RNN/CNN-based natural language inference

(DS-GA 1011, assignment 2)

Melanie Tosik (`mt3685`)

October 2018

1 Overview

For the PyTorch implementation, please see: `https://github.com/melanietosik/nl-inference`.

In the experiments, we evaluate the following hyperparameters:

- Learning rate: `1e-3, 5e-4, 1e-4`
- Size of the hidden dimension
  - `CNN: 50, 100, 200, 500`
  - `RNN: 25, 50, 100, 250`
- Dropout probability: `0.0, 0.2, 0.5`
- Kernel size (`CNN`): `1, 2, 3`

The best `CNN` model is trained using the following set of hyperparameters:

- ⋆ Learning rate: `1e-3`
- ⋆ Size of the hidden dimension: `500`
- ⋆ Dropout probability: `0.0`
- ⋆ Kernel size: `2`

The best `CNN` model achieves `71.5` accuracy on the SNLI validation set and consists of `1303003` trained parameters (cf. `cnn.pt.txt` and `log.cnn_best.txt`).

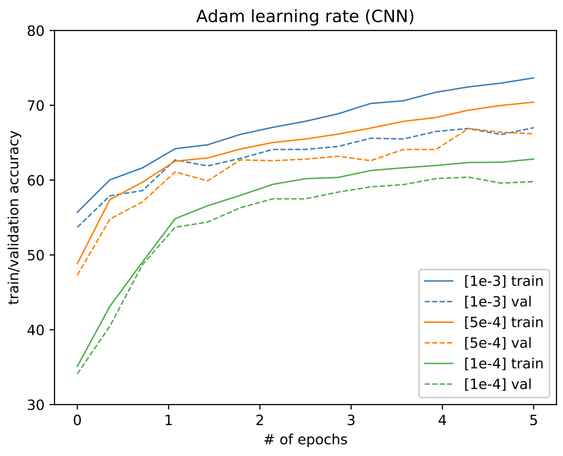
The best `RNN` model is trained using the following set of hyperparameters:

- ⋆ Learning rate: `1e-3`
- ⋆ Size of the hidden dimension: `250`
- ⋆ Dropout probability: `0.0`

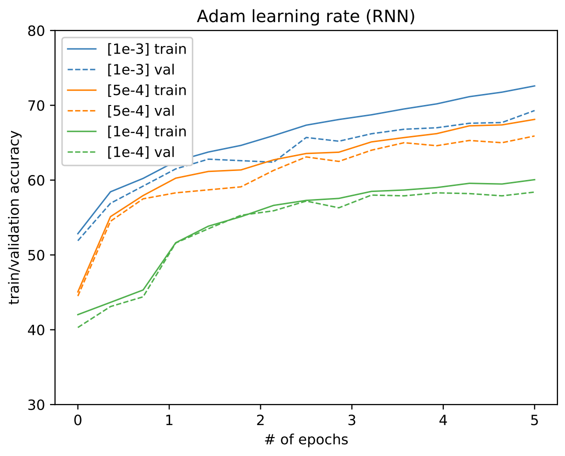
The best `RNN` model achieves `72.8` accuracy on the SNLI validation set and consists of `1079003` trained parameters (cf. `rnn.pt.txt` and `log.rnn_best.txt`).

All models are trained using a batch size of `256`, `8` worker threads, an embedding dimension of size `300`, and a log interval of `100`. Except for the final/best model experiments, the models are trained for `5` epochs.

The word embeddings are the 1 million word vectors trained on Wikipedia 2017 released as part of `fastText` (cf. `wiki-news-300d-1M.vec.zip` at `https://fasttext.cc/docs/en/english-vectors.html`).

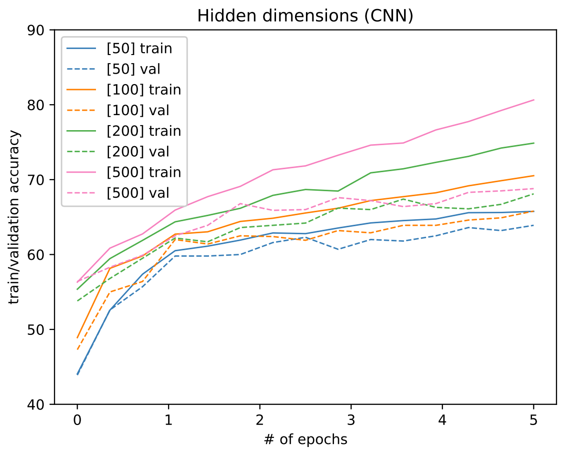


(a) Learning rate (CNN model)

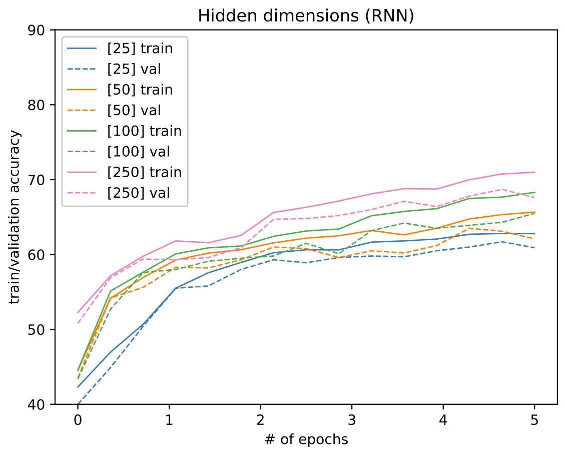


(b) Learning rate (RNN model)

Figure 1: Learning rate for the Adam optimizer.



(a) Hidden dimensions (CNN model)



(b) Hidden dimensions (RNN model)

Figure 2: Size of the hidden dimension.

2 Results

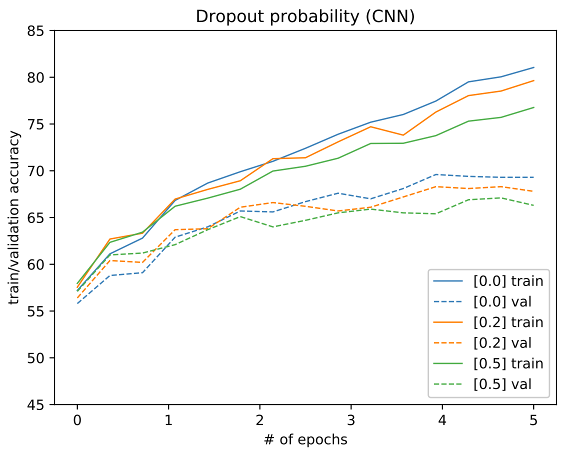
Given the page limit, we can only briefly discuss the most relevant results. Please refer to the `logging/` and `plots/` folders in the GitHub repository for the full set of experiments and results.

2.1 SNLI

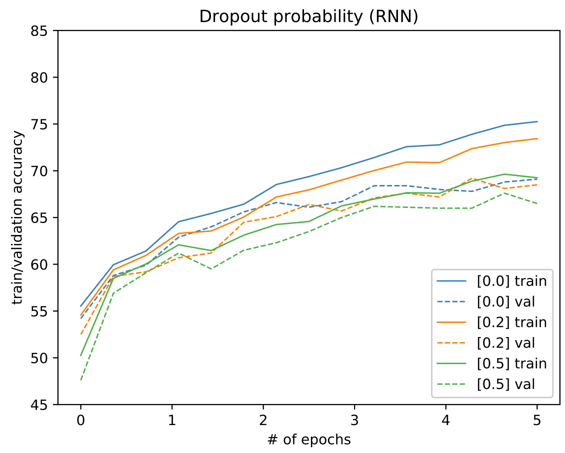
We start by fine-tuning the **learning rate** for the Adam optimizer (`torch.optim.adam`). Figure 1 illustrates the results for three different learning rates. For both the `CNN` and the `RNN` model, the PyTorch default of `1e-3` seems to work best. Most of the training and validation accuracy curves increase steadily at approximately the same rate, indicating that the model is not overfitting to the training data too much.

Next, we can experiment with the **size of the hidden dimension**. The original SNLI paper[1] specifies a hidden dimensionality of 100. Therefore, for the `CNN` model, we evaluate `50, 100, 200, 500`. Since the `RNN` is bi-directional, we choose slightly lower dimensionalities of `25, 50, 100, 200`. The results for both models are shown in Figure 2. Both the `CNN` and the `RNN` model seem to benefit from a larger hidden size in terms of validation accuracy. The `CNN` model also starts overfitting the larger the size of the hidden dimension.

[1]`https://nlp.stanford.edu/pubs/snli_paper.pdf`



(a) Dropout probability (CNN model)



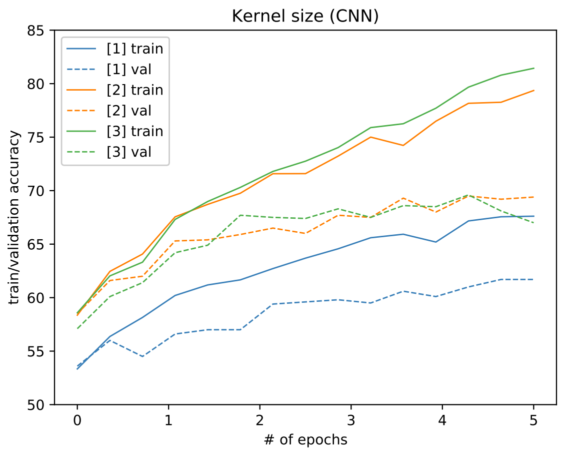
(b) Dropout probability (RNN model)

Figure 3: Dropout probability

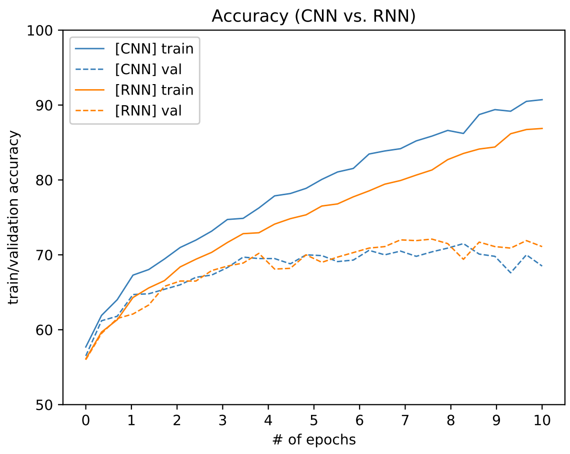
To recap, our two best models are currently trained for `5` epochs, using the Adam optimizer with a learning rate of `1e-3` and a hidden dimensionality of `500` and `250` for the `CNN` and `RNN` model, respectively.

Next we experiment with **dropout**, a regularization technique that randomly zeros out some of the neurons in the network during the forward pass in order to reduce co-adaptability and increase the ability to generalize to unseen data. Figure 3 illustrates the results when using a dropout probability of `0.0, 0.2, 0.5`. Somewhat surprisingly, adding a dropout layer did not improve model accuracy for either model.

For the final experiment, we try out different **kernel sizes** for the `CNN` model only. The results are illustrated in Figure 4 (a). Using a kernel size of `1` does not seem to work well at all. The default kernel size of `3` from the previous lab causes the model to overfit—the validation accuracy starts dropping around the beginning of the last epoch while the training accuracy keeps increasing. A kernel size of `2` performs somewhere in between the two previous settings and therefore seems to be the best choice for now. Intuitively, using bigrams over trigrams also seems fitting given how short most sentences in the SNLI dataset are.



(a) Kernel size (CNN model)



(b) Best CNN vs. best RNN model

Figure 4: Kernel size (CNN) and best model performance (CNN vs. RNN)

Finally, Figure 4 (b) illustrates the training and validation accuracies for the best `CNN` and the best `RNN` model after hyperparameter tuning. The `CNN` model is still overfitting, but achieves `71.5` accuracy on the SNLI validation set when using early stopping. The `RNN` model learning curves seem a little more robust, which is reflected in a final accuracy of `72.8` on the SNLI validation set.

	Premise	Hypothesis	Relation
✓	The little boy is jumping into a puddle on the street .	The boy is outside .	*entailment*
✓	An older couple is resting on a bench .	Two people are sitting next to each other .	*entailment*
✓	There is a street with buildings and one man in black walking on the side of the road .	The buildings are corporate offices .	*neutral*
✗	An African American woman with 2 young girls .	There is a mother with her daughters .	*neutral*
✗	A farmer gives a tour of his farm to local families .	A farmer selling his farm .	*neutral*
✗	Female runners from Japan , Germany and China are running side by side .	The runners are from the US .	*contradiction*

Table 1: Correct and incorrect predictions of the best model (`RNN`) on the SNLI validation set.

Since there is no room for additional tables, please refer to the logs in the `logging/` directory for the accuracy, loss, and number of trained parameters for each individual model and experiment. The details of the two best models are provided in `cnn.pt.txt` and `rnn.pt.txt`. The best model overall is the `RNN` model.

Finally, we can use the `id2token` mapping to reconstruct validation samples for which our best model made correct or incorrect predictions. Table 1 shows three examples of correct and incorrect predictions.

It seems noteworthy that the incorrect predictions all have a *neutral* or *contradiction* relation. Since the labels in the given SNLI data sets are fairly evenly distributed, this could be an indicator that *entailment* relations between the premise and hypothesis are generally easier to classify than *neutral* or *contradiction* pairs. Many of the *neutral* sentence pairs are also ambiguous and therefore difficult to classify correctly. For example, given "An African American woman with 2 young girls.", I would likely assume that "There is a mother with her daughters." as well.

The third incorrect sample prediction on the other hand should have been possible to classify. I think in this case, the model might have failed to disambiguate between the pronoun "us" and the noun "us", i.e. the lowercase token for the country "US". In the future, it would probably make sense to lowercase only the beginning of each sentence during data preprocessing instead of the entire sentence.

2.2 MultiNLI

In order to evaluate the best SNLI models on the MultiNLI dataset, we can first split the `mnli_val.tsv` file into subsets for each genre using the `split_mnli.py` script. Afterwards, `run_mnli.py` will load the best version of the specified model (`CNN` or `RNN`) and evaluate the accuracy on the genre-specific validation set.

Table 2 contains the validation accuracy for each genre for each model. In general, the SNLI models perform much worse (approx. 30%) on the MultiNLI data sets.

	CNN model	RNN model
Fiction	43.42	47.94
Government	43.70	48.23
Slate	43.31	43.41
Telephone	45.87	47.76
Travel	44.50	46.13

Table 2: Validation accuracies on the MNLI dataset

Without fine-tuning the SNLI models on the domain-specific data sets, these results are hardly surprising. Overall, the `RNN` model outperforms the `CNN` model on every single genre. This is in line with our previous observation that the `CNN` model might be overfitting to the SNLI training data and thus might not be able to generalize to new data sources as well as the `RNN` model. Among the 5 genres, it seems that the model accuracies are highest for the *Telephone* category. One plausible explanation for this could be that the SNLI premise-hypothesis sentences resemble simple, short, spoken language, and as such are most similar to the models' original training data.