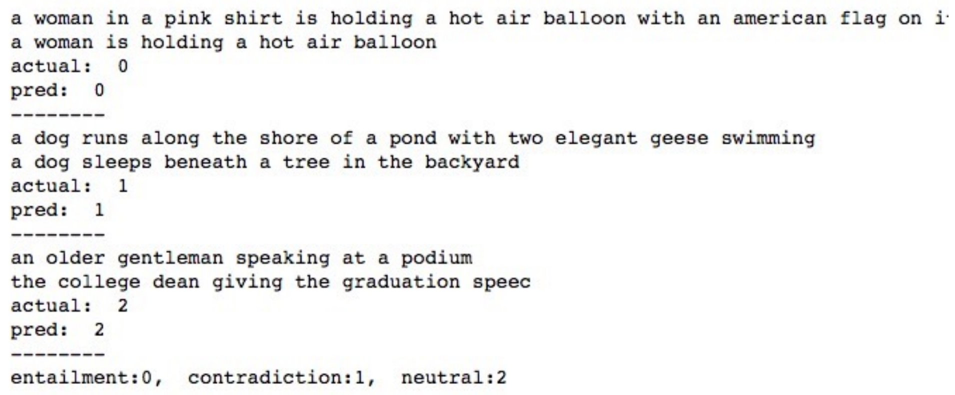


3.3 Final Model

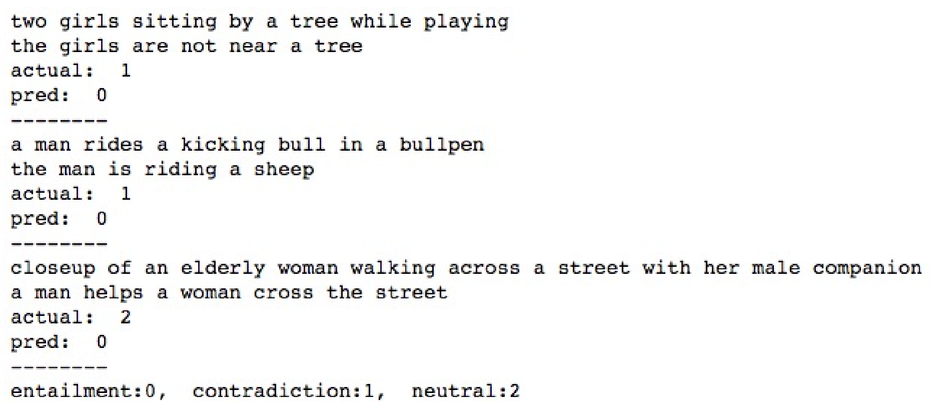
Based on our previous analysis, our final model for RNN contains two bi-directional GRU and two fully-connected layer, with hidden size 100 and dropout rate 0.3. With this model, the accuracy on the validation set can reach 72.9%.

For CNN models, the best configuration is hidden size equals to 300 and kernel size equals to 3. The accuracy on the validation set is 71.6%.

Overall, the best model is the RNN one with the above configuration. Here are 3 correct and incorrect examples:



```
a woman in a pink shirt is holding a hot air balloon with an american flag on i
a woman is holding a hot air balloon
actual:  0
pred:  0
---------
a dog runs along the shore of a pond with two elegant geese swimming
a dog sleeps beneath a tree in the backyard
actual:  1
pred:  1
---------
an older gentleman speaking at a podium
the college dean giving the graduation speec
actual:  2
pred:  2
---------
entailment:0,  contradiction:1,  neutral:2
```



```
two girls sitting by a tree while playing
the girls are not near a tree
actual:  1
pred:  0
--------
a man rides a kicking bull in a bullpen
the man is riding a sheep
actual:  1
pred:  0
--------
closeup of an elderly woman walking across a street with her male companion
a man helps a woman cross the street
actual:  2
pred:  0
--------
entailment:0,  contradiction:1,  neutral:2
```

Analysis: For the incorrect examples, maybe there is some ambiguity within the sentences, which is misleading to our model. The structure of the two sentences are different, which can also mislead the prediction.

4 Evaluating on MultiNLI

After loading the MNLI data, we can feed the validation data into the best CNN and RNN model. We list the accuracy in the table:

genre	model	accuracy
slate	CNN	40.32
telephone	CNN	44.37
government	CNN	42.32
travel	CNN	43.78
fiction	CNN	47.13

genre	model	accuracy
slate	RNN	45.61
telephone	RNN	43.38
government	RNN	46.65
travel	RNN	45.92
fiction	RNN	42.41

Comments: In general, the accuracy of both RNN and CNN decrease dramatically, RNN performs slightly better than CNN in this dataset. From the listed accuracy, RNN is more stable across genres, while CNN models vary greatly.

5 Fine Tuning on MNLI

As stated in the problem, here we take the optimal RNN model to train 3 epoches for each genre using the training samples of this genre. Then we evaluate the validation accuracy. Here are the results before and after fine tuning:

genre	After Fine Tuning	Before Fine Tuning
slate	53.13	45.61
telephone	59.82	43.38
government	60.83	46.65
travel	53.89	45.92
fiction	58.58	42.41