Residual Neural Network: Time Series Classification

Artificial Neural Networks and Deep Learning - a.y. 2022/2023

Marco Bendinelli - 10673478 - *M.Sc. Computer Science and Engineering* -
Marco Cayuela - 10859184 - *M.Sc. Mathematical Engineering* -
Pietro Andrea Cirino - 10628055 - *M.Sc. Mathematical Engineering* -
at Politecnico di Milano

Zero Neurons Networks CodaLab group

Abstract

In this report, we explore and compare the development process we followed to design a Residual Neural Network able to classify 12 different time series classes.

1. Introduction The objective of this project is to correctly classify samples in the multivariate time series format. For this purpose, we studied 3 main models: the 1D Convolutional Neural Network, the Bidirectional LSTM Network and the Residual Network. For each of them, we started from a basic implementation and we tried to optimize it in order to exploit the most of their potential.

2. Pre processing We tried 2 main normalization techniques, but none of those produced an increase in the score. In particular, we tried the *Z-score* and the *MinMaxScaler* normalization (for further information see MinMaxScaler and Zscore). Besides normalization, during this phase we split the data set in 2 sub sets, training and validation, with percentage 80-20% and using the stratified sampling technique, since several classes had fewer elements than others.

3. 1D Convolutional

3.1. Features Extractor The feature extractor of the model is composed by:

- **1D convolution layer** with the dimensionality of the output space equal to 128 and window length equal to 3. We specified ”same” as padding and used the ReLu activation function.
- **MaxPooling1D layer**
- **1D convolution layer** with the same characteristics as the first one.
- **GlobalAveragePooling1D layer**
- **Dropout layer** with rate equal to 0.5 in order to prevent overfitting.

3.2. Classifier We decided to keep this part of the model very simple: the classifier is composed by a dense layer with 128 neurons and ReLu activation function, while the output layer is just a dense layer with Softmax activation function.

3.3. Improvements The described configuration performed badly (61.5 %), so we tried to give more complexity to the model by adding more Convolution and MaxPooling layers with different configurations. However, we were not able to improve the performance, the model became slower and was not able to overcome an accuracy of 60 %.

4. BiLSTM

4.1. Features Extractor We used a bidirectional-LSTM model. We chose an architecture with 2 Bi-LSTM layers, the first one using the full sequence while the second one using only the last output of the sequence, that results in a 128-dimensional vector. The best sizing we found via hyperparameters tuning was 256 neurons for the first one and 128 for the second one.

4.2. Improvements

- **Classifier:** Adding a 128-neuron dense layer did not change the accuracy and in some cases decreased it.
- **Dropout:** We added dropout in the BiLSTM layers. Without dropout the model resulted in overfitting, hence we finally chose a 0.5 ratio for dropout.
- **Batch size:** Changing batch size did not change the final accuracy but influenced the rapidity of convergence. The best batch size was 512.

The final Bi-LSTM model resulted in an accuracy of 67.9 %.

5. Residual Network

5.1. Features Extractor Finally, as an evanescent hint suggested, we decided to build a ResNet-style model with 1D convolution layers, obtaining better results then before. The network we built consists of three residual blocks, each one composed of three convolutions whose output is then added to the residual block's input and then fed to the next layer. The number of filters for convolutions is set to 64 for the first block and 128 for the others, with the ReLU activation function that is preceded by a batch normalization operation. In each residual block the filter's length is set to 3, 5 and 5 respectively for the first, second and third convolution. We used Keras tuner to tune these hyperparameters via random search.

5.2. Global Average Pooling Layer This layer is introduced on top of the model in order to avoid the usage of the Flatten layer, being the latter heavier to manage in terms of complexity and parameters. We also tried a Global Max Pooling Layer which provided less good results.

5.3. Classifier Starting from a simple GAP + Softmax layer as classifier, we tried to add a Dense layer in order to improve the predictive abilities of our model. However, even with a small number of neurons (16, 32, 64) we observed that the Dense layer caused the model to overfit, lowering the accuracy. Since even adding a Dropout layer did not prevent the overfitting, we decided to keep the classifier as simple as possible, with just a Softmax layer after GAP.

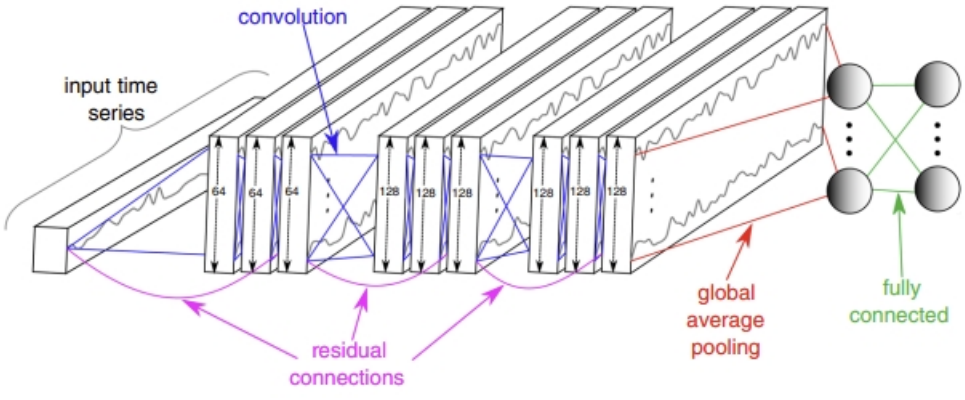


Fig. 1: The Residual Network's architecture

6. Conclusion

6.1. Final Model The final model submitted has a total number of parameters equal to 576,268; its architecture is depicted in Figure 1

The main characteristics of the model are the following:

- **batch_size** : 512
- **loss** : Categorical Crossentropy
- **optimizer** : Adam(1e-3)

and the general structure is:

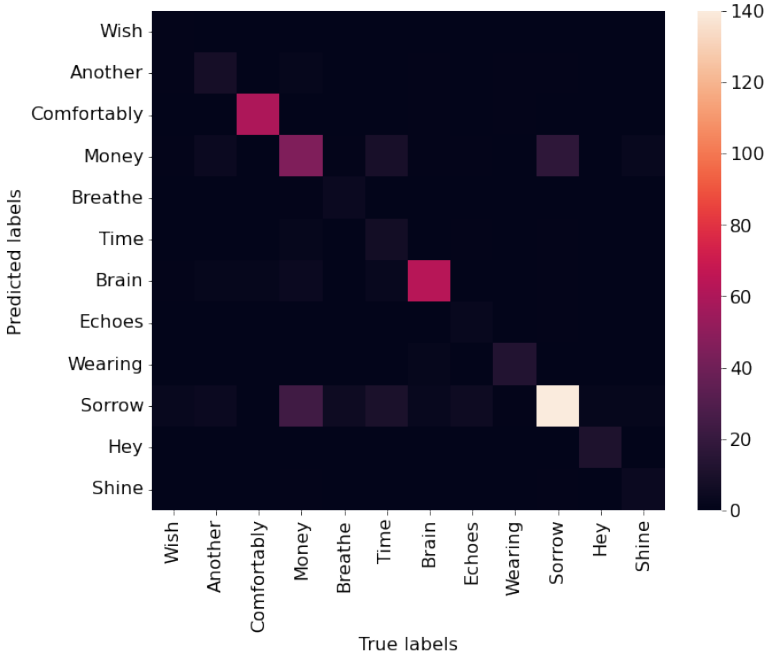
Layer (type)	Output Shape	Param
`Input`	[(None, 36, 6)]	0
`Residual Block 1`	(None, 36, 64)	45312
`Residual Block 2`	(None, 36, 128)	231936
`Residual Block 3`	(None, 36, 128)	297472
`global_avg_pool`	(None, 128)	0
`softmax`	(None, 12)	1548

6.2. Performance The dataset has been split in 80% for training and 20% for validation. The model in training, evaluated with an Early Stopping with patience 50 epochs based on validation accuracy, reached the peak of performance at epoch 156 with the following indexes:

Train. loss	Train. acc.	Val. loss	Val. acc.
0.0511	99.28%	1.0963	73.87%

6.3. Confusion Matrix Finally, we generated the confusion matrix to identify the correctness of classification for each class with F1-score, precision and recall, on the dataset split in 70% for training and 15% each for validation and testing. Testing set performance indexes:

- **Accuracy** : 72.43%
- **Precision** : 77.83%
- **Recall** : 57.67%
- **F1-score** : 62.58%



6.4. Leadboard Evaluation

- Development phase accuracy : **73.76%**
- Final phase accuracy: **72.82%**