Natural Language Inference on RNN/CNN Models (DS-GA-1011)

Ziv Schwartz[1]

[1]New York University

October 30, 2018

I. Abstract

GitHub Repository: https://github.com/zivschwartz/Natural-Language-Inference-with-RNN-CNN

This assignment deals with constructing an optimal Recurrent and Convolutional Neural Network (RNN/CNN) model that can take on the Stanford Natural Language Inference (SNLI) task. The SNLI dataset contains two pieces of text: a premise and a hypothesis. After training a model, predictions will be able to be made labeling the premise as **entailing** the hypothesis, **contradicting** the hypothesis, or neither *entailing* nor *contradicting* and thus being **neutral** to it. As this task is based on natural text, ambiguity will be introduced regarding the labeling of each premise and hypothesis pair. The SNLI dataset contains 100,000 training instances and 1,000 validation instances. After the best model is built, testing on the Multi-Genre Natural Language Inference (MNLI) – a variant of SNLI which covers spoken and written text from different sources, or "genres" – is performed to determine how well the optimal model performs on different genres. The MNLI dataset contains 5,000 validation instances. The Master Code can be found at my GitHub repository.

II. Data Preprocessing and Vocabulary Embedding

The Data is distributed having already gone some preprocessing (separated by punctuation and apostrophes). Each sentence is then loaded and tokenized based off whitespace, splitting sentence pairs into pairs of word lists. The premise and hypothesis labels are recoded into numerical values with **contradiction = 0, neutral = 1, entailment = 2**. Following this, a vocabulary embedding was built using the FastText pre-trained word vector sets. Using the SNLI dataset, a PyTorch training and validation data loader were constructed to train on both a Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) and test validation accuracies.

III. Baseline Models on SNLI Dataset

Default parameters for both RNN and CNN models: embedding size=100, hidden size=200, number of layers=1, number of classes=3, vocab size= id-to-word length, learning rate = 3e-4, number of epochs = 5, criterion = Cross Entropy Loss, optimizer = Adam.

Training on accuracies on local machine (Could not set up Google Cloud or Prince): Very long run times. To remain consistent, model accuracies are from taken from the maximum value of the series.

A. RNN

The baseline RNN model is built in a similar fashion to the one presented in lab 4, only that there is double the input data (two sentences: premise and hypothesis). For RNN, we utilize a single-layer bi-directional Gated Recurrent Unit (GRU). Since the task utilizes pairs of sentences (premise & hypothesis) and a bi-directional GRU, hidden size increases by four times the original output. Another important feature of the RNN is that when using a padded sequence for the vector representations, the sentences have to be fed into the encoder in descending sorted order. After doing this, it is necessary to re-sort the sentence pairs to their original pairings before concatenating the final state of the encoder. This single vector concatenation is then passed through two fully-connected layers (with ReLU activation in between) and passed through a testing model. Given the size of the data and since I used a local machine, training and validation took a considerable amount of run time and I had to interrupt my kernel after about 90 iterations (trained over-night). Refer to Figure 1 for accuracies over each iteration. Max accuracies: **Training Accuracy: 38.968, Validation Accuracy: 41.3** (a lot of fluctuation present). I tried several variations in trying to improve RNN accuracies but due to run-time and memory constraints this is the best accuracy recorded.

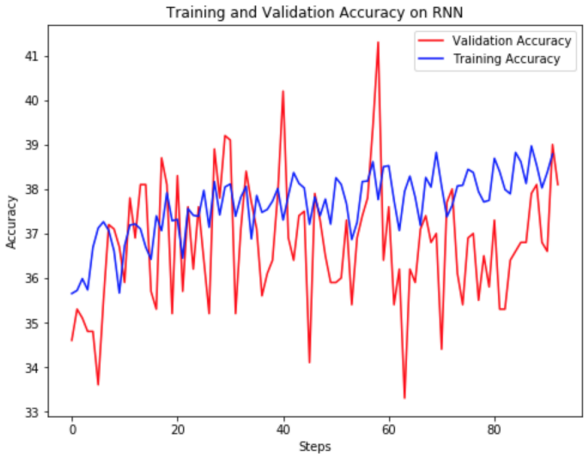

Figure 1: Training and Validation Accuracy by step on the RNN model: a lot of variation present in the accuracies

B. CNN

The baseline CNN model is built in a similar fashion to the one presented in lab 4, only that there is double the input data (two sentences: premise and hypothesis). For CNN, we implement a two layer one-dimensional convolution with ReLU activations. Since the input data is in pairs of sentences, the hidden size increases by two times the original output. With CNN, sorting the sentences and padding the vectors is not required. Instead, we perform max-pooling on each particular sentence within the pair before concatenating the final state representations into a single vector. This concatenation is then passed through two fully-connected layers (with ReLU activation in between) and later passed through a testing model. Given the size of the data and since I used a local machine, training and validation took a considerable amount of time and I had to interrupt my kernel after about 50 iterations (taking over 3 hours). Refer to Figure 2 for accuracies over each iteration. Max accuracies: **Training Accuracy: 66.998, Validation Accuracy: 62.1.** This model results in far better validation accuracy than the RNN model.

IV. Hyperparameter Tuning

Taking the best model so far, CNN, we tune certain parameters to see if the validation accuracy increases.

Hyperparameter: Hidden Size (Default = 200)

Test Hidden Size = 400. Max accuracies: **Training Accuracy: 64.562, Validation Accuracy: 61.8.**

The computation for these accuracies also took a considerable amount of time (3+ hours) for even fewer iterations, around 30 instances and the validation accuracies were very similar, however staying consistent, we will stick with hidden size of 200 as it resulted in a higher max validation accuracy. A closer look into the different effects of hyperparameters. Hidden size is one of the hyperparameters that depends on the size of the dataset. Often increasing the hidden size of the layers will lead to overfitting. For this example, doubling the default hidden size does not alter accuracy by that much. Kernel size is another hyperparameter which can be tuned, it refers to the convolutional correlation, indicating the height by width filter mask size. Increasing kernel size reduces computation time and the number of neurons in the subsequent layer of the neural network, thus losing out on more details. A smaller kernel size gives a lot more details to the layer and may also result in overfitting. Another hyperparameter which can be implemented to avoid overfitting is regularization, in particular dropout and weight decay. During each iteration of training, the dropout layer (either in input or hidden layer) randomly removes particular nodes within the network. Weight decay refers to implementing either L1 or L2 norm penalties to certain nodes.

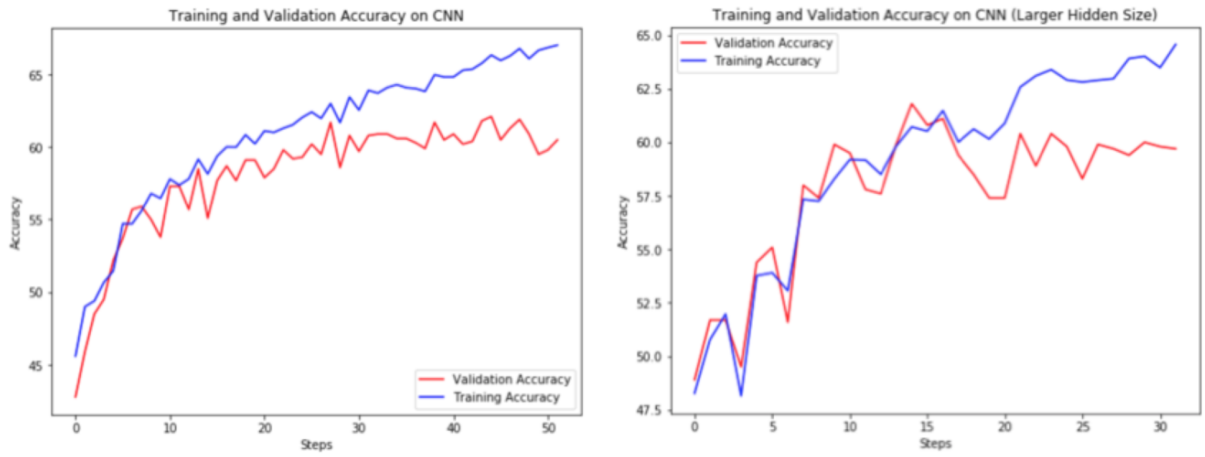

Figure 2: Side by side plots of training and validation accuracy on CNN model with hidden size = 200 and hidden size = 400

V. Model Predictions

Using the optimal model we can also determine the correctness of the predictions on the validation dataset. Looking at the first 8 values of the predicted values to the validation target datasets, we observe the following:

- predictions = (**0**,0,1,**2**,1,**2**,1,1)
- labels_val[:8] = (**0**,2,2,**2**,2,**2**,2,0)

Matching the values between the two determines the level of correctness.

Correct predictions: 1st, 4th, and 6th pairing

Incorrect predictions: 2nd, 3rd, and 5th pairing

The model could have predicted these wrong due to the difference in size between the training and validation sets.

The actual reviews and their polarity can be found in the Master Code file in the GitHub repository.

VI. Evaluating on MNLI Dataset

After establishing the optimal model, it is then evaluated on an entirely new dataset, the Multi-Genre Natural Language Inference (MNLI) – a variant of SNLI which covers spoken and written text from different sources, or "genres". There are five genres present in the MNLI dataset: fiction (995 instances), government (1016 instances), slate (1002 instances), telephone (1005 instances), and travel (982 instances).

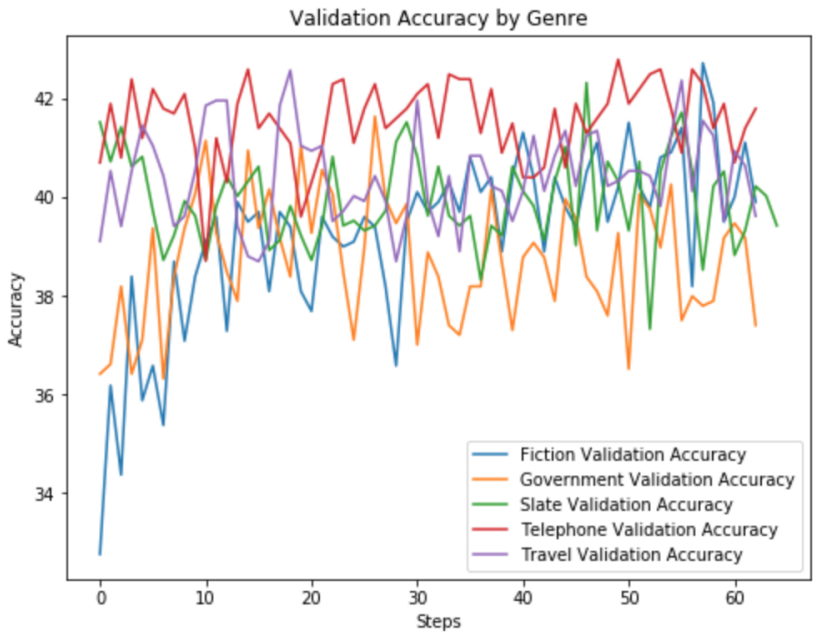

Figure 3: Validation Accuracy by MNLI genres

From the plot, there is a lot of overlap between the validation accuracies of the different genres. Looking solely at the trend of validation accuracies over all steps of training the model, it seems that the Telephone genre has the highest accuracies over the steps. The Government genre, on the other hand, seems to have the worst validation accuracy over the steps of the different types of genres.

Max validation accuracies per genre over steps:

- Fiction: 42.714
- Government: 41.634
- Slate: 42.315
- Telephone: 42.786
- Travel: 42.566