Artificial Neural Network and Deep Learning

Homework 2

Angelo Capponcelli Pasquale Castiglione Mehdi Samimi

Dataset

Data Exploration

Not having enough information about the dataset, we performed data exploration. In particular, after printing the summary statistics, we plotted the distribution of the classes and some random samples in order to extract something valuable.

Class Imbalance

From the histogram of the of the classes, a problem of class imbalance was spotted. We tried to overcome this problem using different techniques. Firstly we tried to set the *class_weight* parameter in the *Model.fit()* function, then we tried to random *oversample* the smaller classes adding noise to the new samples and finally we tried *data augmentation* using the library *tsaug*[1].
Unfortunately none of the approaches listed above gave us the desired results, so we stayed with the original dataset.

Preprocessing

The dataset was firstly shuffled and then split into training set and validation set. During development a ratio of 75/25 turned out to be the best choice. In the data exploration phase we observed that values were widely scattered from -10000 to $+50000$ so we decided to standardize[2] them in order to shift and scale the input into a distribution having mean 0 with standard deviation 1. We performed the standardization directly inside the model adding a *keras.layers.Normalization* layer after the input layer. For this computation we used the mean and the variance of the training dataset.

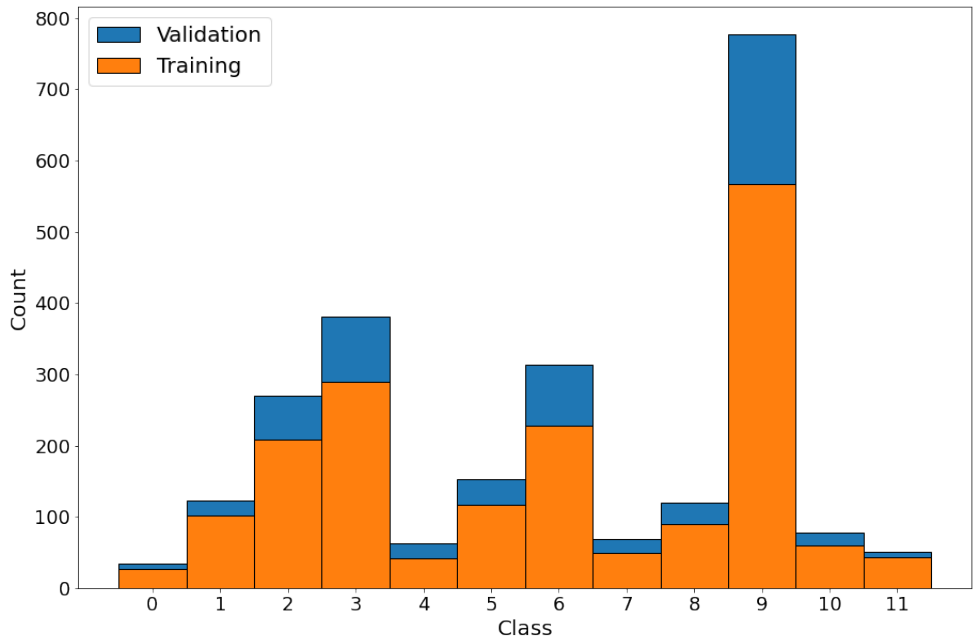

Figure 1: Histogram of the classes

[1] `https://tsaug.readthedocs.io/en/stable/`
[2] $X_{new} = \frac{X-\mu}{\sigma}$

Model Development

First Models

We started with a simple model composed of a stack of three blocks each one containing a *Conv1D* layers with 128 filters of 3x3 each, followed by a *MaxPooling1D* layer. The last block was directly connected to the output layer. This first model led us to an accuracy of ≈ 0.65.
Willing to improve the accuracy score we made the model more complex by adding two *Bidirectional LSTM* with 128 units each just before the output layer. This latter approach led us to an accuracy score of ≈ 0.69.

Overfitting

As the accuracy score went up, the model started to suffer overfitting so, in order to overcome this, we implemented some expedients.
Firstly we added a *Dropout* layer with a rate of 0.5 just before the output layer, then we reduced the units of the LSTM layers from 128 to 100, then we added a *Dropout* layer with a rate of 0.3 after each convolutional block and finally we set the *recurrent_regularizer* parameter in the LSTMs to *keras.regularizers.L1L2(1e-2, 1e-2)*. After some trials, keeping only one *Bidirectional LSTM* showed better result so we decided to remove the second one. After using all of these, we were able to reduce the overfitting but also to increase the accuracy.

Final Model

Therefore the final model was composed of a feature extraction part, a recurrent part and the output layer:

- **Feature Extraction Part:** Composed of a stack of three blocks, each one composed of a *Conv1D*, a *MaxPooling1D* and a *Dropout*.
- **Recurrent Part:** Consisted of a *Bidirectional LSTM* layer with 100 units by a *Dropout* layer.
- **Output:** A single dense layer with 12 units and a softmax activation function.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization (Normalizatio  (None, 36, 6)            0
 n)

 conv1d (Conv1D)             (None, 36, 128)           2432

 max_pooling1d (MaxPooling1D  (None, 18, 128)          0
 )

 dropout (Dropout)           (None, 18, 128)           0

 conv1d_1 (Conv1D)           (None, 18, 128)           49280

 max_pooling1d_1 (MaxPooling  (None, 9, 128)           0
 1D)

 dropout_1 (Dropout)         (None, 9, 128)            0

 conv1d_2 (Conv1D)           (None, 9, 128)            49280

 max_pooling1d_2 (MaxPooling  (None, 4, 128)           0
 1D)

 dropout_2 (Dropout)         (None, 4, 128)            0

 bidirectional (Bidirectiona  (None, 200)              183200
 l)

 dropout_3 (Dropout)         (None, 200)               0

 dense (Dense)               (None, 12)                2412

=================================================================
Total params: 286,604
Trainable params: 286,604
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Summary of the Final Model

Training

For the training, after some experiments, a *batch_size* of 64 resulted to be the right trade off between overfitting and accuracy. The training lasted 100 epochs and we made use of two callbacks: *early stopping* with a patience of 10 in order to avoid overfitting and to reduce waste of time and *ReduceLROnPlateau* with a patience of 5 and a minimum learning rate of 1e-6 to improve the learning. With all these tweaks, we were able to reach an accuracy score of ≈ 0.72 on the training dataset and an accuracy score of ≈ 0.73 on the test set in Codalab.

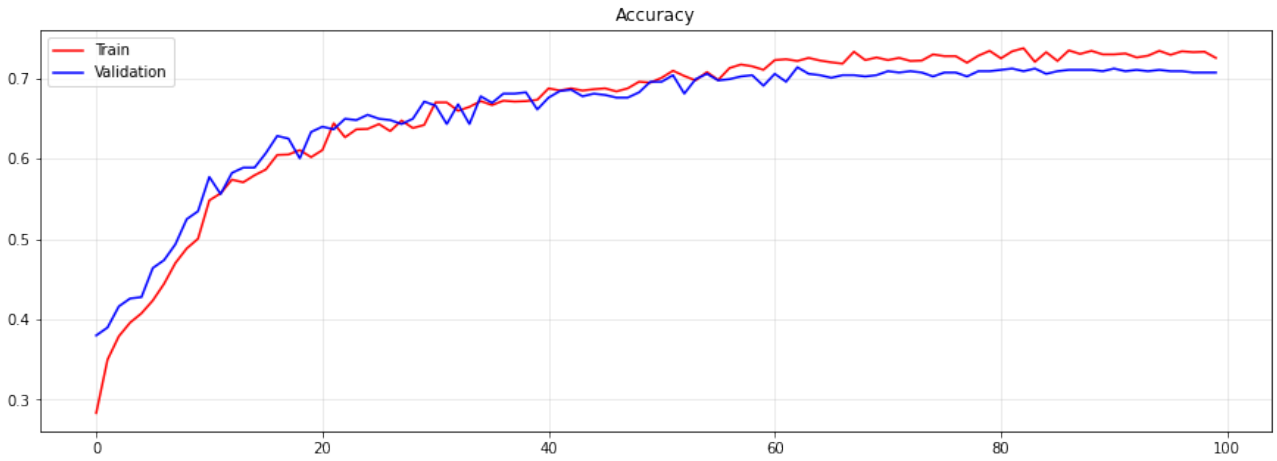

Figure 3: Accuracy

Conclusion

The model turned out to be pretty much robust, showing the same behavior with both the training set and test set. Furthermore it was also a relatively simple model that allowed to have little training times.