ANNDL – Challenge Two

team_durian

Stefano Maxenti – 10526141 – 970133 Riccardo Mencucci – 10534455 – 970801

Introduction

We are provided with a dataset containing a timeseries made up of 7 features (*Sponginess*, *Wonder level*, *Crunchiness*, *Loudness on impact*, *Meme creativity*, *Soap slipperiness*, *Hype root*). We try different approaches and, eventually, our best model reaches a RMSE score of **3.82** and MAE score of **2.37**. In the ZIP file there are the different notebooks used; binary models are available on Codalab and on this link.

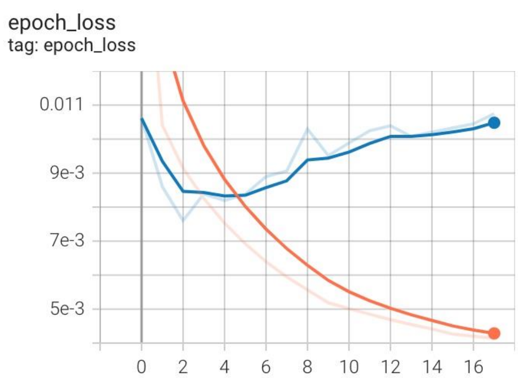
Data pre-processing

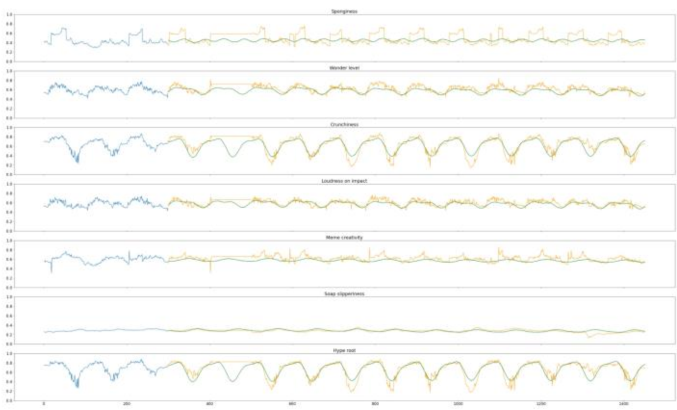
In most experiments, we normalize each feature independently with its own minimum and maximum value. We then apply an inverse transformation to the predicted data to obtain the denormalized predictions for error evaluation. We experiment with other normalization techniques, such as the Yeo–Johnson power transformation, single mean and variance common to all features and independent standard scale (normalize each feature independently with its own mean and variance), but they did not lead to a meaningful performance change. We use independent normalizations to account for the diversity of ranges for the various features, to easily explain the model and because of its low computational cost.

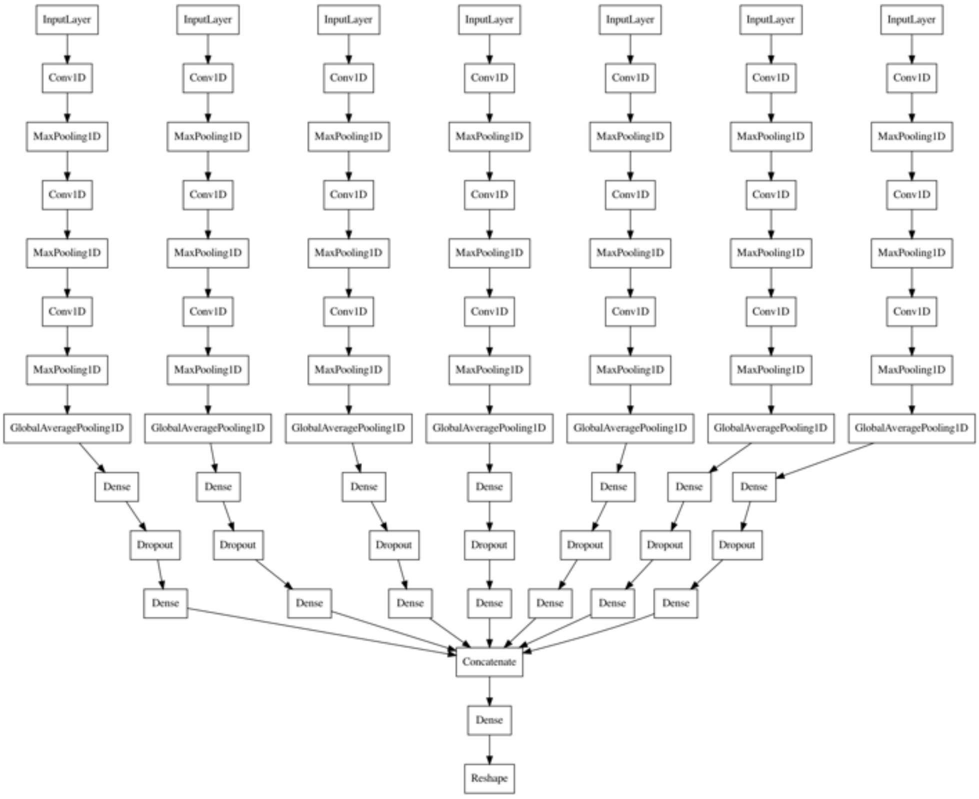
Custom models

Multi-input-CNN

This model gives us the best score in Codalab, achieving RMSE equal to 3.82. To develop this model, we decide to proceed with separate convolutional stacks of repeated Convolution, Activation, Maxpooling independent for each feature. We then concatenate the 7 previous stacks, and we add a dense layer with size $1152 \cdot 7$ before the Reshape layer. We predict 1152 values as originally requested in Phase 2 of the challenge. We use large convolutional filters to widen the receptive field without losing any information about rapidly changing features. Experiments with dilated convolutions and smaller filters lead to worse results.
Since there may be correlation between the different features, an output dense layer after the concatenation is needed to better back-propagate the error of the final prediction to all the features.
The input is pre-processed into 7 tensors with shape $(None, window, 1)$. We try different window sizes and eventually we choose a size of 300 with stride equal to 1. This gives us more than 60.000 training windows that can be used to provide the requested forecasting.
We do not use memory cells in this model. The approach is completely convolutional, and it proves to be incredibly fast using graphics card even with more than 36 millions parameters. As one can see, convergence is almost immediate.







ConvLSTM

Starting from the model introduced in class, we develop two alternatives. Both are based on the combination of LSTM, GRU and Convolution. The main difference between them is the number of parameters in the memory cells. This influences the training time, which goes from 1 minute per epoch to over 5 minutes per epoch.

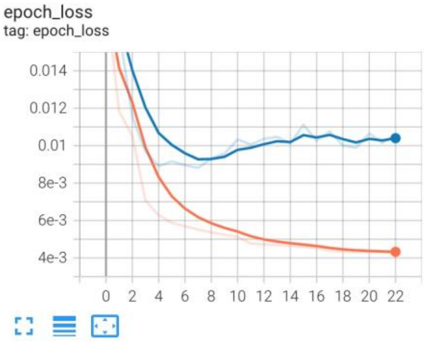
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (InputLayer)           [(None, 300, 7)]          0
_________________________________________________________________
bidirectional (Bidirectional (None, 300, 256)          139264
_________________________________________________________________
conv1d (Conv1D)              (None, 300, 128)          98432
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 150, 128)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 150, 512)          788480
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 150, 256)          393472
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 75, 256)           0
_________________________________________________________________
gru (GRU)                    (None, 75, 128)           148224
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 75, 512)           197120
_________________________________________________________________
global_average_pooling1d (Gl (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 1024)              525312
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 8064)              4136832
_________________________________________________________________
reshape (Reshape)            (None, 1152, 7)           0
=================================================================
Total params: 6,951,936
Trainable params: 6,951,936
Non-trainable params: 0
_________________________________________________________________
```

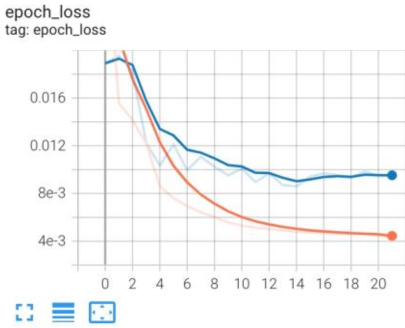
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (InputLayer)           [(None, 300, 7)]          0
_________________________________________________________________
bidirectional (Bidirectional (None, 300, 2048)         8454144
_________________________________________________________________
conv1d (Conv1D)              (None, 300, 128)          786560
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 150, 128)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 150, 2048)         9445376
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 150, 256)          1573120
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 75, 256)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 75, 512)           393728
_________________________________________________________________
global_average_pooling1d (Gl (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 1024)              525312
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 8064)              4136832
_________________________________________________________________
reshape (Reshape)            (None, 1152, 7)           0
=================================================================
Total params: 25,839,872
Trainable params: 25,839,872
Non-trainable params: 0
_________________________________________________________________
```

Two experiments are notable with this general idea:

- ConvLSTM_small: RMSE 4.04, about 7 millions parameters
- ConvLSTM_BIG: RMSE 4.01, about 26 millions parameters

Notably, in ConvLSTM_BIG we use bidirectional LSTM cells with size 1024. Considering the small increment in RMSE with respect to ConvLSTM_small, we think that this model is not as valuable in a cost-benefit perspective.

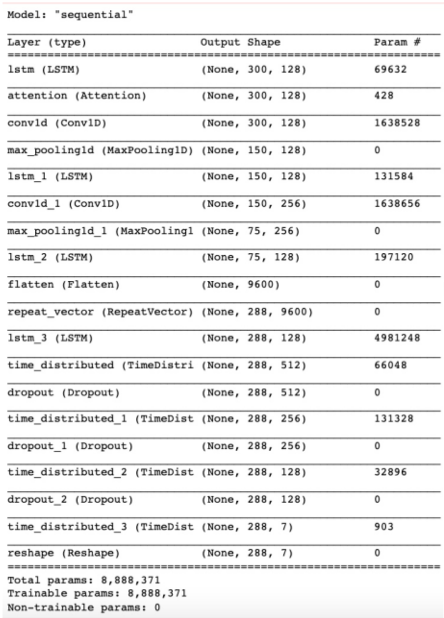




Autoregressive encoder-decoder

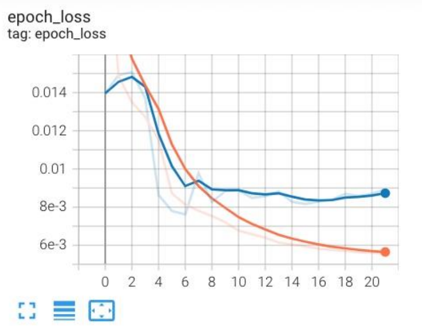
In our experiments, autoregression does not prove to be a better solution than directly forecasting the total number of values, because of error amplifications. Nevertheless, here we show a possible (bad) implementation directly based on the encoder-decoder architecture: we combine LSTM and Convolutional layers in the encoder section; we use the RepeatVector layer to replicate the input vectors in the decoder, made up of TimeDistributed Dense layers. Since it is autoregressive, we append the predicted 288 values to the following input for three times, in order to get the required 864 values. We try to use an Attention layer as well.

We obtain RMSE equal to 4.75 and notice that the training is longer than the pure convolutional model even with much fewer parameters, due to the presence of Attention and memory cells.


Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 300, 128)	69632
attention (Attention)	(None, 300, 128)	428
conv1d (Conv1D)	(None, 300, 128)	1638528
max_pooling1d (MaxPooling1D)	(None, 150, 128)	0
lstm_1 (LSTM)	(None, 150, 128)	131584
conv1d_1 (Conv1D)	(None, 150, 256)	1638656
max_pooling1d_1 (MaxPooling1	(None, 75, 256)	0
lstm_2 (LSTM)	(None, 75, 128)	197120
flatten (Flatten)	(None, 9600)	0
repeat_vector (RepeatVector)	(None, 288, 9600)	0
lstm_3 (LSTM)	(None, 288, 128)	4981248
time_distributed (TimeDistri	(None, 288, 512)	66048
dropout (Dropout)	(None, 288, 512)	0
time_distributed_1 (TimeDist	(None, 288, 256)	131328
dropout_1 (Dropout)	(None, 288, 256)	0
time_distributed_2 (TimeDist	(None, 288, 128)	32896
dropout_2 (Dropout)	(None, 288, 128)	0
time_distributed_3 (TimeDist	(None, 288, 7)	903
reshape (Reshape)	(None, 288, 7)	0

Total params: 8,888,371
Trainable params: 8,888,371
Non-trainable params: 0



Conclusion and what we learnt

One of the biggest challenges we face is not having any information on the nature and origin of the data. The features were renamed to meaningless names, which as a side effect makes it hard to identify trends, make reasonable assumptions or otherwise reason about the data with human intuition. We have to blindly rely on deep learning and its ability to automatically extract features from any dataset. All our models implement general-purpose architectures, which we are not able to design with the application domain in mind. The results show that deep learning works, in that it's able to generalize from the input data, but also that deep learning alone is insufficient for a prediction with a small error over a long prediction horizon.

In our trials, we find out that convolutional neural networks are good tools to understand trends and to learn from sequences. They are very fast to train on graphics cards and tends to converge in few epochs before overfitting. LSTM, on the other hand, are theoretically more powerful in learning from sequences (they are used in more advanced approach such as seq2seq) but they are slower to train because of their intrinsic complexity.