ACA – Project P17

Authors: Roberto Bigazzi 899411 – Filippo Collini 905628

Professor: Cristina Silvano – Project Instructor: Ahmet Erdem

Project Description

Name: Tensorflow Neural Network Complexity Scaling
Code: P17
Type: Programming(Python)
Description:
Tensorflow being a high-level python framework mainly for Neural Network applications may behave quite differently based on the complexity of application requirements. The main concern is upon reaching some certain thresholds, it may not perform well or its performance scaling may not be as desirable as before. Therefore, the objective of this project is the explore and discover the ranges of parameters related to computation complexity and make an analysis on the scaling of the execution time of the inference phase (not the training phase).
The generation of benchmarks must be automated as much as possible and it must be configurable with a config-file. The suggested degrees of freedoms are ”number of layers”, ”size of convolutions”, ”size of input”, ”batching of input”, etc. The expected deliverables of the project are:
Configuration files for benchmark generation.
Python scripts that run the Tensorflow benchmarks.
Analysis results with graphs to justify the claims.

Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

Tested Hardaware

We used 2 machines to run more differentiated tests:

- OS: Ubuntu Linux 19.04
- Processor: Intel Core i7-4710HQ
- GPU: NVIDIA GeForce GTX 860M 2048 MB
- RAM: 16 GB

And

- OS: macOS Mojave 10.14.6
- Processor: Intel Core i5-5257U
- GPU: Intel Iris Graphics 6100 1536 MB
- RAM: 8 GB

Models built

- **Fashion MNIST Classifier**

Model that will learn to classify fashion clothes from Zalando database.

- **Text Classifier**

Model that will predict if a review of a movie is a good or a bad review.

Degrees of freedom

- Size of input
  - 28x28x1 grayscale images (Fashion MNIST)
  - 56 characters consisting in a truncated review
- Number of layers
  - 4, 6, 8, 10
- Size of batching
  - 1, 4, 16, 64
- Size of convolutions
  - 3, 5, 8

MNIST Model


```
Model with:
    batch size: 1
    number of layers: 4
    convolution size: 3
Model structure:

Tensor shape: (None, 28, 28, 1)
---------------------------------- Layer: input_1
Tensor shape: (None, 28, 28, 1)
---------------------------------- Layer: conv2d_1
Tensor shape: (None, 26, 1, 64)
---------------------------------- Layer: max_pooling2d_1
Tensor shape: (None, 1, 1, 64)
---------------------------------- Layer: flatten_1
Tensor shape: (None, 64)
---------------------------------- Layer: dense_1
Tensor shape: (None, 64)
---------------------------------- Layer: dropout_1
Tensor shape: (None, 64)
---------------------------------- Layer: dense_2
Tensor shape: (None, 10)
```

Input: 28x28x1 grayscale images

Model made of:

- Convolutional Layer
- Pooling Layer
- 2 Fully Connected Layers

With 6, 8, 10 total layers we'll add every time a Convolutional and a Pooling layer.
Always present the Flatten, Dense and Dropout layers.

Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

Text Classification Model


```
Model with:
    batch size: 1
    number of layers: 4
    convolution size: 3
Model structure:

Tensor shape: (None, 56)
---------------------------------- Layer: input_1
Tensor shape: (None, 56)
---------------------------------- Layer: embedding_1
Tensor shape: (None, 56, 256)
---------------------------------- Layer: reshape_1
Tensor shape: (None, 56, 256, 1)
---------------------------------- Layer: conv2d_1
Tensor shape: (None, 54, 1, 64)
---------------------------------- Layer: max_pooling2d_1
Tensor shape: (None, 1, 1, 64)
---------------------------------- Layer: flatten_1
Tensor shape: (None, 64)
---------------------------------- Layer: dense_1
Tensor shape: (None, 64)
---------------------------------- Layer: dropout_1
Tensor shape: (None, 64)
---------------------------------- Layer: dense_2
Tensor shape: (None, 2)
```

Input: 56 characters consisting in a truncated review

Model made of:

- Embedding Layer
- Reshape Layer
- Convolutional Layer
- Pooling Layer
- 2 Fully Connected Layers

With 6, 8, 10 total layers we'll add every time a Convolutional and a Pooling layer.
Always present the Flatten, Dense and Dropout layers.

Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

Training phase

The model has been trained for all the combinations of our degrees of freedom with these specifications:

- Epochs: 10
- Learning rate: 0.001
- Dropout probability: 0.75

It was evaluated in the inference phase with 10 iterations.

At the end we'll output a .csv file headed by Layers, Batch Size, Convolution Size, Total Time and Mean Time which we'll be our so called *features* for the data analysis.

Total Time is the entire time that takes the inference.

Mean Time is the mean of the inference times in the 10 iterations.

Config file

```
1  [CONFIGURATION]
2  Training_Iterations = 10
3  Batch_Size = 1 4 16 64
4  Tot_Layers = 4 6 8 10
5  Filter_Size = 3 5 8
6  Learning_Rate = 0.001
7  Dropout = 0.75
8  Evaluation_Iterations = 10
9
```

All of the degrees of freedom are configurable with this config file present in the project folder.

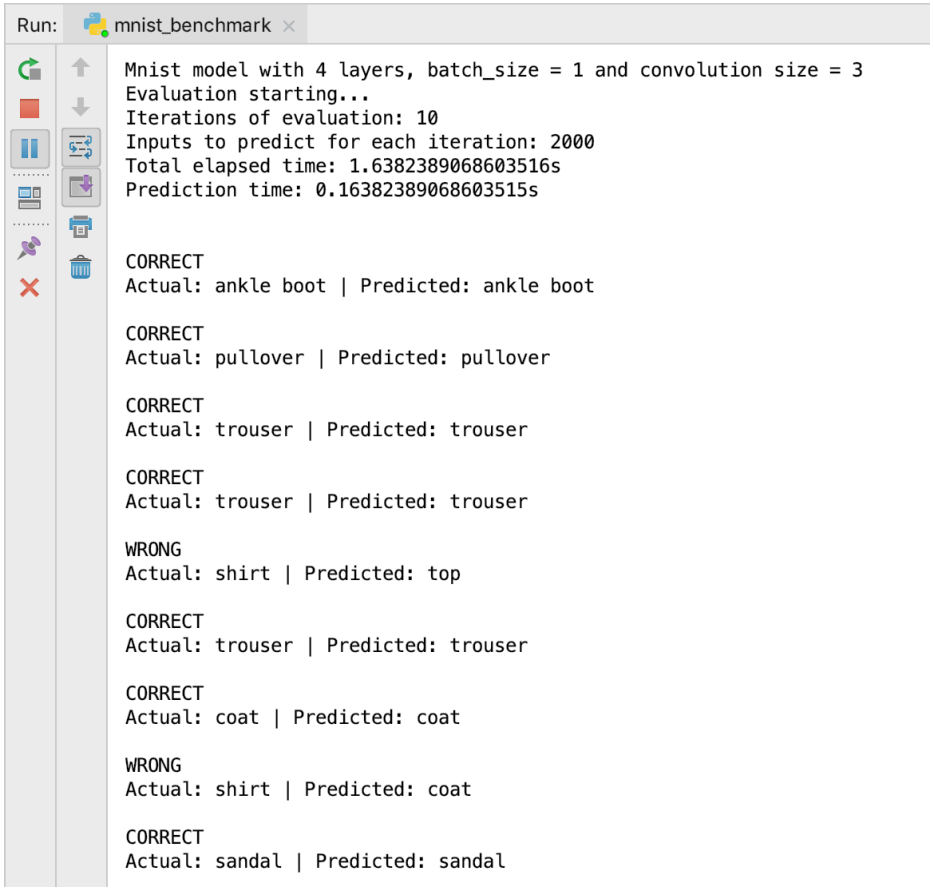
Benchmarks

Both the benchmarks, besides of writing the final .csv file, will output the model structure (as shown in the previous slides) and for each iteration:

- The sizes of the degrees of freedom of that specific iteration
- The total elapsed time
- The prediction time
- The prediction that the model made next to the actual value to be predicted

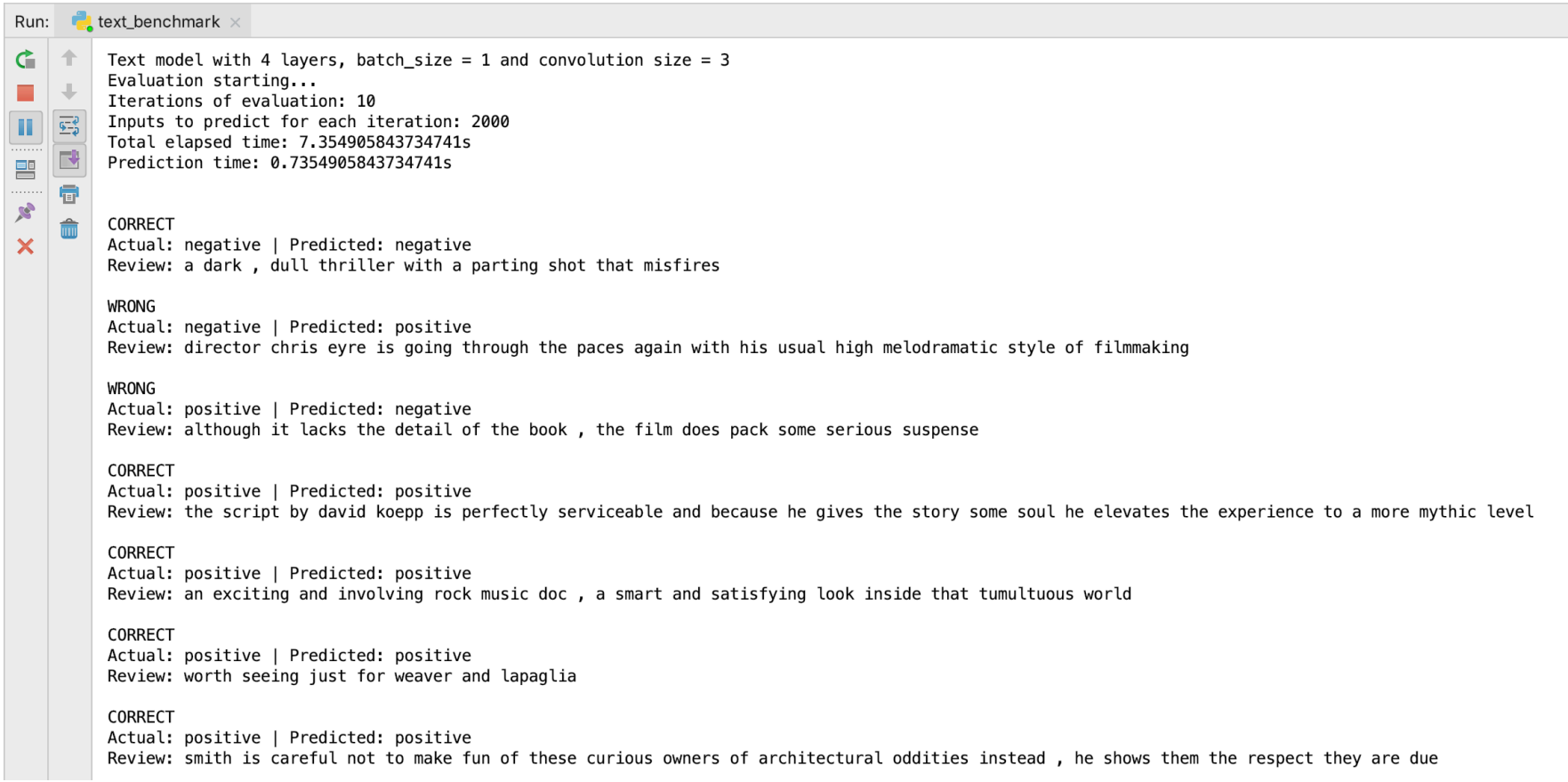
In the next slides we can see the two examples.

Benchmarks



Fashion MNIST benchmark output

Benchmarks



Text Classifier benchmark output

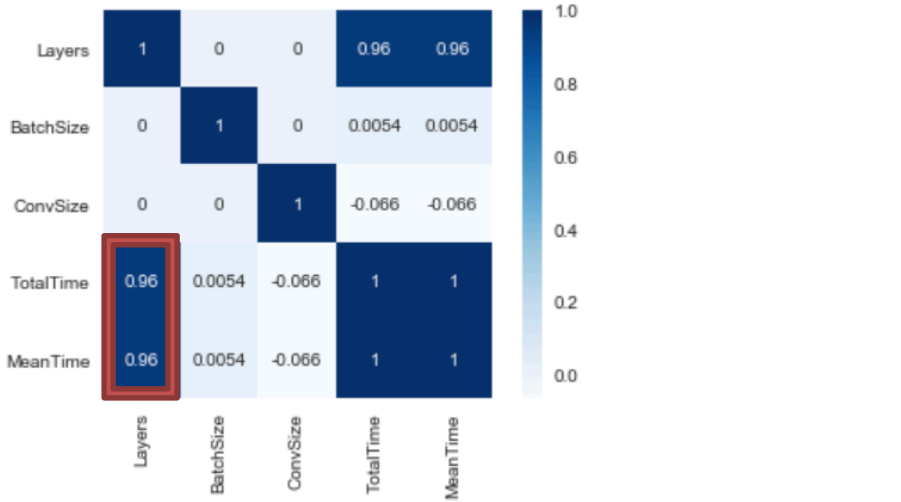
Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

Analysis of the results – Correlation Matrix

Fashion MNIST

If we plot the correlation matrix between all our features we can spot a very high correlation between the number of layers and the time features. This means that the inference time is driven principally by the number of layers in the model.

```
In [3]: cov=data.corr(method='pearson')
        sns.heatmap(cov,square=True,annot=True,cmap="Blues")

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x1a141ac860>
```



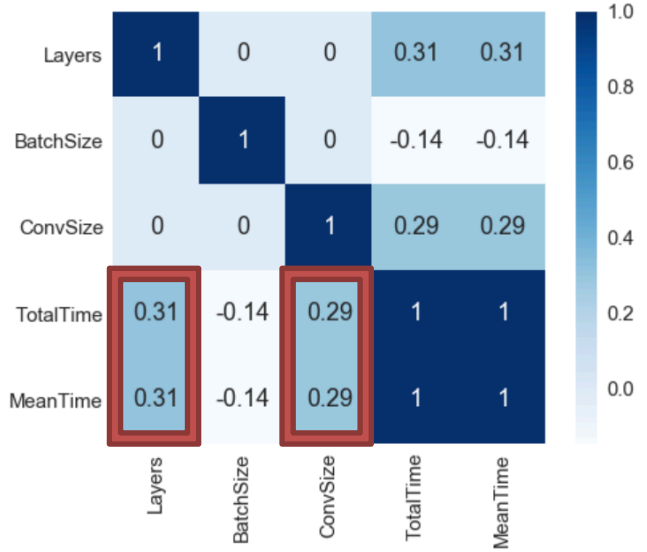
Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

Analysis of the results – Correlation Matrix

Text Classifier

Here we cannot spot very high correlations like we did for the Fashion MNIST model. In this case the time features are equally correlated to both Number of Layers and Convolutional Size.

```
In [9]: cov=data.corr(method='pearson')
        sns.heatmap(cov,square=True,annot=True,cmap="Blues")
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1a24459d30>
```



Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

Analysis of the results - Boxplots

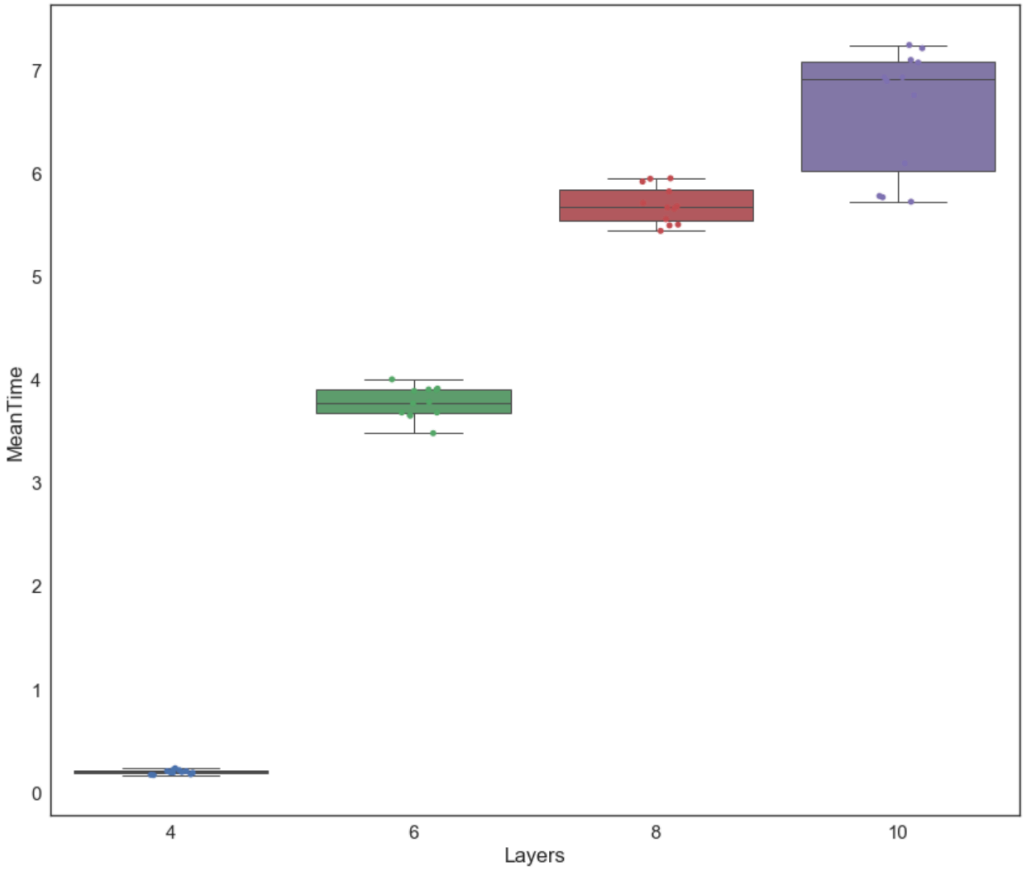
```
In [4]: matplotlib.rcParams['figure.figsize'] = (14.0, 12.0)
sns.set_context("notebook", font_scale=1.5, rc={"lines.linewidth": 1})

ax = sns.boxplot(x="Layers", y="MeanTime", data=data)
ax = sns.stripplot(x="Layers", y="MeanTime", data=data, jitter=True, edgecolor="gray")
```

Fashion MNIST

Here we plotted some boxplots to show how are distributed the values of the Mean inference Time for each number of layers involved.

We can clearly see how the Mean inference Time increases in like a logarithmic way.

It's interesting even the fact that, with more layers, also the variance of the values of Mean Time increases.



Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

Analysis of the results - Boxplots

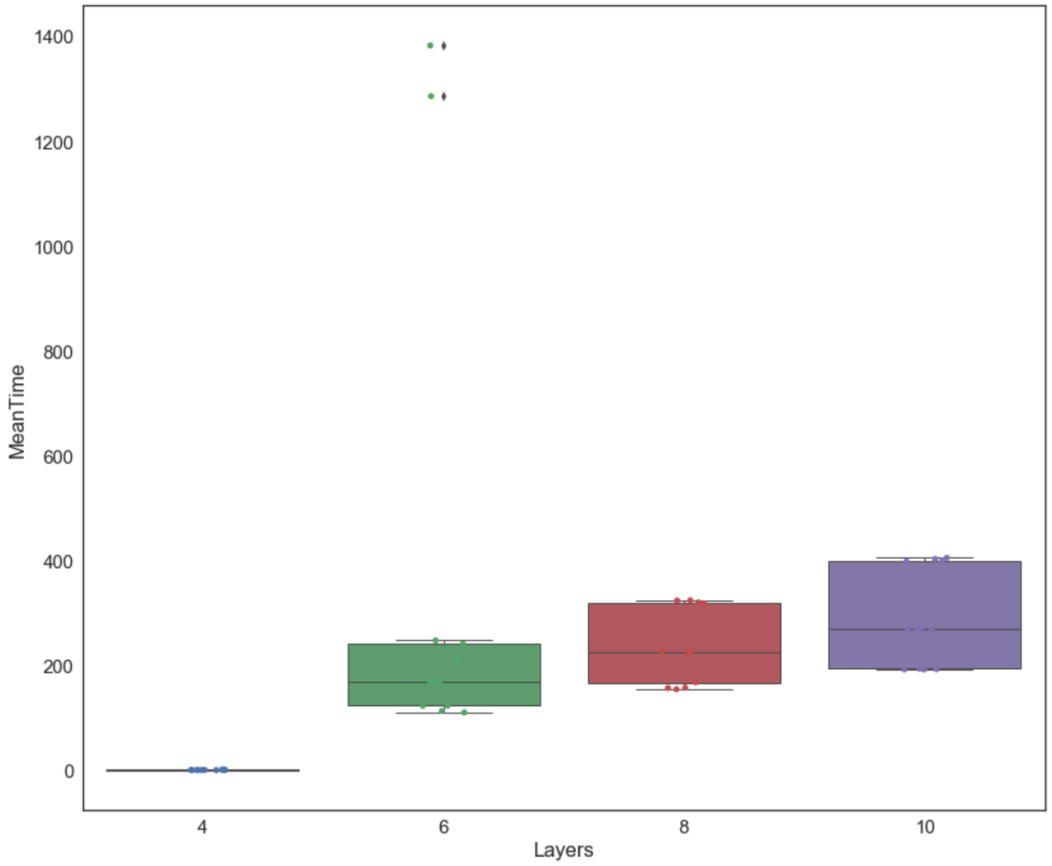
```
In [10]: matplotlib.rcParams['figure.figsize'] = (14.0, 12.0)
sns.set_context("notebook", font_scale=1.5, rc={"lines.linewidth": 1})

ax = sns.boxplot(x="Layers", y="MeanTime", data=data)
ax = sns.stripplot(x="Layers", y="MeanTime", data=data, jitter=True, edgecolor="gray")
```

Text Classifier

Again we plotted some boxplots to show how are distributed the values of the Mean inference Time for each number of layers involved.

Here the differences are not as sharp as in the other model.

Again we can see that, with more layers, also the variance of the values of Mean Time increases.



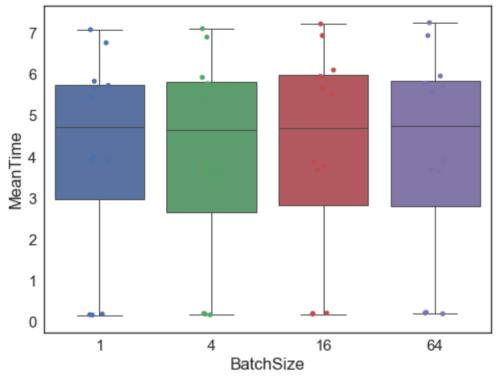
Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

Analysis of the results - Boxplots

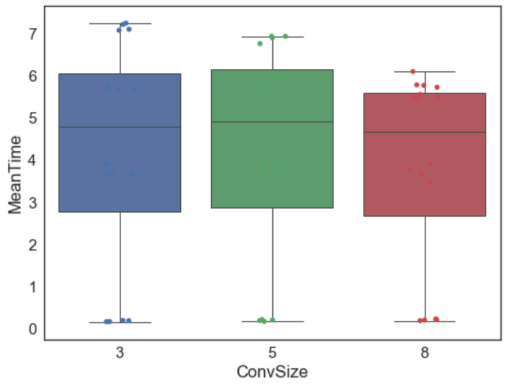
Fashion MNIST

When we try to plot the same boxplots to see the relation between Mean Time and both Batch Size and Convolution Size we get the results in the figures below.
We have not significant differences in the inference time neither for Batch Size nor for Convolution Size.
Moreover we thought that increasing the Convolution Size (which is the size of the filters) the Mean Time could have been less but this doesn't happen with ConvSize = 5, and happens only a little bit with ConvSize = 8.

```
In [6]: matplotlib.rcParams['figure.figsize'] = (8.0, 6.0)
sns.set_context("notebook", font_scale=1.5, rc={"lines.linewidth": 1})

ax = sns.boxplot(x="BatchSize", y="MeanTime", data=data)
ax = sns.stripplot(x="BatchSize", y="MeanTime", data=data, jitter=True, edgecolor="gray")
```




```
In [7]: matplotlib.rcParams['figure.figsize'] = (8.0, 6.0)
sns.set_context("notebook", font_scale=1.5, rc={"lines.linewidth": 1})

ax = sns.boxplot(x="ConvSize", y="MeanTime", data=data)
ax = sns.stripplot(x="ConvSize", y="MeanTime", data=data, jitter=True, edgecolor="gray")
```



Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

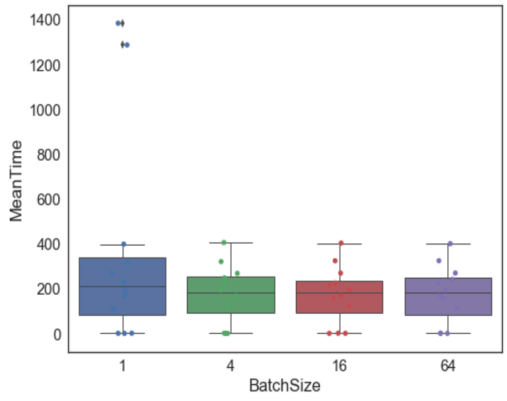
Analysis of the results - Boxplots

Text Classifier

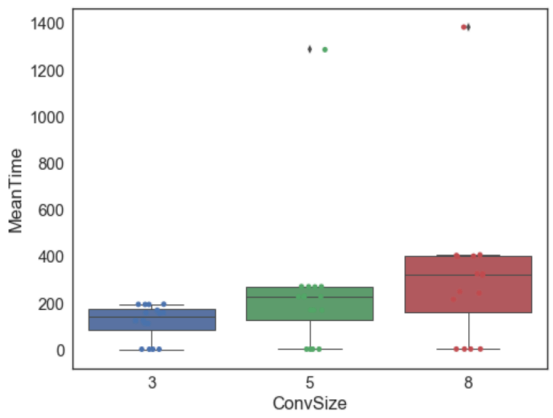
As we could expect from the correlation matrix shown before we can see that the convolution size still have an impact on the Mean inference Time, keeping the same characteristic of increase the variance of the results proportionally to the size of the convolution.

The Batch Size confirms to be more or less unrelated to the inference time.

```
In [11]: matplotlib.rcParams['figure.figsize'] = (8.0, 6.0)
         sns.set_context("notebook", font_scale=1.5, rc={"lines.linewidth": 1})

         ax = sns.boxplot(x="BatchSize", y="MeanTime", data=data)
         ax = sns.stripplot(x="BatchSize", y="MeanTime", data=data, jitter=True, edgecolor="gray")
```




```
In [12]: matplotlib.rcParams['figure.figsize'] = (8.0, 6.0)
         sns.set_context("notebook", font_scale=1.5, rc={"lines.linewidth": 1})

         ax = sns.boxplot(x="ConvSize", y="MeanTime", data=data)
         ax = sns.stripplot(x="ConvSize", y="MeanTime", data=data, jitter=True, edgecolor="gray")
```



Roberto Bigazzi, Filippo Collini,
Assoc. prof: Cristina Silvano, DEIB Dept.

References

Starting point:

- https://www.tensorflow.org/
- https://www.tensorflow.org/tutorials/layers#intro_to_convolutional_neural_networks
- https://www.tensorflow.org/tutorials/deep_cnn#highlights_of_the_tutorial

Code we borrowed and tutorials we used:

- https://github.com/bhaveshoswal/CNN-text-classification-keras
- https://www.pyimagesearch.com/2019/02/11/fashion-mnist-with-keras-and-deep-learning/

Our GitHub repository:

- https://github.com/filippocollini/tensorflow_benchmark