TDT4171 — Artificial Intelligence Methods
Assignment 8 - Deep Learning

Erik Storås Sommer - 535006

March 2023

Exercise - FNN and RNN with Keras

In this assignment, the objective was to implement and train feedforward and recurrent neural networks using Keras. Several different architectures and hyperparameters were tested to evaluate their performance. This report presents the results from theise decisions.

Hyperparameters

The hyperparameters used for the two neural networks are presented in figure 1. I found little to no difference in accuracy when adding more than 2 hidden layers. The activation function used for the hidden layers is relu, since it is the most common activation function for hidden layers and switching to others did not make a big difference in accuracy. The activation function for the LSTM layer is sigmoid and the recurrent activation function is tanh. The default values for these two parameters are the oposite, but I found better results switching them. Forthermore, since the target for the dataset is either 0 or 1, the output layer has only one neuron using the sigmoid activation function. The loss function used is binary crossentropy, and the optimizer is adam. All are suitable for this type of problem. Additionally, the average length of the training data was utilized to set the maxlen parameter, which is used to pad the training and test data to a uniform length.

```
46    # Hyperparameters
47    input_length = data["x_train"].shape[1]
48    input_dim = data["vocab_size"]
49    hidden_dim = 64
50    hidden_layers = 2
51    output_dim = 1
52    activation_fnn = 'relu'
53    activation_rnn = ['sigmoid', 'relu']
54    recurrent_activation_rnn = 'tanh'
55    output_activation = 'sigmoid'
56    loss = 'binary_crossentropy'
57    optimizer = 'adam'
58    metrics = ['accuracy']
59    epochs = 2
60    dropout = 0.2
```

Figure 1: Hyperparameters used for the neural networks

Results

By using the parameters in figure 1, the model was able to achieve a test accuracy of above 90% on both the feedforward and recurrent neural network. The time it took to train the feedforward model was greatly lower than the recurrent model, but the recurrent model was able to achieve a higher accuracy. This is due to the fact that the recurrent model uses previous input to make a more educated prediction, while the feedforward model is only able to use the current input. The results are presented in figure 2.

```
eriksommer@dhcp-10-22-39-132 deep_learning % python3 deep_learning.py
2023-03-14 13:29:49.001492: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow
PU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
1. Loading data...
2. Preprocessing data...
3. Training feedforward neural network...
2023-03-14 13:29:59.923560: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow
PU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/2
12283/12283 [==============================] - 47s 4ms/step - loss: 0.2460 - accuracy: 0.8961
Epoch 2/2
12283/12283 [==============================] - 47s 4ms/step - loss: 0.2016 - accuracy: 0.9167
4079/4079 [==============================] - 4s 992us/step - loss: 0.2183 - accuracy: 0.9099
Model: Feedforward NN.
Test accuracy: 0.910
4. Training recurrent neural network...
Epoch 1/2
12283/12283 [==============================] - 255s 21ms/step - loss: 0.2206 - accuracy: 0.9091
Epoch 2/2
12283/12283 [==============================] - 256s 21ms/step - loss: 0.1539 - accuracy: 0.9384
4079/4079 [==============================] - 32s 8ms/step - loss: 0.1459 - accuracy: 0.9412
Model: Recurrent NN.
Test accuracy: 0.941
```

Figure 2: Output from running the attached python file