Deep Learning - Comparison of DNN models for email classification

Evgeny Manturov

May 12, 2024

1 Environment and Hardware

The environment used for this project is a **Python 3.11.8** installation with **IPython 8.20**. All the requirements of the virtual Anaconda environment are contained in **conda-requirements.txt** and can be installed along with the new venv using:

```
conda create -n env_name --file conda-requirements.txt
```

Training the model on CPU is lengthy and inefficient. In order to make the training orders of magnitude shorter, it is recommended to use an nVidia GPU with at least 12Gb of VRAM and CUDA capability. This requires CUDA and cuDNN installed.

2 Dataset

The dataset is a .csv file containing 5172 samples of email word counts. Each sample contains email number, 3000 word count of most frequent words among all emails, and a label indicating if the email is a spam email or not.

3 Preprocessing

The preprocessing steps include:

1. Removal of email number column - it replicates the index and serves no purpose in future classification
2. Removal of duplicate rows - "dumb" oversampling is useless here is the same email can appear in multiple batches anyway - 4631 rows remain
3. Outliers with >2000 repetitions of the same word are not removed - all emails are Min-Max scaled using the standardisation algorithm. This step is mandatory and reduces the metrics significantly if not performed.
4. Since the models' output layers are the pairs of sigmoid-activated values, representing the probability of the email to belong to class 0 (not spam) and class 1 (spam) respectively (which add up to 1), the label is divided into 2 columns, representing if the email is spam or not spam.
5. Validation dataset is selected to ensure the models perform well on the data never seen before. This step can be omitted if the model is in the inference mode when running test data, and test data is never a part of train data. Validation fraction is selected to be 10% of the dataset (463 rows).
6. The remaining dataset is divided into train and test sets, and both feature sets are Min-Max standardised.

The total dataset balance was analysed - there are 3170 emails classified as non-spam and 1461 emails classified as spam. The dataset is balanced enough, and the metric considers precision and recall for both predicted classes.

4 Common model attributes

Some parameters are selected independent of the neural network selected. The task is binary classification, so:

- Binary cross-entropy loss was selected
- F1 score metric was selected, as required by the task
- Sigmoid probability output layers were selected instead of logits to identify how certain the model is in its selection, if needed.
- Adam loss function with weight decay (L2 regularized) was selected
- 2 Dropout layers with 0.2 dropout rate each were inserted into each model to prevent overfitting
- Both models were trained with train and tested with test dataloaders with 64 and 128 batch size.

5 Linear NN model

The first model tested is the linear model consisting of 3 dense layers with 6000 perceptrons each, activated by simple ReLU activation functions. The first maximum test F1 was achieved after 5 epochs with batch size of 128 and was recorded as 96.90%. Considering anti-overfitting measures, it was safe to train the model for 3 more epochs to increase robustness. Validation F1 score was recorded to be 0.96 or 96%, which is high enough for production use.

6 Convolutional NN model

The second model tested is the 1D-convolutional model with Convolutional and Pooling layers, activated intermittedly by Gaussian error linear units. The first maximum test F1 was achieved after 53 epochs with batch size of 128 and was recorded as 91.17%. Considering anti-overfitting measures, it was safe to train the model for 11 more epochs to increase robustness. Validation F1 score was recorded to be 0.86 or 86%, which is not high enough for production use (See section 7).

7 Conclusion

The models used for this binary classification task are very different in structure and have their pros and cons:

- Convolutional NN is less accurate on validation dataset than Linear NN, which is a substantial drawback of it for production use ($\approx 10\%$ flat difference in F1 score)
- Convolutional NN is ≈ 500 times lighter in storage than Linear NN (500kB vs 215MB), which means the CNN may be deployed easier and on devices with little storage capacity.
- Convolutional NN is x4.4 slower to train than Linear NN, considering the number of epochs each NN requires to achieve maximum test F1 (3.6s vs 16s, 8 epochs vs 64 epochs).

Convolutional NNs have the property of remembering and using location of the object scanned by the kernel. The kernel also scans expressions (word sequences of the length of the kernel itself). This implies that CNN must be more effective if an actual email is given to it, with preprocessing performed (removal of punctuation, casefold and word stemming/lemmatizing). For this kind of data, Linear NN is better in terms of F1 score. Quantization can be applied to it to reduce its storage weight.