Banana Plants Treatment Classification

Itamar Gozlan 23/04/2020

- Introduction

Introduction

- Banana plants are very important and are a big part of the nutrition of many parts of the world. more than 100 billion bananas are eaten every year in the world, making them the most popular agricultural product
- Banana plants may have all kinds of diseases and there is all sort of kinds of treatments
- In This project, we will try to distinguish between 4 kinds of Banana treatments by observing the pictures only
- There are 120 plants treated with 4 different levels of water and fertilizer stress
- Photographed daily, 17 days consecutively, 11-28 September 2018 (except 19/09)
- The images resolution is (4032 x 3024)

Introduction

- the plants are getting 4 different treatments (A, B, C, D)
- Where the quality of the treatment is from A, that includes the highest quality treatment to D which get the lowest quality treatment

20180924 A 06

20180918 B 05

20180922 C 21

20180928 D 04

The data

- There are about 2000 pictures from 4 categories (500 each)
- Each photo contains it's plant ID and the date that it was pictured
- To avoid bias that might arise from identifying the same plant there is a complete separation between Train\Test and Validation
- ID's 05, 15, 25 are used for validation only
- Rest of the dataset is randomly distributed between test (0.2) and train (0.8)

Goals and motivation

- Are the differences between the pictured banana plants can be noticed by a Convolutional Neural Network?
- Compare several convolutional neural networks architectures and data inputs, analyze results and draw insights
- Obtain a prediction that gives the possibility to distinguish between different banana plants that got different treatment

- Experiment 1 - With Full Background

Experiment methods

- Using Native (CIFAR-10 based) neural network and transfer Learning (MobileNet, GoogleNet)
- Perform the same experiment with and without data augmentations
- CIFAR-10 network
  - 400 epochs
  - Batch size 32
- Transfer Learning networks
  - 300 epochs
  - Batch size 32
- Augmentations:
  - Horizontal Flip, Vertical Flip, Width shift, Height shift, Shear, Rotation

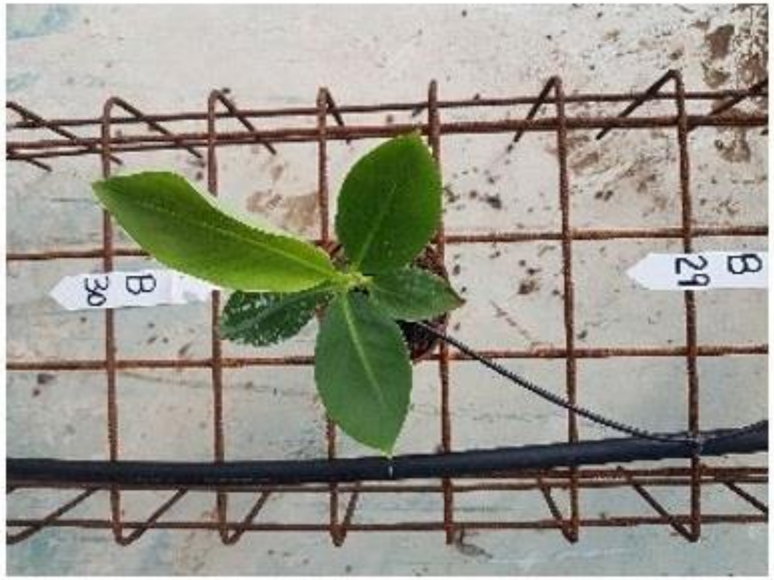
Results & Observations

GoogleNet		MobileNet		CIFAR-10 Based	
Non-Aug	**With-Aug**	**Non-Aug**	**With-Aug**	**Non-Aug**	**With-Aug**
53%	48%	51%	61%	69%	72%

- The architecture is important
- There is a notable difference in the pictures between the different treatments, although might not seem very obvious to a non-expert viewer
- Too much information is confusing!
    - GoogleNet performs better without augmentation
    - Transfer learning yields lower accuracy in total
    - But more data from the same kind yields better results!

Predictions (CIFAR-10 Based)

- True Predictions



- False Predictions

Experiment 2 - Without Background (Segmented)

Moving forward – old data in new representation

- Working with the same data BUT without the background
- How much information does the background add to the classification?
- Labels on the pictures! Need to isolate the main features the network learns from

Moving forward – old data in new representation

20180917 A 04:

20180917 B 07

Moving forward – old data in new representation

20180926 C 21

20180921 D 04

Experiment Method

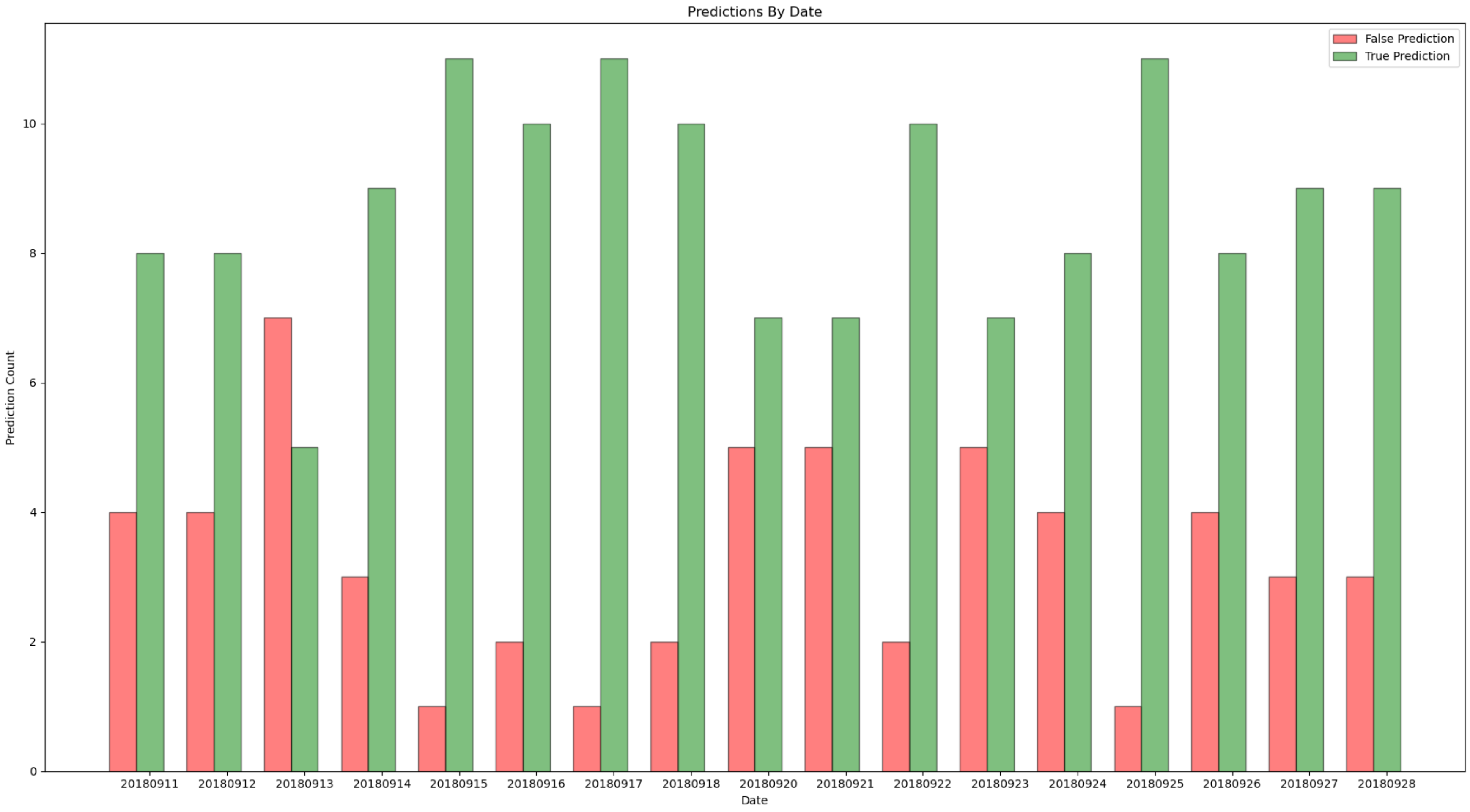
- Using cropped plants as is
- Train the same Network from the previous experiment (CIFAR-10 Based)
- Train the top Transfer Learning networks from before (MobileNet, GoogleNet)
- Use Keras built-in data augmentation and also try running without any augmentations
- CIFAR-10 network
  - 400 epochs
  - Batch size 32
- Transfer Learning networks
  - 300 epochs
  - Batch size 32
- Augmentations:
  - Horizontal Flip, Vertical Flip, Width shift, Height shift, Shear, Rotation

Result & Observations

Experiment	GoogleNet		MobileNet		CIFAR-10 Based	
	Non-Aug	With-Aug	Non-Aug	With-Aug	Non-Aug	With-Aug
Original pictures	53%	48%	51%	61%	69%	72%
Segmented Pictures	25%	-	36%	-	72%	81.8%

- There is a notable difference between the categories – no labels on the picture
- Too much information is confusing!
  - Transfer Learning almost fails completely
  - BUT additional relevant data, using augmentations on a network that wasn't trained before improves accuracy
- The background only interrupts! Without the background, on a native network the accuracy is higher
- With the background the results of the Transferred learning networks is better – that in a way is sanity check (because we expect it to handle a lot of details better)

CIFAR-10 based network – By Day



TRUE PREDICTION BY DATE
20180911 : 8
20180912 : 8
20180913 : 5
20180914 : 9
20180915 : 1
120180916 : 1
020180917 : 1
120180918 : 1
020180920 : 7
20180921 : 7
20180922 : 1
020180923 : 7
20180924 : 8
20180925 : 1
120180926 : 8
20180927 : 9
20180928 : 9

FALSE PREDICTION BY DATE
20180911 : 4
20180912 : 4
20180913 : 7
20180914 : 3
20180915 : 1
20180916 : 2
20180917 : 1
20180918 : 2
20180920 : 5
20180921 : 5
20180922 : 2
20180923 : 5
20180924 : 4
20180925 : 1
20180926 : 4
20180927 : 3
20180928 : 3

accuracy = 0.7254

CIFAR-10 based network – False Predictions

False Predictions

y_true = C y_res = D
False Prediction
Date 20180920

y_true = B y_res = D
False Prediction
Date 20180913

True Predictions

y_true = C y_res = C
True Prediction
Date 20180912

y_true = B y_res = B
True Prediction
Date 20180916

- Experiment 3 - Following A Hint

Experiment 3 - Introduction

- Following an experts “hint” about the connection between the treatment to the leaves growth rate
- Introducing a novel concept of Augmentation that aids exploiting data that has some sort of sequential connection

20180911 C 08

20180912 C 08

20180913 C 08

Experiment 3 - Introduction

- With the triplets, augmentation each image separately is possible, thus getting a substantial augmentation boost compared to a single image.
- Using this new method, we can increase augmentation exponentially by augmenting each picture in the sequence separately
- Forming triplets from sequential days to create “new” data set as a form of augmentation

Experiment Method

- Using plants as triplets
- Train the same Network from the previous experiment (CIFAR-10 Based)
- Train the top Transfer Learning networks from before (MobileNet, GoogleNet)
- Use Keras built-in data augmentation and also try running without any augmentations – **Without horizontal flip - to maintain the order**
- with the plants as triplets
  - Every 3 consecutive pictures were transformed to a triplet
  - For example:
    - 53722_20180911_153147_RGB_Treat_A_04.jpg
    - 53717_20180912_160043_RGB_Treat_A_04.jpg
    - 53721_20180913_151116_RGB_Treat_A_04.jpg
    - Result: 53722_53717_53721_20180911_20180912_20180913_RGB_Treat_A_04.jpg
- Distribution to Test\Train\Validation in the same way as Experiment I
  - Plants 05,15,25 were separated as validation group (as triplets)
  - Other plants were randomly distributed to train and test with the ratio of 0.2 test, 0.8 train

Result & Observations

Experiment	GoogleNet		MobileNet		CIFAR-10 Based	
	Non-Aug	With-Aug	Non-Aug	With-Aug	Non-Aug	With-Aug
Original pictures	53%	48%	51%	61%	69%	72%
Segmented Pictures	25%	-	36%	-	72%	81.8%
Triplets	34%	25%	25%	32%	74%	84%

- The CIFAR-10 network architectures allow a certain flexibility in the input data form
  - Excelled among the other experiments with augmentation and without in all forms of data (with background, without background and with triplets)
- Improvement can be acquired by exploiting the sequential connection

Experiment 4

- A vs ALL

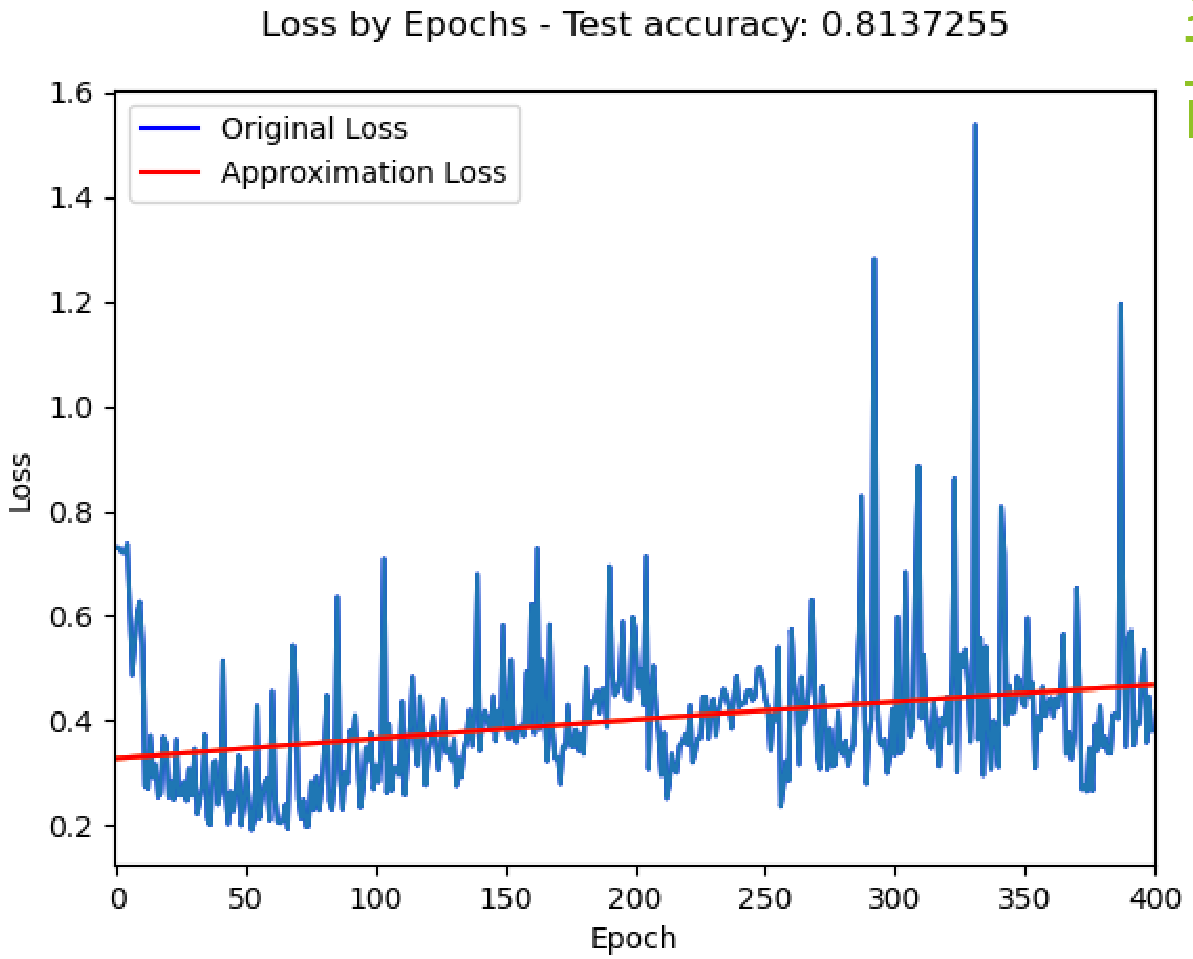
Further questions

- Can we use another hint to improve accuracy even more?
- Heading a new direction with 2-treatment categorization (A vs the rest)
- How much flexible the CIFAR-10 network can be?
- 2-category prediction A vs ALL (B,C,D)
- What about A vs A/B/C? what will be the hypothesis?
  - Following the previous knowledge we have on the dataset we might expect higher accuracy when the treatment is the farthest from A (in quality)
- What can we say about the prediction by day?

A vs. ALL – Experiment Method

- Using cropped plants as is
- Data/Train/Validation contains only A and 0.33 of each shuffled category (0.33 from B,C and D regardless to the date)
- Plants with ID 05,15,25 are strictly reserved for validation
- Train the same Network from the previous experiment (CIFAR-10 Based)
- CIFAR-10 network
  - 400 epochs
  - Batch size 32

A vs ALL (B,C,D)



y_true = C
y_res = NOT_A
True Prediction
Date 20180923

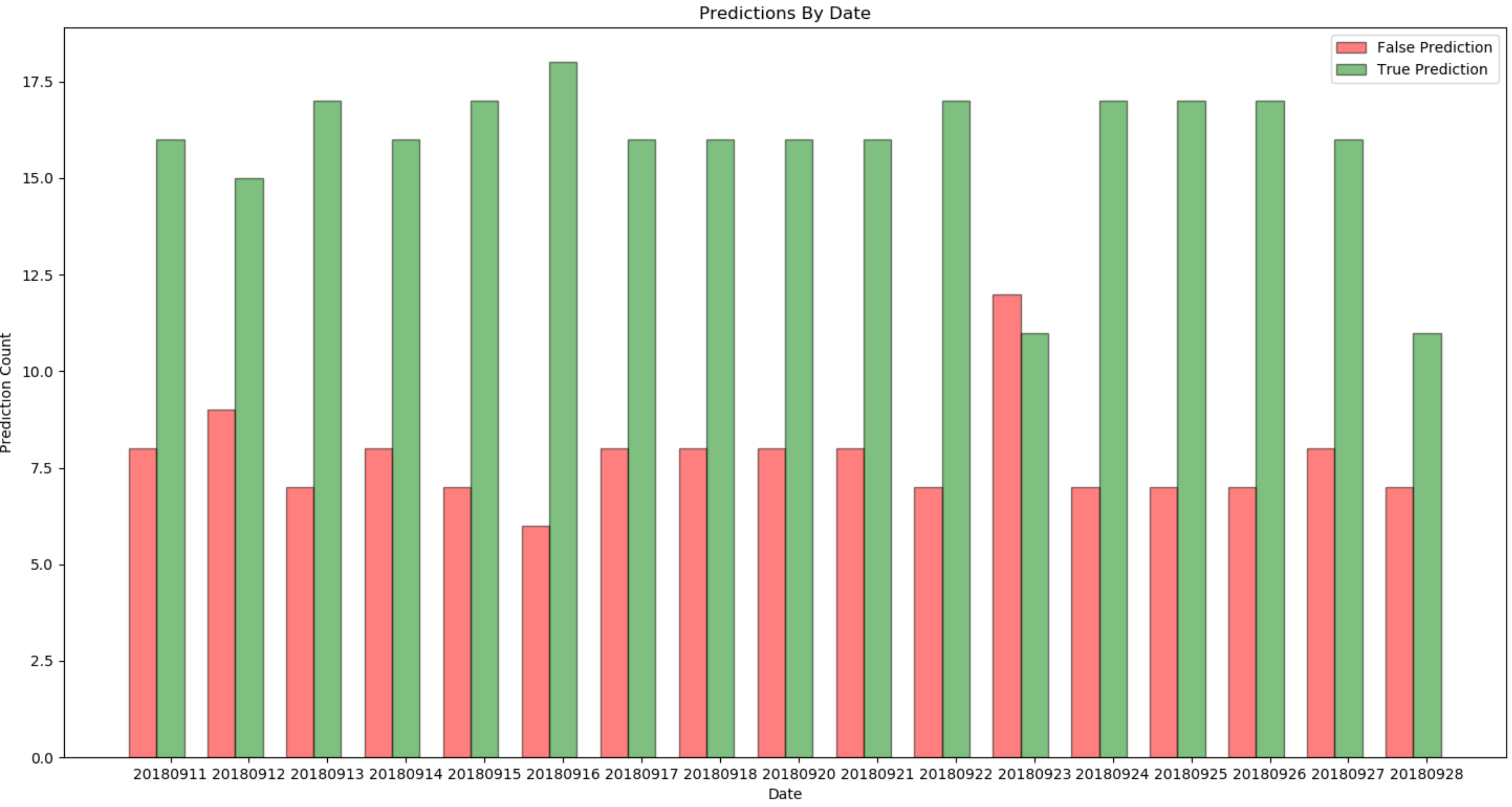
y_true = NOT_A
y_res = A
False Prediction
Date 20180921

A vs. Each – Experiment Methos

- Using cropped plants as is
- Data/Train/Validation contains only A and B/C/D each at the time
- Plants with ID 05,15,25 are strictly reserved for validation
- Train the same Network from the previous experiment (CIFAR-10 Based)
- CIFAR-10 network
  - 400 epochs
  - Batch size 32

Results Summary

A vs. ALL	A vs. B	A vs. C	A vs. D
81.3%	77.45%	93.13%	99%



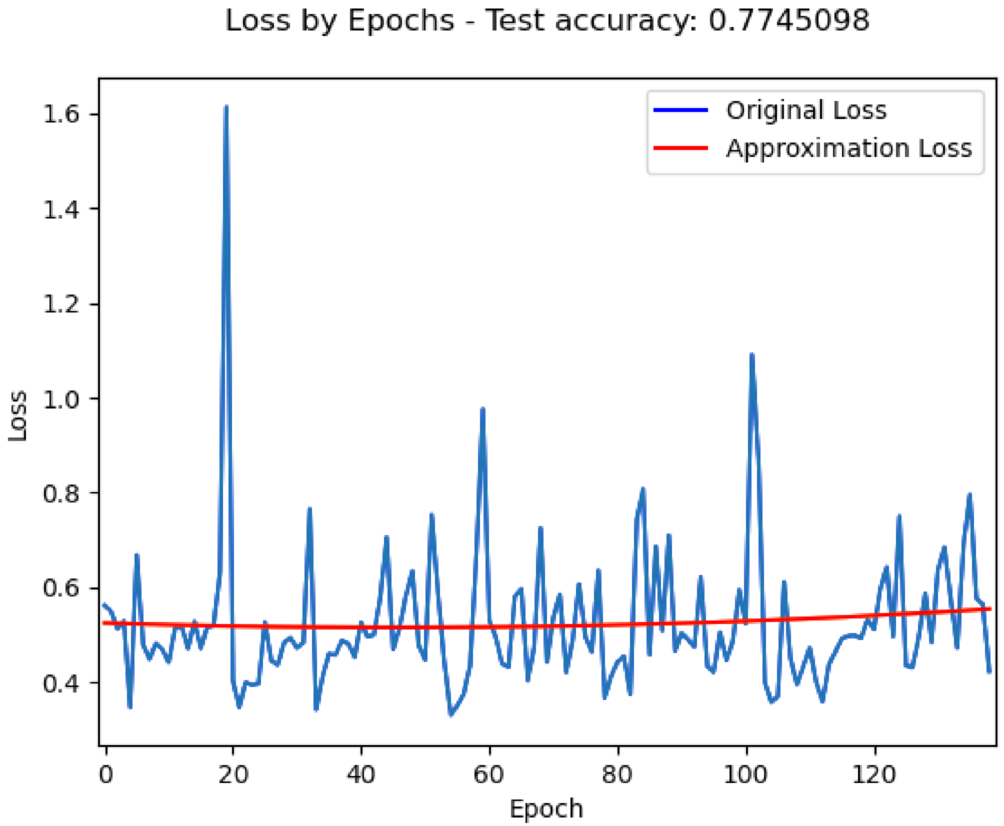
TRUE PREDICTION BY DATE

20180911 : 16
20180912 : 15
20180913 : 17
20180914 : 16
20180915 : 17
20180916 : 18
20180917 : 16
20180918 : 16
20180920 : 16
20180921 : 16
20180922 : 17
20180923 : 11
20180924 : 17
20180925 : 17
20180926 : 17
20180927 : 16
20180928 : 11

FALSE PREDICTION BY DATE

20180911 : 8
20180912 : 9
20180913 : 7
20180914 : 8
20180915 : 7
20180916 : 6
20180917 : 8
20180918 : 8
20180920 : 8
20180921 : 8
20180922 : 7
20180923 : 12
20180924 : 7
20180925 : 7
20180926 : 7
20180927 : 8
20180928 : 7

A vs B



y_true = A y_res = A

True Prediction

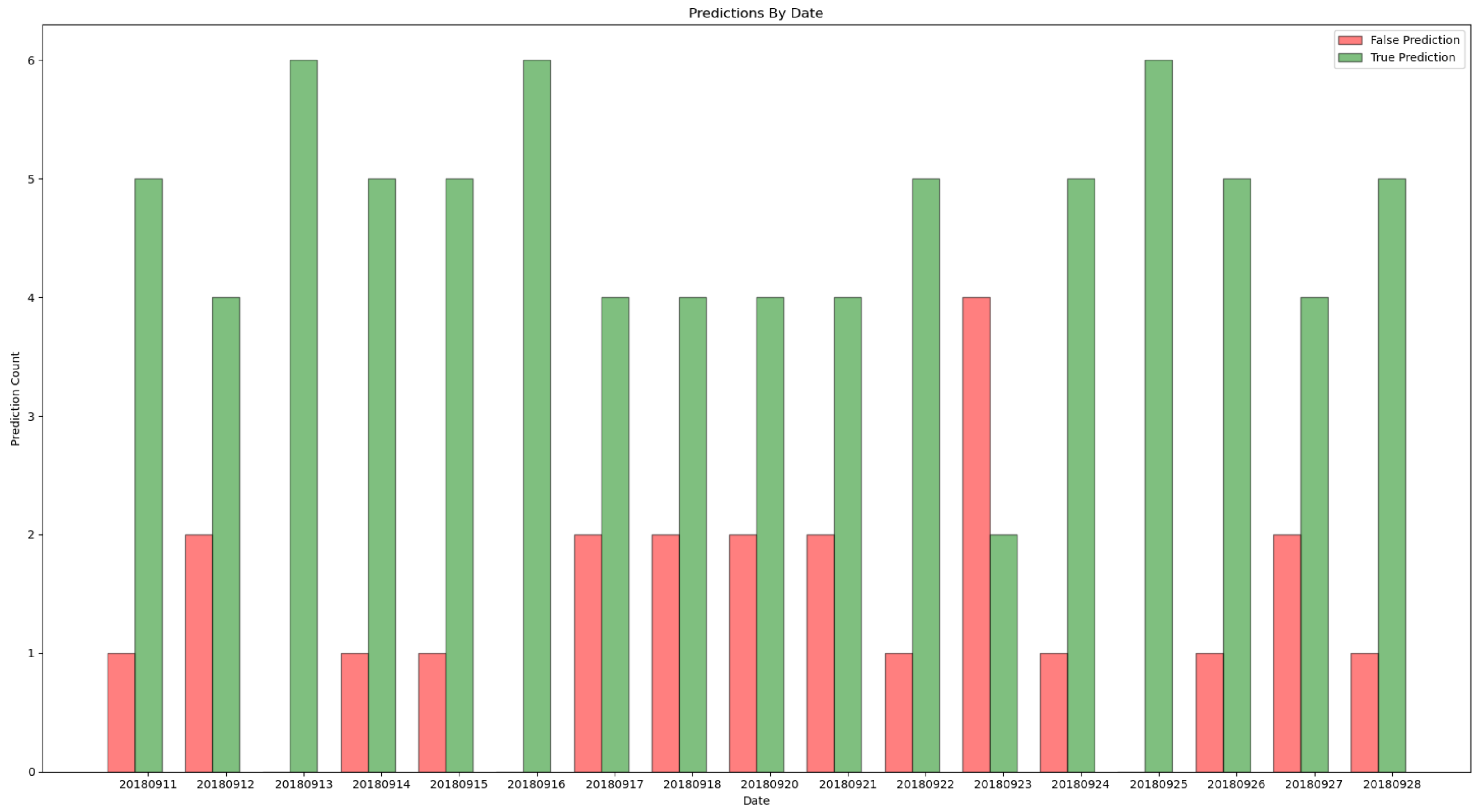
Date 20180912

y_true = A y_res = B

False Prediction

Date 20180917

A vs B



TRUE PREDICTION BY DATE

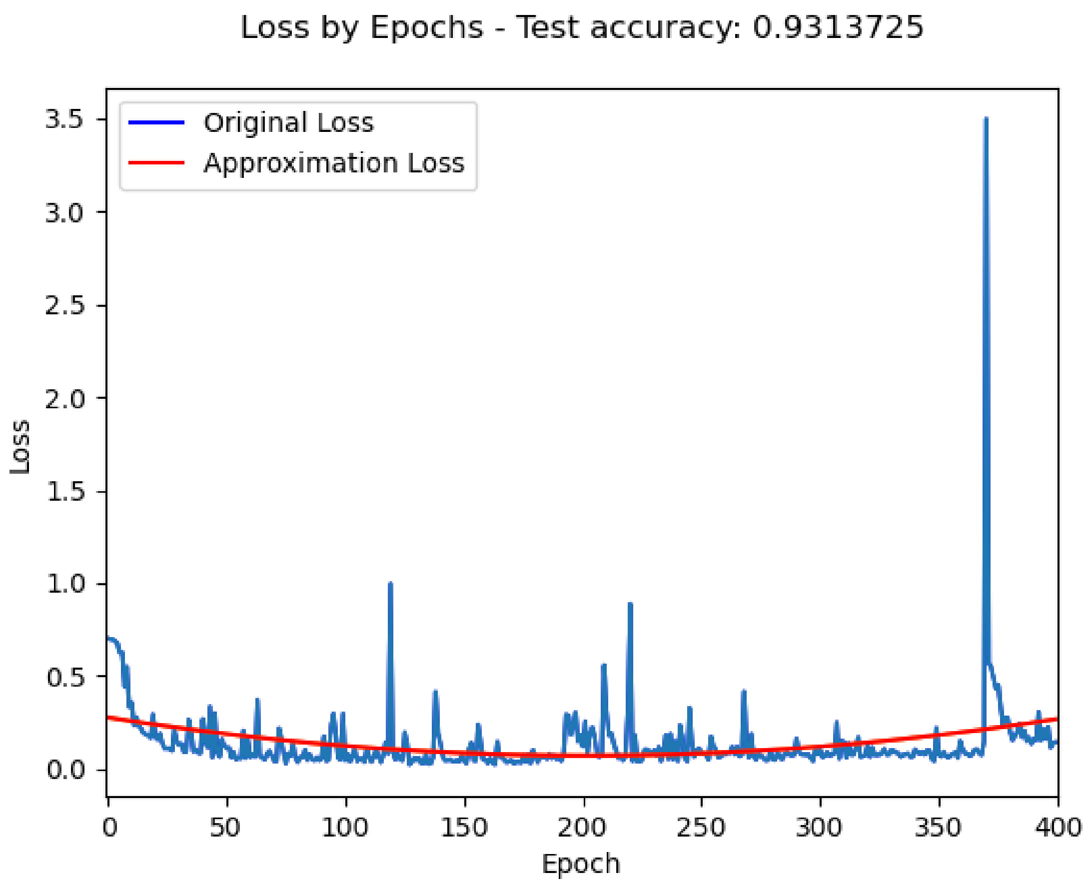
20180911 : 5
20180912 : 4
20180913 : 6
20180914 : 5
20180915 : 5
20180916 : 6
20180917 : 4
20180918 : 4
20180920 : 4
20180921 : 4
20180922 : 5
20180923 : 2
20180924 : 5
20180925 : 6
20180926 : 5
20180927 : 4
20180928 : 5

FALSE PREDICTION BY DATE

20180911 : 1
20180912 : 2
20180914 : 1
20180915 : 1
20180917 : 2
20180918 : 2
20180920 : 2
20180921 : 2
20180922 : 1
20180923 : 4
20180924 : 1
20180926 : 1
20180927 : 2
20180928 : 1

accuracy = 0.7745098039215687

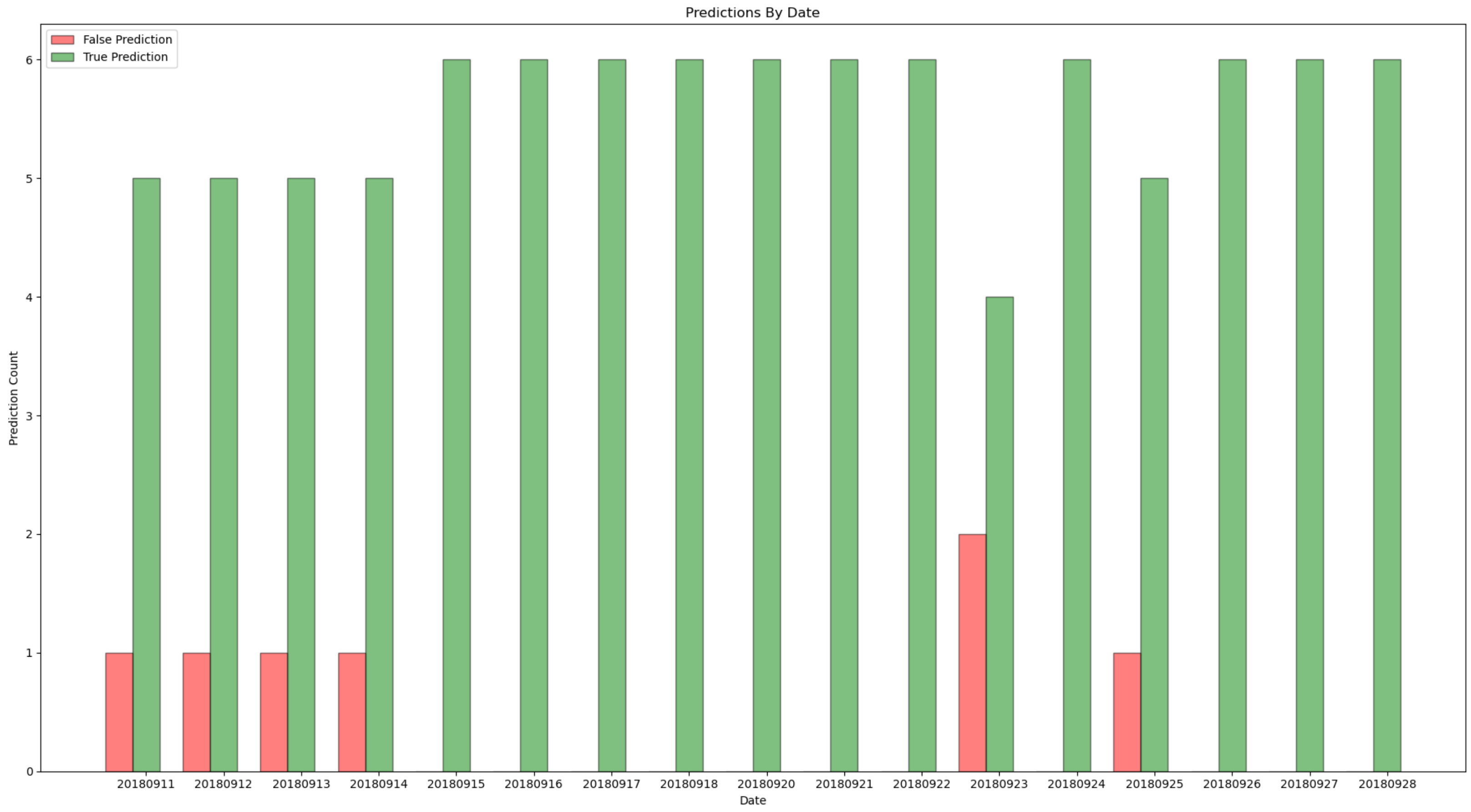
A vs C



y_true = A y_res = A
<u>True Prediction</u>
Date 20180928

y_true = C y_res = A
<u>False Prediction</u>
Date 20180923

A vs C



TRUE PREDICTION BY DATE

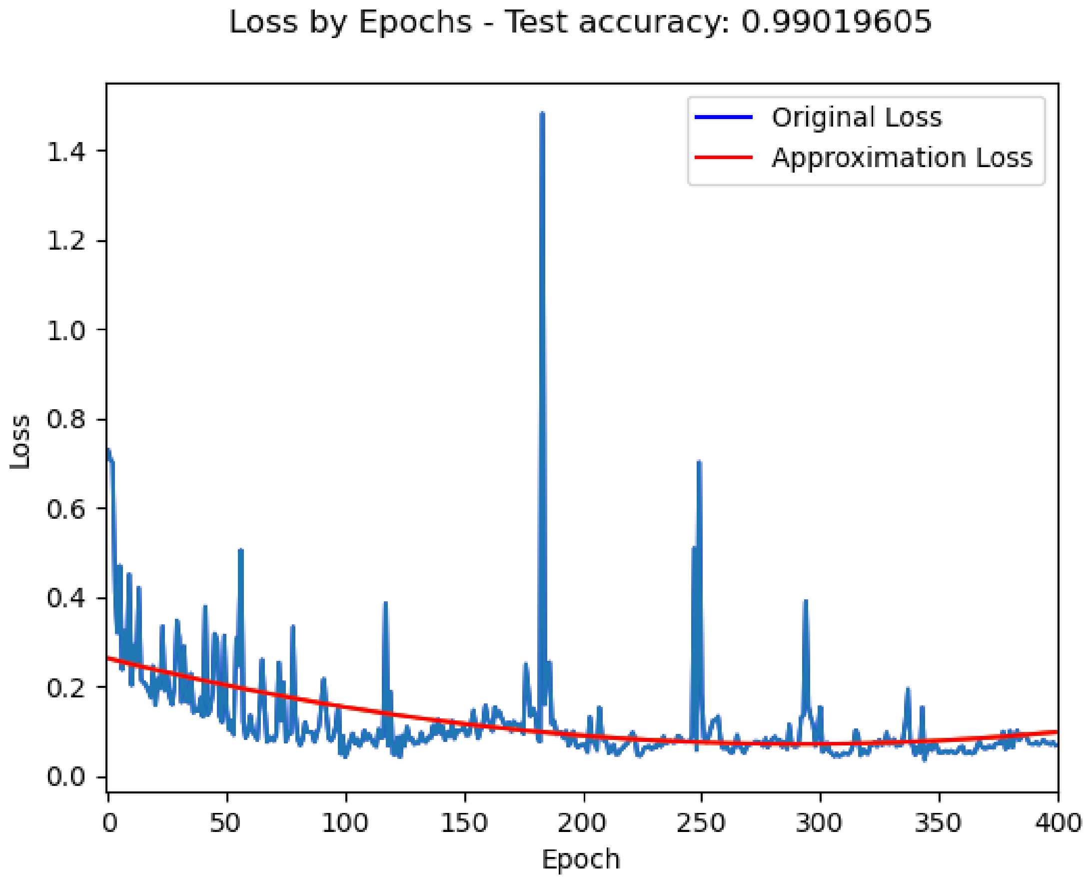
20180911 : 5
20180912 : 5
20180913 : 5
20180914 : 5
20180915 : 6
20180916 : 6
20180917 : 6
20180918 : 6
20180920 : 6
20180921 : 6
20180922 : 6
20180923 : 4
20180924 : 6
20180925 : 5
20180926 : 6
20180927 : 6
20180928 : 6

FALSE PREDICTION BY DATE

20180911 : 1
20180912 : 1
20180913 : 1
20180914 : 1
20180923 : 2
20180925 : 1

accuracy = 0.93137

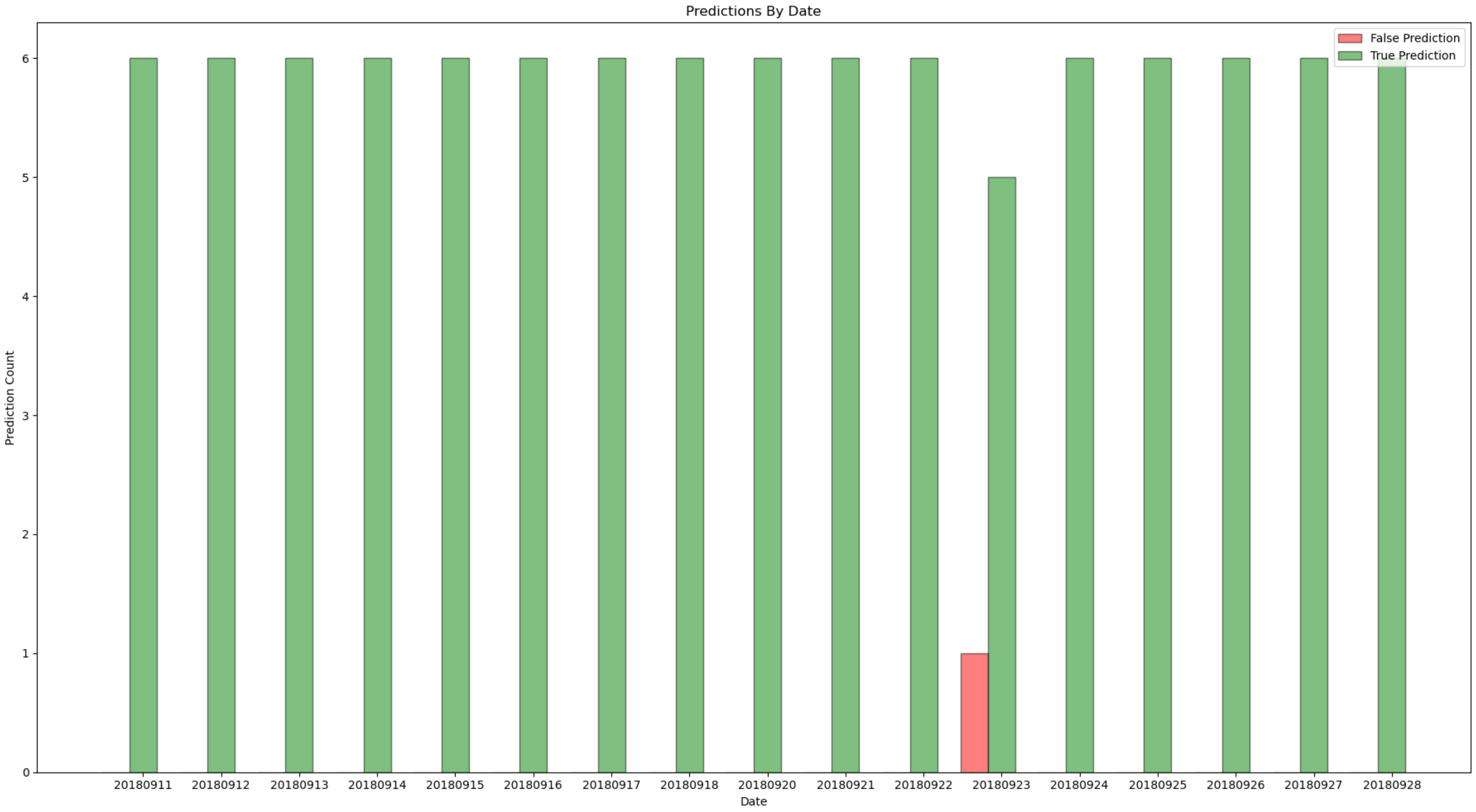
A vs D



y_true = A y_res = A
True Prediction
Date 20180926

y_true = A y_res = D
False Prediction
Date 20180923

A vs D



TRUE PREDICTION BY DATE

20180911 : 6
20180912 : 6
20180913 : 6
20180914 : 6
20180915 : 6
20180916 : 6
20180917 : 6
20180918 : 6
20180920 : 6
20180921 : 6
20180922 : 6
20180923 : 5
20180924 : 6
20180925 : 6
20180926 : 6
20180927 : 6
20180928 : 6

FALSE PREDICTION BY DATE

20180923 : 1

accuracy = 0.99

Main conclusions

- Notable differences between A,B,C,D
- The treatment scale is well expressed in the pictures and the network succeeds in finding it
- The CIFAR-10 network architectures allow a certain flexibility in the input data form and works very well on this dataset
- Higher accuracy rates around 14/09/2018 and 24/04/2018 but not in a notable way

Supplementary Material

Future work

- How much can we push the CIFAR-10 network?
  - Different datasets?
  - Close dataset? (Thermal)
- Can we exploit the triplets idea even more? Augmentation for each picture separately, exploit similar links in other datasets with sequential connection
- What about other “pre-trained” networks? Other architecture will work well on the same dataset?
- Is the high accuracy more a data-set quality or architecture dependent?

Common Terms

- epochs: Integer. Number of epochs to train the model. An epoch is an iteration over the entire (X,Y) data provided - iterations on a dataset (Train and Test)
- batch_size: Integer or None. Number of samples per gradient update. In this project I used 32, so that means that each time the weights get updated it will consider 32 pictures
- steps_per_epoch: Total number of steps (batches of samples) before declaring one epoch finished and starting the next epoch. To cover all the dataset I used “train_size / batch_size”

When do the weights get updated?

- The weights get update when ever a batch is done
- For example, if we have 400 epochs, the dataset is 1600 and the batch size is 32:
  - $\frac{dataset\ size\ 1600}{batch\ size\ 32} = 50$ times each epoch
  - $50 \cdot 400 = 20{,}000$ times per train
- Example from Keras outputs (every line is an update):

```
Epoch 1/400

 1/50 [..............................] - ETA: 7:32 - loss: 2.0419 - accuracy: 0.2188
 2/50 [>.............................] - ETA: 3:48 - loss: 2.1069 - accuracy: 0.2188
 3/50 [>.............................] - ETA: 2:34 - loss: 1.9851 - accuracy: 0.2292
 .
 .
 .
 47/50 [==========================>...] - ETA: 5s - loss: 1.4266 - accuracy: 0.2602
 48/50 [===========================>..] - ETA: 3s - loss: 1.4253 - accuracy: 0.2600
 49/50 [===========================>..] - ETA: 2s - loss: 1.4237 - accuracy: 0.2642
 50/50 [============================>.] - ETA: 1s - loss: 1.4228 - accuracy: 0.2639
 51/50 [==============================] - 97s 2s/step - loss: 1.4213 - accuracy: 0.2667 - val_loss: 1.3691 - val_accuracy: 0.4012
```

- Augmentations

Data Augmentations

Horizontal Flip

- Vertical Flip

Data Augmentations

- Width shift

- Height shift

Data Augmentations

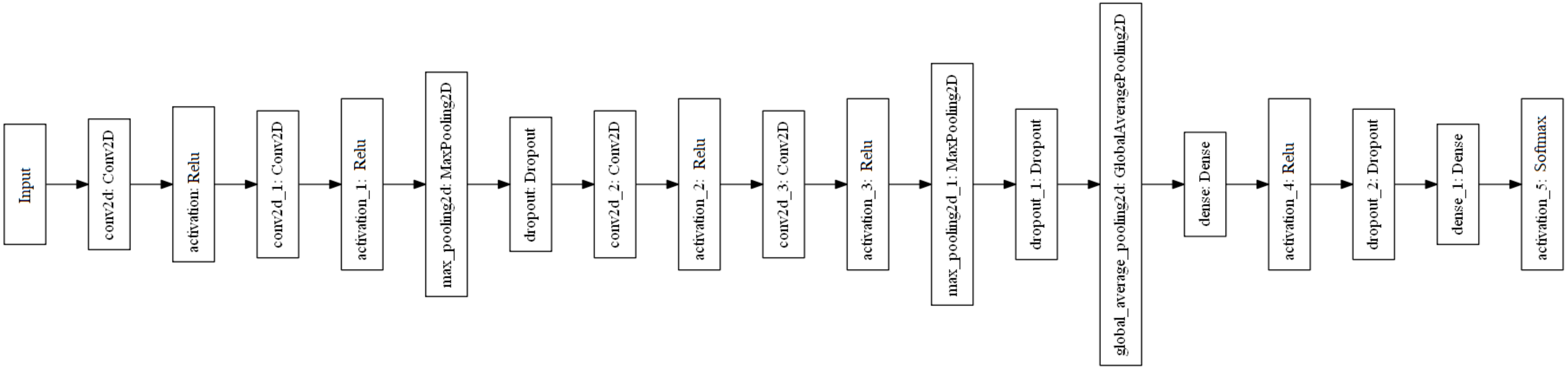
- Shear

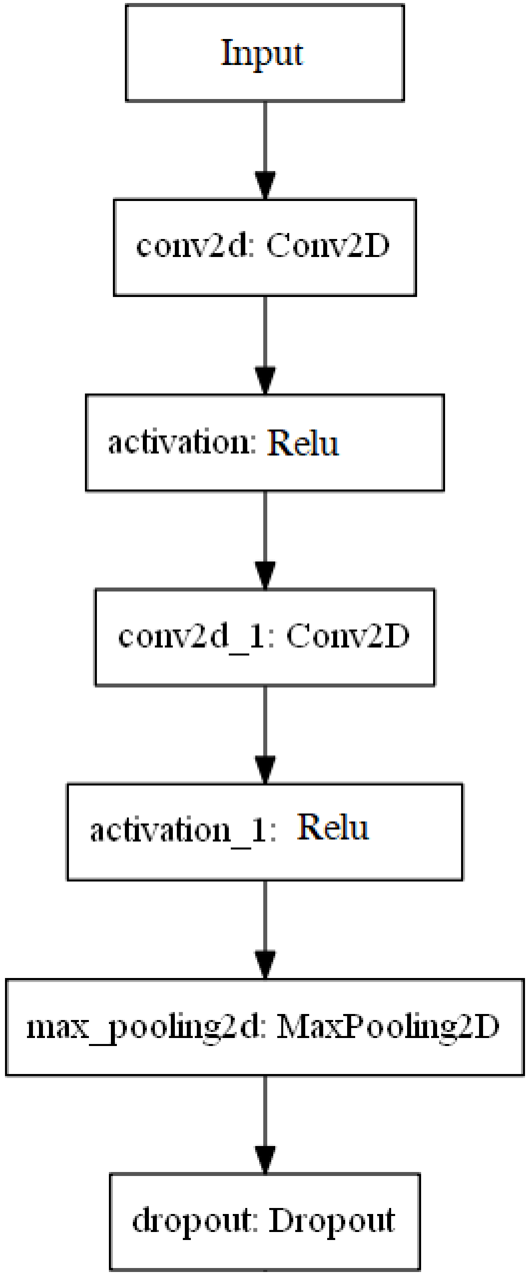
- Rotation

CIFAR 10-KERAS BASED MODEL ARCHITECTUE

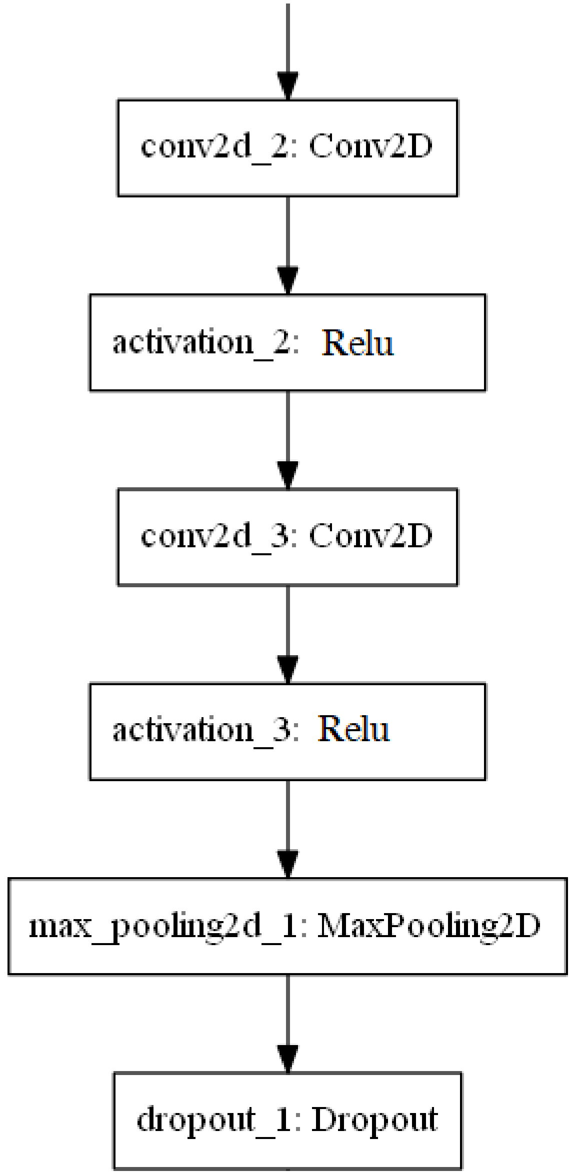
Itamar Gozlan

The complete architecture

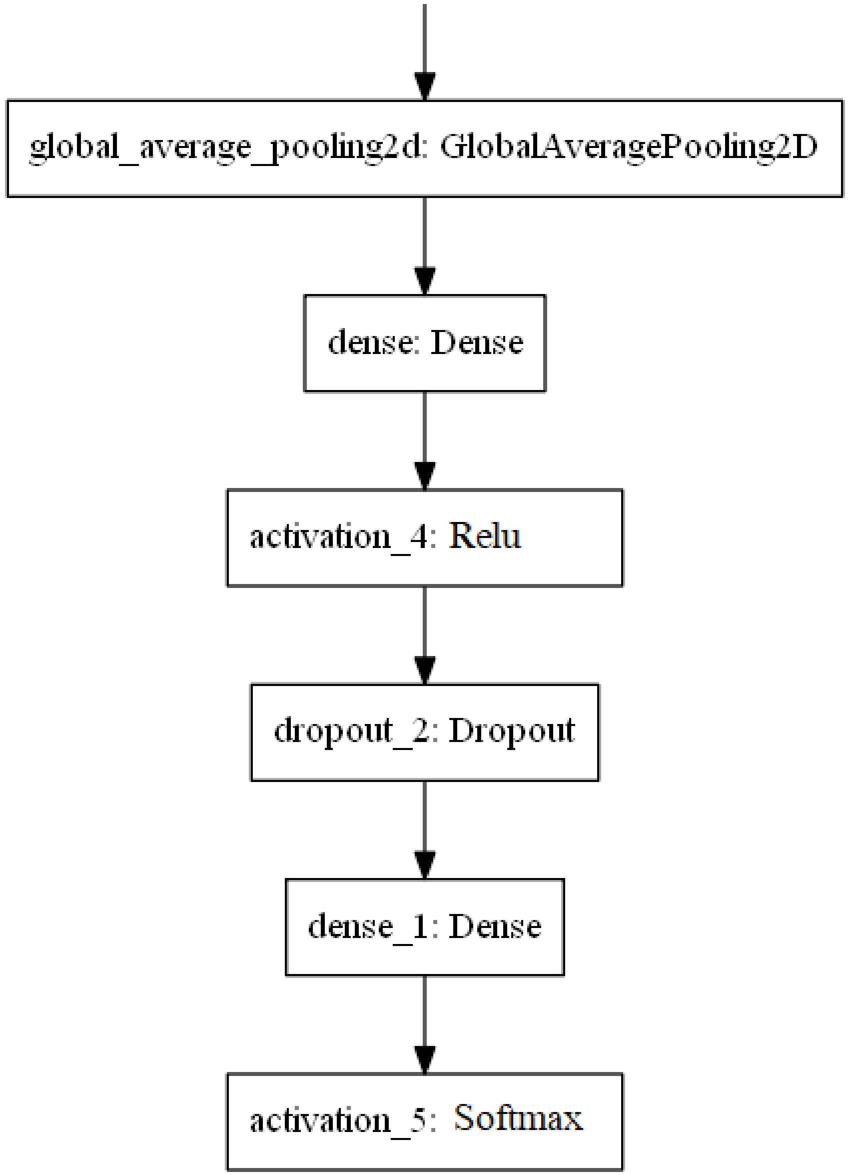




- Input dimension is depends on the experiment. initial experiment dimensions: (336, 252)
- Conv2D: 32 filters, kernel size 3x3
- Relu activation layer
- Conv2D: 32 filters, kernel size 3x3
- Relu activation layer
- MaxPooling2D – pool size (2,2)
- Dropout rate 0.25



- Conv2D: 64 filters, kernel size (3,3)
- Relu activation layer
- Conv2D: 64 filters, kernel size (3,3)
- Relu activation layer
- MaxPooling2D – pool size (2,2)
- Dropout rate 0.25



- GlobalAveragePooling2D - Global average pooling operation for spatial data.
- Dense (densely-connected NN layer)
- Relu activation layer
- Dropout rate 0.5
- Dense (densely-connected NN layer)
- Softmax