Artificial Neural Networks and Deep Learning
Homework 2

Nicola Dean	Marco Fasanella	Raffaello Fornasiere	Christian Spano
10674826	10617541	10790353	10823764
nicola.dean @mail.polimi.it	marco.fasanella @mail.polimi.it	raffaello.fornasiere @mail.polimi.it	christian.spano @mail.polimi.it

1 Introduction

CHRISTIAN + MARCO TIME SERIES MARCO

Type of Text	Font Size	Style
paper title	15 pt	bold
author names	12 pt	bold
author affiliation	12 pt	
the word "Abstract"	12 pt	bold
section titles	12 pt	bold
document text	10 pt	
captions	10 pt	
abstract text	10 pt	
bibliography	10 pt	
footnotes	9 pt	

Table 1: Sizes and styles of fonts used.

2 Preprocessing

When it comes to TimeSeries classificaion/forecasting, preprocessing is a crucial step to grant optimal results. In this section will be described all the preprocessing techniques we tried or discarded.

2.1 Normalization

After visualizing the data, we noticed that samples has different scale factor based on targets. To avoid advantaging some of the classes we decided to normalize the dataset.

2.1.1 Per Column

As first approach, we normalized the dataset by **FULL columns**

- **MinMax** Does not work properly; When applied, the validation accuracy dropped to 33
- **Mean / Std** This technique does not improve the accuracy on vanilla models.

2.1.2 Per Sample

The next method we tried, was to normalize each sample (TimeSerie) independently, to completly remove the different scales of the targets.

- **MinMax** Improve performances but not enough, at least on vanilla models
- **Mean / Std** This was the **Game Changer** for us, it improves the performances on Vanilla models from 63% to 67%

2.2 Augmentation

In the Homework1 challenge, we noticed the importance of augmentation on this field and so we decided to give it a try also on time series. We tried both "by hand" and Library approach with the following results.

2.2.1 Numpy

Using the function *np.random.normal* it is possible to generate random noise with a certain mean std and shape. To obtain augmented samples we copied the dataset multiple time, and then, for each time serie, we calculated it's standard deviation and applied the augmentation like follow:

$$x = x + np.random.normal(0, std, x.shape) * w$$

Figure 1: With x be a single time series, w be a weight of our choice

The result of augmantation was immediate and bring us a solid 68% using the vanilla BiLstm

2.2.2 TSUNG/Librery

RAFFAELLO

2.3 Seasonal + Trend preprocess

Christian

2.4 Expanding Window size

2.5 Adding New Features

Christian

3 Vanilla Models

MARCO

4 Net Concatenations

4.1 LSTM + CNN

4.2 CNN + LSTM

5 Heterogeneus Layers

5.1 LSTM + CNN

5.2 CNN + LSTM

5.3 CNN + DENSE

5.4 ALTRI DI RAFFAELLO

6 Our best Model

Unexpectedly bla bla bla

7 Conclusion

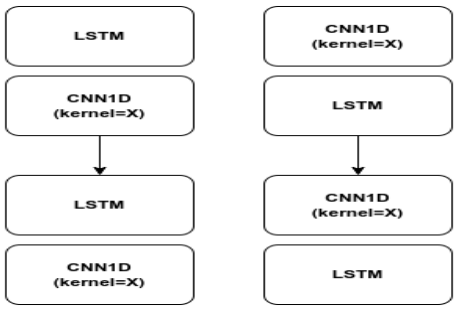

Figure 2: LSTM-CNN and CNN-LSTM model schematics

8 Adapt 2D models to 1D convolutions

8.1 InceptionNet

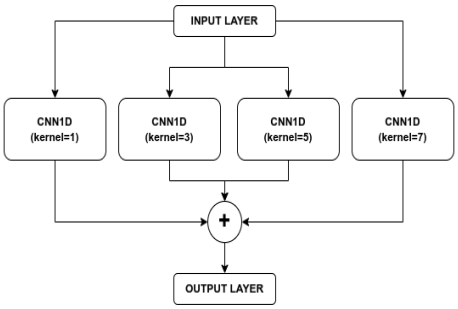

Figure 3: CNN1D Inception Like Net

8.2 Resnet

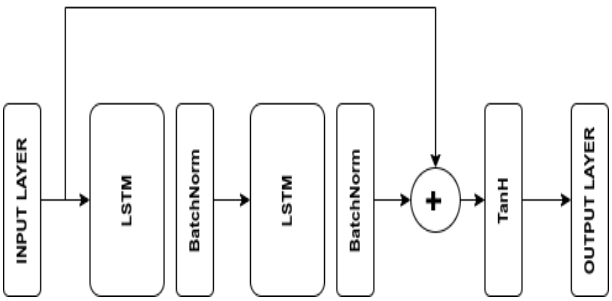

Figure 4: Resnet Like LSTM Net