Deep Learning

CISC 867

Project 1

By:

Name : Abdelrahman Magdy El-Hamoly

**email:** 21amme1@queensu.ca

Name : Kareem Gamal Mahmoud Mohamed

**email:** 21kgmm@queensu.ca

Name : Ahmed Mohamed Gaber

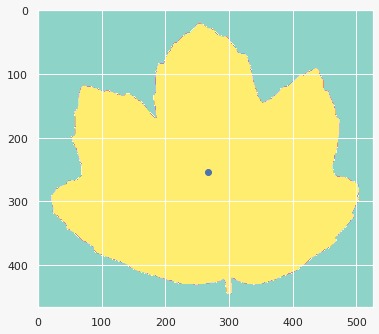
**email:** 21amga@queensu.ca

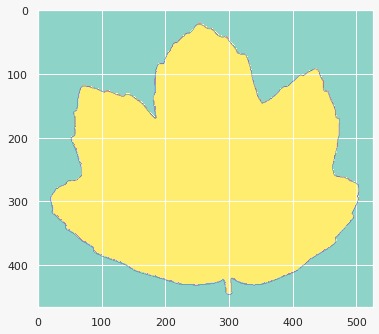
Supervised by\ Dr. Hazem Abbas

Part 1 :

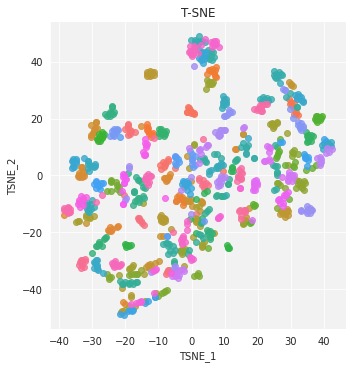
Description and cleaning:

The data is clear ,there are no missing values , there are no outliers, the data is normalized because the range of the values between zero and 1, there is no duplication.

Draw some of the images:



The distribution of the target points:



**Part 2 : Simple MLP Model**

We decided to tune the following hyperparameters:

* Batch Size
* Hidden Nodes Size
* Drop Rate
* Optimizer

In the beginning we had built a function named ‘base\_line\_model’ to build several models with different hyperparameters

This function has default hyperparameters which are

[ optim = Adam() , bat\_size = 32, hid\_nodes = 512, drop\_rate = 0.5 ]

Text

Description automatically generated

Let’s Start with Batch Size. We Used the following values:

1. Batch\_size=32

Tune with different optimizers.

Trial 1: Adam optimizer

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Chart, line chart

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 85 epochs instead of 100 epochs

Trial 2: RMS\_Prop optimizer

Text

Description automatically generated

Text

Description automatically generated

Chart, histogram

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

Trial 3: SGD optimizer

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Chart, line chart, histogram

Description automatically generated

Loss ratio for

validation & training

as we can see the SGD is very bad because the curve is not stable at all

So, from all of that the RMSprop is the best one

Now we will use the RMSprop optimizer with different batch size.

Trial 4: with Batch size = 32

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Chart, histogram

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs.

Trial 5: with Batch size = 64

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Chart, histogram

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

Trial 6 : with Batch size = 128

Graphical user interface

Description automatically generated with medium confidence

Text

Description automatically generated

Chart

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

So, from all of that the RMSprop is the best optimizer and the batch size = 64 is the best model’s hyperparameter Now we will use the RMSprop optimizer and batch size 64 with different number of hidden nodes.

Trial 7 : with 1024 hidden nodes

Text

Description automatically generated

Text

Description automatically generated

Chart, histogram

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 20 epochs instead of 100 epochs

Trial 8 : with 2048 hidden nodes

Graphical user interface

Description automatically generated

Graphical user interface, text

Description automatically generated

Chart, histogram

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 20 epochs instead of 100 epochs

Trial 9 : with 512 hidden nodes

Graphical user interface, website

Description automatically generated

Text

Description automatically generated

Chart, histogram

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

So, from all of that the RMSprop is the best optimizer and the batch size = 64 and the number of hidden nodes are 1024 nodes ,that is the best model hyperparameters Now we will use the RMSprop optimizer and batch size 64 with 1024 hidden nodes and with different drop out rate .

Trial 10 : Drop out rate = 0.3

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Chart, histogram

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 20 epochs instead of 100 epochs

Trial 11: Drop out rate = 0.6 Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Chart

Description automatically generated

Loss ratio for

validation & training

Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

Trial 12: finally this is the best model with the best hyperparameters :

Graphical user interface, application, website

Description automatically generated

* Graphical user interface, text

  Description automatically generatedBatch Size = 64
* Hidden Nodes Size = 1024
* Drop Rate = 0.5
* Optimizer = RMSprop

Chart, histogram

Description automatically generated

so the best model is model\_11 = base\_line\_model(rms\_prop , 64, 1024, 0.5) which has accuracy 0.9646 on test set and 0.1812 loss on test set also and has 100% accuracy on training data

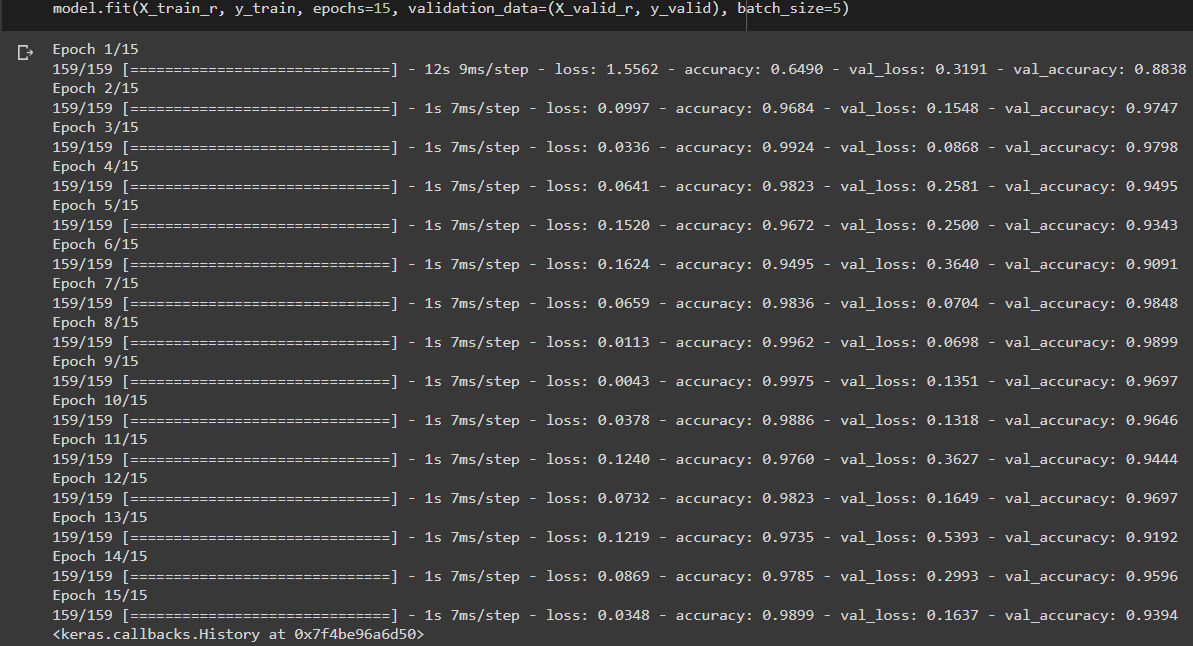
**CNN Model**

We decided to tune the following hyperparameters:

* Batch Size
* Hidden Nodes Size
* Dropout Rate
* Optimizer

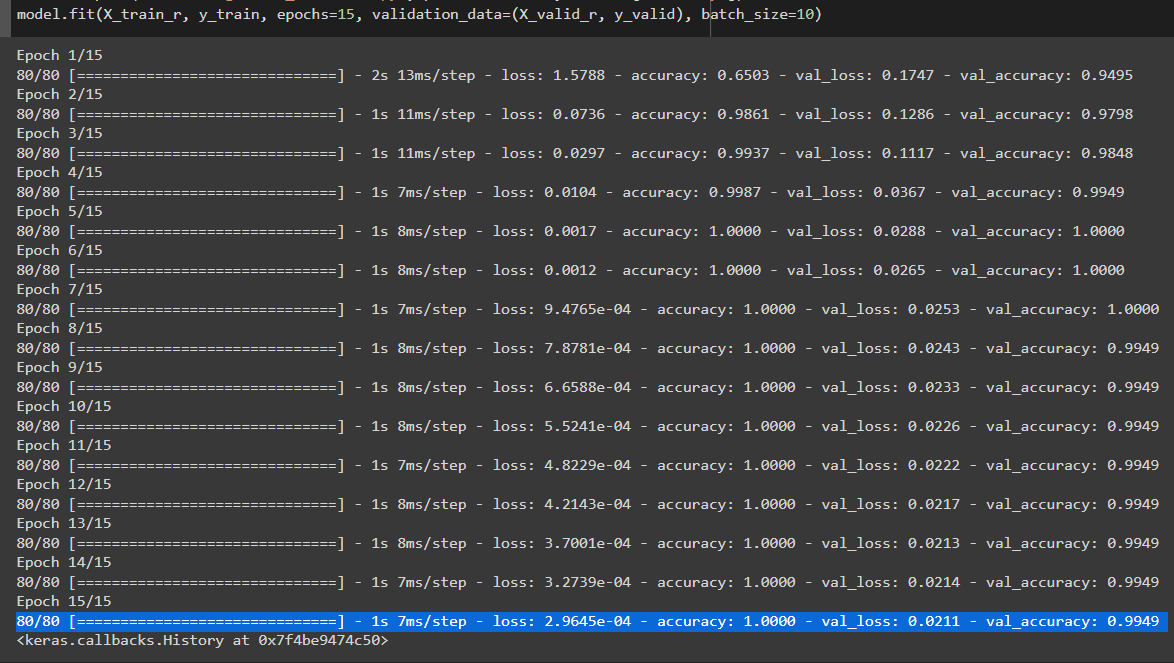
Let’s Start with Batch Size. We Used the following values:

1. Batch\_size=5



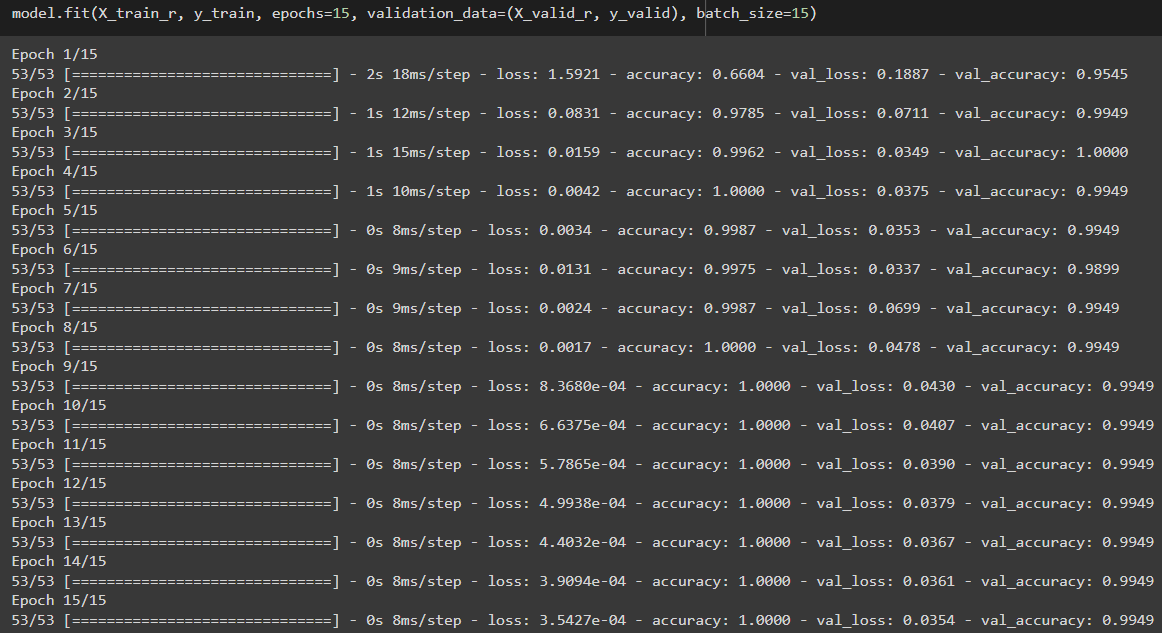
loss: 0.0348 - accuracy: 0.9899 - val\_loss: 0.1637 - val\_accuracy: 0.9394

1. Batch\_size=10



loss: 2.9645e-04 -accuracy: 1.0 -val\_loss: 0.0211 -val\_accuracy: 0.9949

1. Batch\_size=15

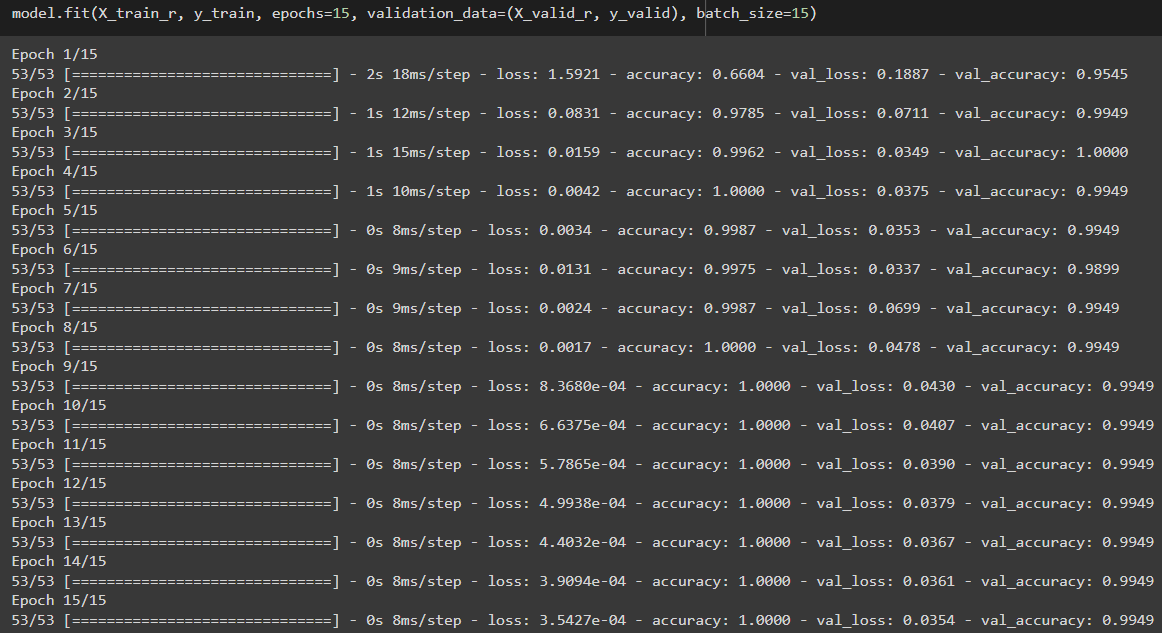


loss: 3.5427e-04 - accuracy: 1.0000 - val\_loss: 0.0354 - val\_accuracy: 0.9949

The best batch size was 10 so we stick with it.

Next we go with hidden nodes Size. We chose the next values:

1. Hidden nodes= 512



loss: 3.5427e-04 -accuracy: 1.0000 -val\_loss: 0.0354 -val\_accuracy: 0.9949

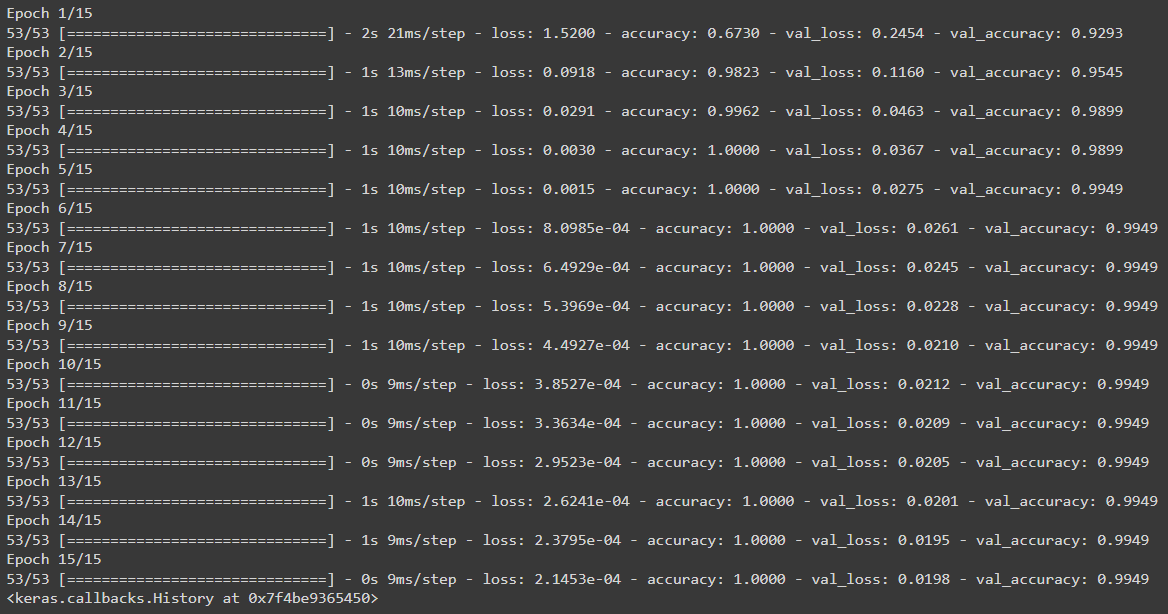
1. Hidden nodes= 256

Text

Description automatically generated

loss: 6.3737e-04 -accuracy: 1.00 -val\_loss: 0.0273 -val\_accuracy:0.9949

1. Hidden nodes= 1024



loss:2.1453e-04 -accuracy: 1.00 -val\_loss: 0.0198 -val\_accuracy: 0.9949

The best hidden nodes number was 1024 so we stick with it.

Next we go with dropout rate. We chose the next values:

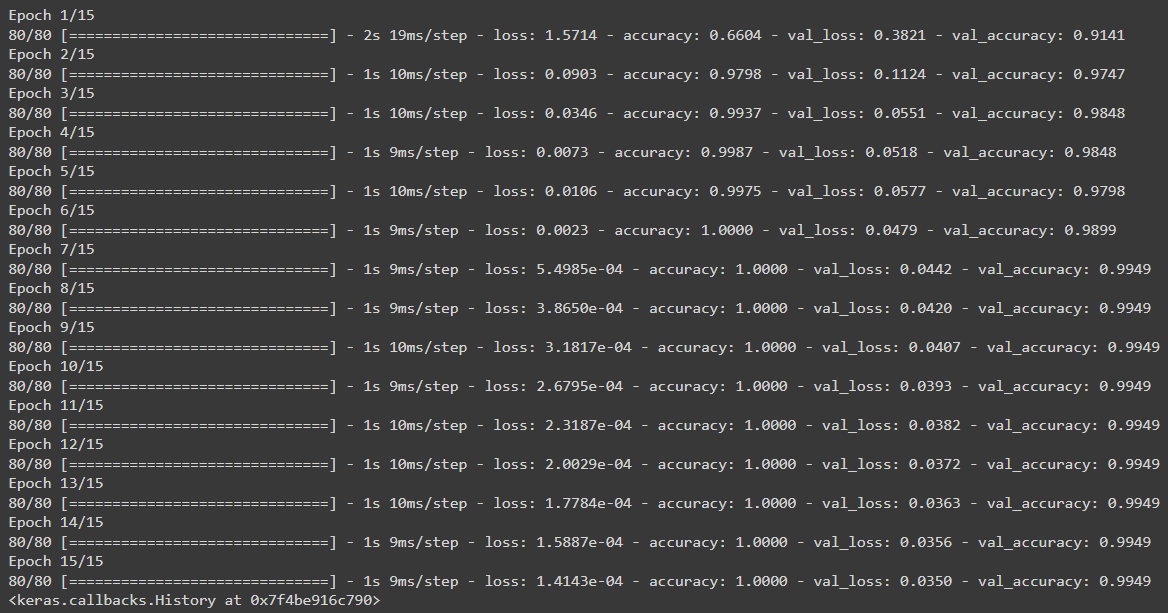
1. Dropout= 0.2

Text, calendar

Description automatically generated

loss: 1.2680e-04 -accuracy: 1.000 -val\_loss: 0.0208 -val\_accuracy: 0.9949

1. Dropout= 0.3



1. Dropout= 0.1

Text

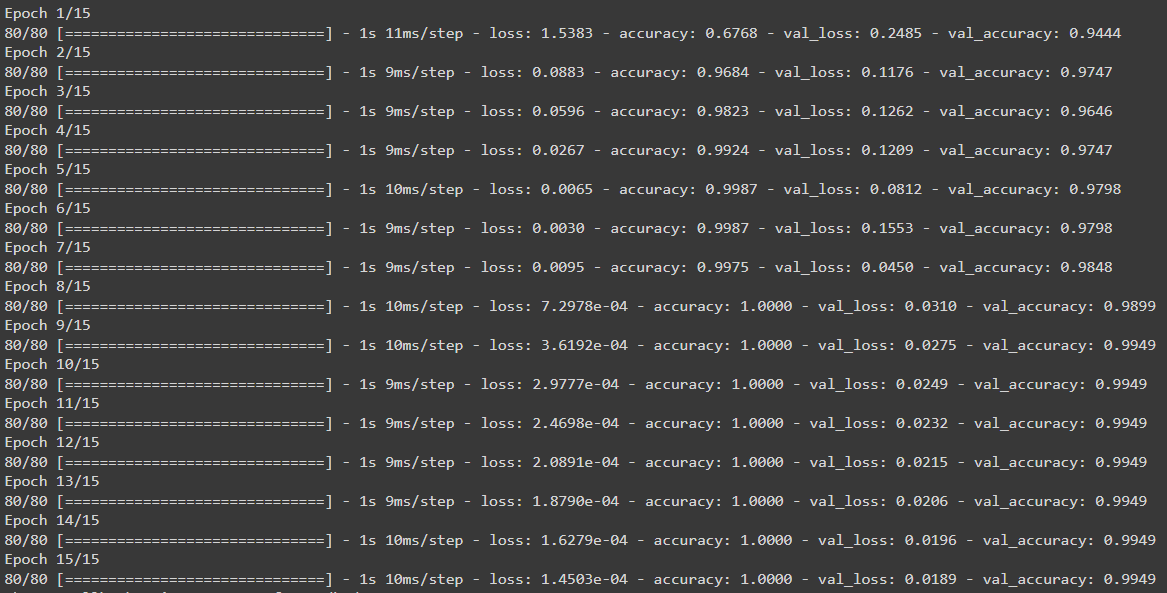
Description automatically generated

loss: 0.0575 - accuracy: 0.9924 - val\_loss: 0.1904 - val\_accuracy: 0.9646

The best Dropout out rate was 0.2 so we stick with it.

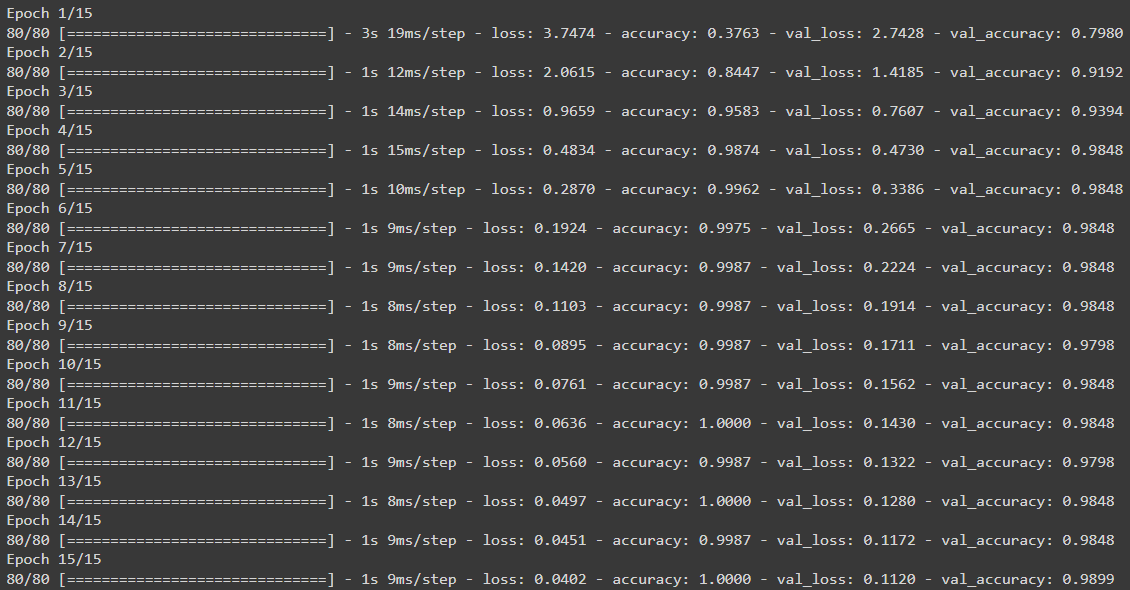
Next we go with Optimizers. We chose the next values:

1. Adam



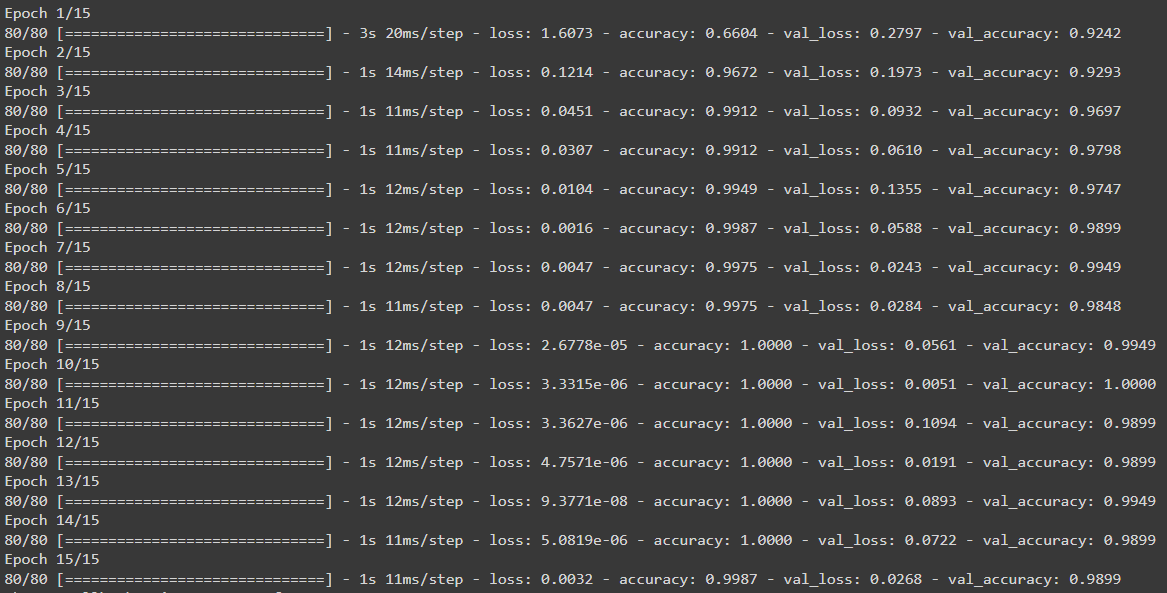
loss: 1.4503e-04 -accuracy: 1.0000 -val\_loss: 0.0189 -val\_accuracy: 0.9949

1. SGD



loss: 0.0402 - accuracy: 1.0000 - val\_loss: 0.1120 - val\_accuracy: 0.9899

1. RMSProp



loss: 0.0032 -accuracy: 0.9987 -val\_loss: 0.0268 -val\_accuracy: 0.9899

The best Optimizer was Adam so we stick with it.