DSGA-1011 Natural Language Processing with Representation Learning Homework 2

Xintian Han (xh1007)

October 30, 2018

1 SNLI Classification

We build CNN and RNN encoders together with a fully-connected network to solve the 3-class classification problem in SNLI. We use an encoder (either a CNN or an RNN) to map each string of text (hypothesis and premise) to a fixed-dimension vector representation. We concatenate or multiply the two representations and feed them through a network of 2 fully-connected layers with a ReLU activation. We use pretrained word embeddings from fast-Text to embed our word tokens. We retrain the embedding of UNK token in our task. For the CNN, we use a 2-layer 1-D convolutional network with ReLU activations. For the RNN, we use a single-layer, bi-directional GRU and take the last hidden state as the encoder output. (We sum up the hidden states from two directions.) The embedding size is 300 when we use pretrained fast-Text word vectors. We use the following set of hyperparameters for RNN and CNN.

- The size of the hidden dimension of the CNN and RNN `hidden_size`: 300, 400, 500.
- The kernel size of the CNN `kernel_size`: 3,5 (For kernel size 3, we use padding = 3 and for kernel size 5, we use padding = 5.)
- Different ways of interacting the two encoded sentences `is_concat`: concatenation and element-wise multiplication.
- Regularization: weight decay `is_wd` and dropout `is_dropout`.

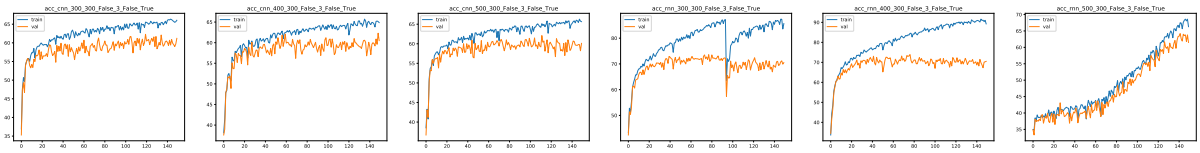
Some fixed hyperparameters are optimizer (ADAM), learning rate (initial learning rate = 0.0001 and the i-th epoch has the learning rate initial learning rate/ i), fixed hidden dimension `hid_dim`= 300 in fully-connected network. We use max sentence length 50 which is greater than 95% quantile of the whole sentence lengths. All networks we trained for 10 epochs. The github link for this homework is https://github.com/XintianHan/nlp_2018

1.1 Vary Hidden Sizes

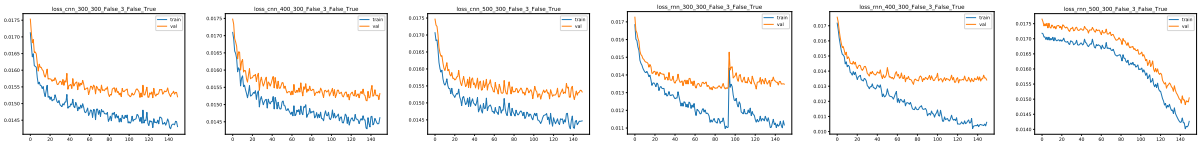
We first use dropout, no weight decay, element-wise multiplication and kernel size 3. We vary hidden size for both RNN and CNN. We also choose the best model that has the best validation accuracies during each epoch of training. The results are in Table 1 and Figure 1. The title of the figure for train and validation loss and accuracy curves are:
'loss_' and 'acc_' plus '`encoder_hidden_size_hid_dim_is_concat_kernel_size_is_wd_is_dropout`'.
We find that hidden size 300 is the best for CNN and 400 for RNN. Actually, hidden size 300 and 400 for CNN have the same validation accuracy, but due to the Occam's razor, we choose the one with lower hidden size. Higher hidden size will have higher train accuracy but may have poorer generalization. Here the lowest hidden size is the best for CNN. For RNN, we notice that when hidden size = 500, the training may not converge. But due to the limit time and resources, we train each model 10 epochs. So our best choice for hidden size of RNN is 400. It is faster than 500 but more accurate than 300 when training 10 epochs.

Network	CNN			RNN		
hidden size	300	400	500	300	400	500
validation accuracy	**62.3**	62.3	62.1	73.6	**73.9**	64.2
number of parameters	632103	962303	1352503	1175103	1806303	2557503

Table 1: Accuracy on Validation Set and Number of Parameters When We Vary Hidden Sizes, and Fix the Others.



(a) 300 CNN (b) 400 CNN (c) 500 CNN (d) 300 RNN (e) 400 RNN (f) 500 RNN



(g) 300 CNN (h) 400 CNN (i) 500 CNN (j) 300 RNN (k) 400 RNN (l) 500 RNN

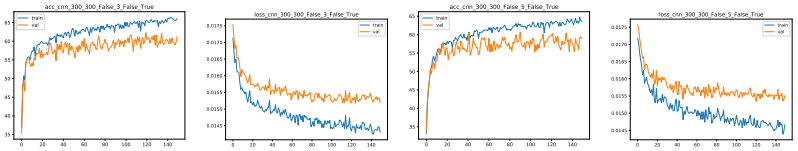
Figure 1: Accuracy and Loss on Validation and Train Set When We Vary Hidden Sizes, and Fix the Others.

1.2 Vary Kernel Size

We use the previously tuned best `hidden_size` = 300 and vary kernel size only for CNN. The results are shown in Table 2 and Figure 2. The best kernel size is 3. Usually, small kernel size catches high frequency features. Here, maybe high frequency features are more important. Another view is that kernel size 3 is corresponding to 3-gram and kernel size 5 is corresponding to 5-gram. 3-gram may play a more important part in SNLI classification. Actually, the accuracy is very close so kernel size does not matter too much.

kernel size	3	5
Validation Accuracy	**62.3**	60.7
number of parameters	632103	992103

Table 2: Accuracy on Validation Set and Number of Parameters When We Vary Kernel Sizes, and Fix the Others.



(a) 3 (b) 3 (c) 5 (d) 5

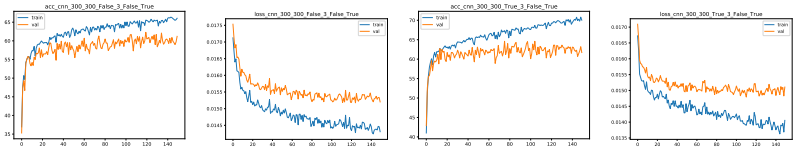
Figure 2: Accuracy and Loss on Validation and Train Set When We Vary Kernel Sizes, and Fix the Others.

1.3 Concatenation or Element-wise Multiplication

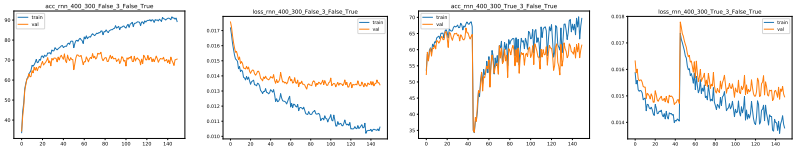
We choose hidden size 300 and kernel size 3 for CNN and hidden size 400 for RNN. We vary the way of combining two representations after encoding. The results are shown in Table 3 and Figure 3. We find that CNN prefers concatenation and RNN prefers element-wise multiplication. In GRU, we some sense of multiplication between features through gates. Maybe this makes RNN prefers multiplication. It is hard to say why one way of combining representations is better than the other.

Network	CNN		RNN	
Combining Method	Concatenation	Element-wise Multiplication	Concatenation	Element-wise Multiplication
Validation Accuracy	**64.6**	62.3	66.7	**73.9**
number of parameters	722103	632103	1926303	1806303

Table 3: Accuracy on Validation Set and Number of Parameters When We Vary Combining Methods, and Fix the Others.



(a) False, CNN (b) False, CNN (c) True, CNN (d) True, CNN



(e) False, RNN (f) False, RNN (g) True, RNN (h) True, RNN

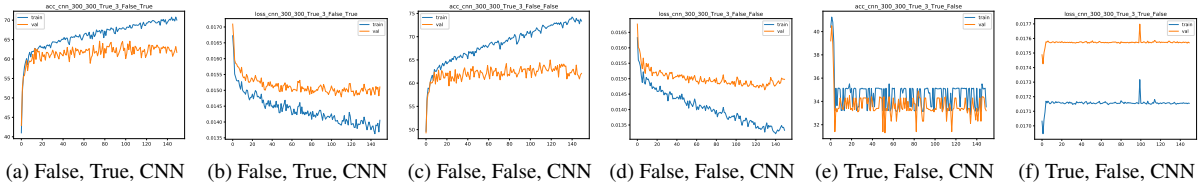
Figure 3: Accuracy and Loss on Validation and Train Set When We Vary Combining Methods, and Fix the Others.

1.4 Vary Regularizations

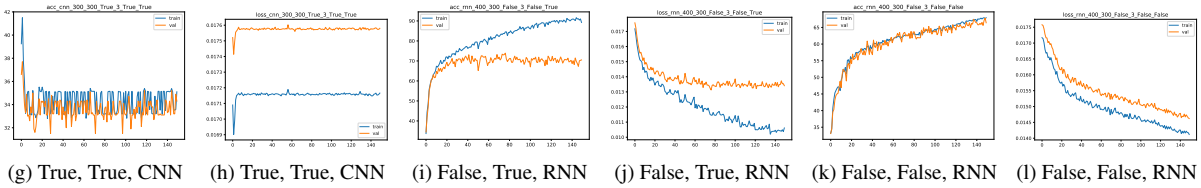
We use weight decay = 5e-4 in Adam and a dropout layer between two fully connected layers. The results are in Table 4 and Figure 4. Weight decay does not work for this task. The reason could be the penalty level is too high or the same learning rate does not work for weight decay. To be honest, I do not have time to explore learning rate and weight decay. Dropout is better for RNN but not for CNN. For CNN, the difference of validation accuracy for with dropout and without dropout is very small, like 0.4. So we could not make a strong conclusion that dropout does not work here.

Network	CNN				RNN			
`is_dropout`	True		False		True		False	
`is_wd`	True	False	True	False	True	False	True	False
validation accuracy	37.7	64.6	40.5	**65.0**	36.0	**73.9**	35.3	67.8
number of parameters	722103	722103	722103	722103	1806303	1806303	1806303	1806303

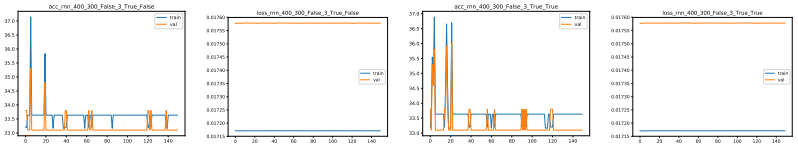
Table 4: Accuracy on Validation Set and Number of Parameters When We Vary Combining Methods, and Fix the Others.



(a) False, True, CNN (b) False, True, CNN (c) False, False, CNN (d) False, False, CNN (e) True, False, CNN (f) True, False, CNN



(g) True, True, CNN (h) True, True, CNN (i) False, True, RNN (j) False, True, RNN (k) False, False, RNN (l) False, False, RNN



(m) True, False, RNN (n) True, False, RNN (o) True, True, RNN (p) True, True, RNN

Figure 4: Accuracy and Loss on Validation and Train Set When We Vary Regularization Methods, and Fix the Others.

1.5 Best Hyperparameters and Test Accuracy

After ablation hyperparameters search, we have the best hyperparameters combination for both CNN and RNN. CNN has lower number of trained parameters but has lower accuracy. They are:

- **CNN** `hidden_size` = 300, `is_concat` = True, `kernel_size` = 3, `is_wd` = False, `is_dropout` = False. **Validation Accuracy: 65.0**
- **RNN** `hidden_size` = 400, `is_concat` = False, `is_wd` = False, `is_dropout` = True. **Validation Accuracy: 73.9**

1.6 List 3 Correct Predictions and 3 Incorrect Predictions

Correct Predictions.

- **Entailment**. 'bicycles stationed while a group of people socialize .' 'People get together near a stand of bicycles .'
- **Contradiction**. 'Man in white shirt and blue jeans looking to the side while walking down a busy sidewalk .', 'Man has a blue shirt on .'
- **Neutral**. 'A little boy watches a Ferris Wheel in motion .', 'A boy is waiting in line for the Ferris Wheel .'

Incorrect Predictions.

- **Entailment predicted as Contradiction** 'Two people are in a green forest .', 'The forest is not dead .' **Possible Reason**: Since 'dead' and 'green' are opposite words, it is possible for the network not to learn that the 'not dead' and 'green' are the same; and we do not use bi-gram.
- **Contradiction predicted as Neutral**'Two women , one walking her dog the other pushing a stroller .', 'There is a snowstorm .' **Possible Reason**: This contradiction is not obvious. The network could hardly get that 'walking a dog' is not possible in a snowstorm. There is no obvious contradicted word or phrase pair.
- **Neutral predicted as Contradiction**'A group of numbered participants walk down the street together .', 'Participants wait for the beginning of the walkathon .' **Possible Reason**: 'wait' and 'walk down the street' looks like a contradicted pair.

1.7 Evaluating on MultiNLI

There are five different genres: 'fiction', 'telephone', 'slate', 'government', and 'travel'. We show the accuracy for different genes. We test our best RNN and CNN model on the MultiNLI validation set. The results are shown in 5. The accuracies on the new dataset is much lower than the old dataset. And the accuracies from different genres are similar but not the same. Some of the genres have higher accuracy for RNN and some have higher accuracy for CNN. They do not overlap. The conclusion is that the trained model learned may not have very good performance on a dataset with different distribution.

Genre	fiction	telephone	slate	government	travel
Accuracy (RNN)	41.2	42.0	38.1	39.5	39.7
Accuracy (CNN)	39.2	37.2	39.5	37.6	39.0

Table 5: Accuracy on MultiNLI Validation Set for best CNN and RNN model.

2 Bonus: Fine tune on new data set

We load the best RNN model and train it for 5 epochs to fine-tune for each genre. We get accuracy in Table 6. We also show the accuracy for training without loading the pretrained model (Plain Training). It is very bad. So Fine-Tuning does play an important part in training on small dataset.

Genre	fiction	telephone	slate	government	travel
Accuracy (RNN)	41.2	42.0	38.1	39.5	39.7
Accuracy (RNN) Fine Tune	45.3	50.7	43.7	53.3	49.7
Accuracy (RNN) Plain Training	34.5	36.1	35.9	29.9	36.0

Table 6: Accuracy on MultiNLI Validation Set for best CNN and RNN model.