ZERO DEFORESTATION MISSION DT-34

INTRODUCTION

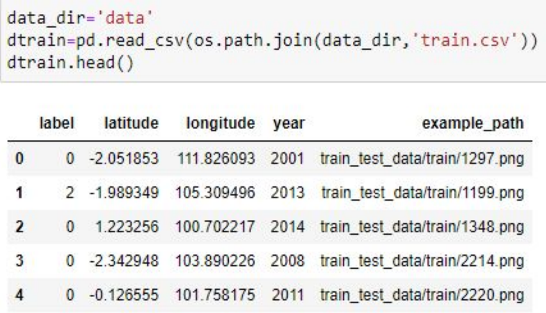
Today we will be coding a Deep Learning project where we need to classify images of different types of deforestation.

In this presentation we will be talking about which steps did we take to make it work. We must say first that this is a summary of our steps. In the notebook it is more detailed, so check that out :)

1. Load Data from CSV file.
2. Artificial Neural Network(ANN) vs Convolutional Neural Network(CNN)
3. Final Prediction
4. Conclusion

1.Load Data from CSV File

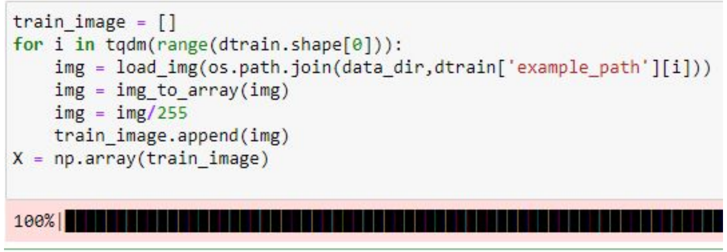
We will use panda to load 'train.csv'



```
data_dir='data'
dtrain=pd.read_csv(os.path.join(data_dir,'train.csv'))
dtrain.head()
```

	label	latitude	longitude	year	example_path
0	0	-2.051853	111.826093	2001	train_test_data/train/1297.png
1	2	-1.989349	105.309496	2013	train_test_data/train/1199.png
2	0	1.223256	100.702217	2014	train_test_data/train/1348.png
3	0	-2.342948	103.890226	2008	train_test_data/train/2214.png
4	0	-0.126555	101.758175	2011	train_test_data/train/2220.png

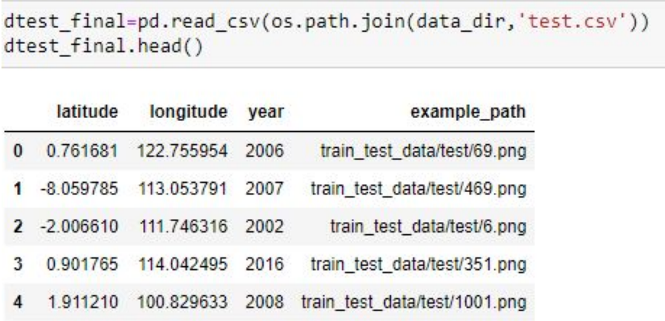
Create an array with all the images:



```
train_image = []
for i in tqdm(range(dtrain.shape[0])):
    img = load_img(os.path.join(data_dir,dtrain['example_path'][i]))
    img = img_to_array(img)
    img = img/255
    train_image.append(img)
X = np.array(train_image)
```

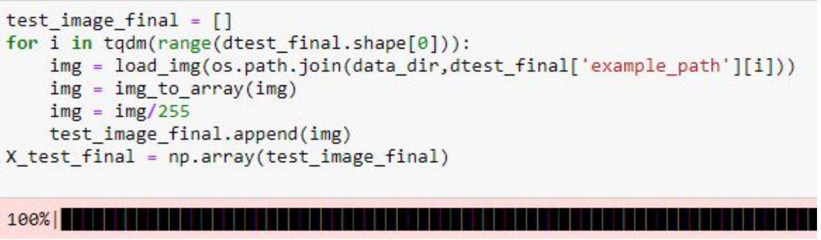
```
100%|
```

Loading 'test.csv' the same way as we did earlier



```
dtest_final=pd.read_csv(os.path.join(data_dir,'test.csv'))
dtest_final.head()
```

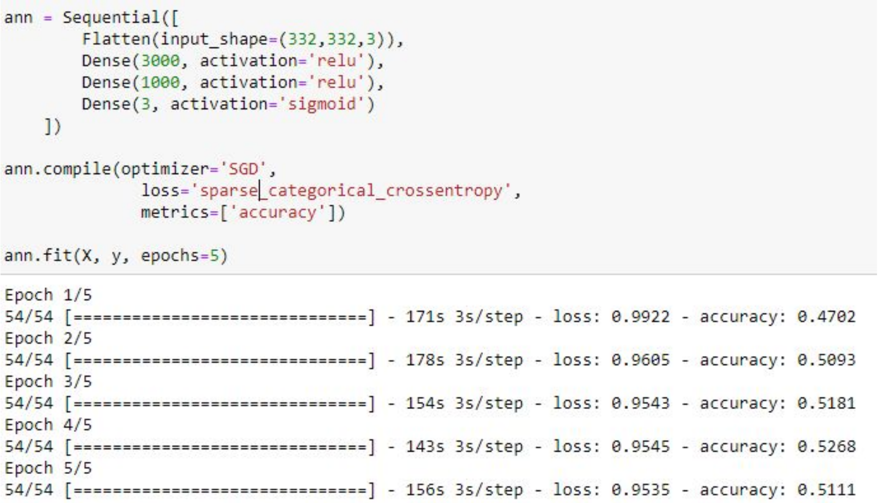
	latitude	longitude	year	example_path
0	0.761681	122.755954	2006	train_test_data/test/69.png
1	-8.059785	113.053791	2007	train_test_data/test/469.png
2	-2.006610	111.746316	2002	train_test_data/test/6.png
3	0.901765	114.042495	2016	train_test_data/test/351.png
4	1.911210	100.829633	2008	train_test_data/test/1001.png



```
test_image_final = []
for i in tqdm(range(dtest_final.shape[0])):
    img = load_img(os.path.join(data_dir,dtest_final['example_path'][i]))
    img = img_to_array(img)
    img = img/255
    test_image_final.append(img)
X_test_final = np.array(test_image_final)
```

```
100%|
```

2.1.Artificial Neural Network(ANN)



```
ann = Sequential([
        Flatten(input_shape=(332,332,3)),
        Dense(3000, activation='relu'),
        Dense(1000, activation='relu'),
        Dense(3, activation='sigmoid')
    ])

ann.compile(optimizer='SGD',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

ann.fit(X, y, epochs=5)
```

```
Epoch 1/5
54/54 [==============================] - 171s 3s/step - loss: 0.9922 - accuracy: 0.4702
Epoch 2/5
54/54 [==============================] - 178s 3s/step - loss: 0.9605 - accuracy: 0.5093
Epoch 3/5
54/54 [==============================] - 154s 3s/step - loss: 0.9543 - accuracy: 0.5181
Epoch 4/5
54/54 [==============================] - 143s 3s/step - loss: 0.9545 - accuracy: 0.5268
Epoch 5/5
54/54 [==============================] - 156s 3s/step - loss: 0.9535 - accuracy: 0.5111
```

What we need to do now is build our model in order to train it.

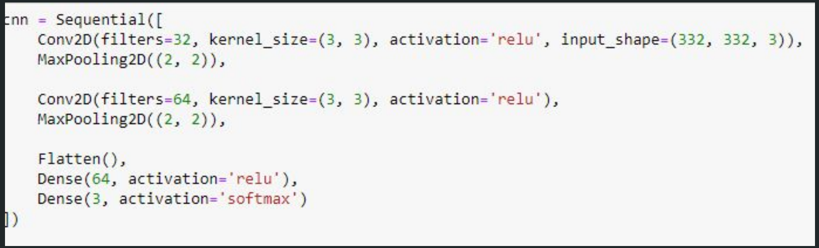
1. First we are building a simple artificial neural network.
2. Then we will compile it using a loss of 'sparse_categorical_crossentropy' because we have three types of deforestation. The activation is pretty standard both ‘relu’ and ‘sigmoid’
3. We are training it with 5 epochs.

2.2.Convolutional Neural Network(CNN)

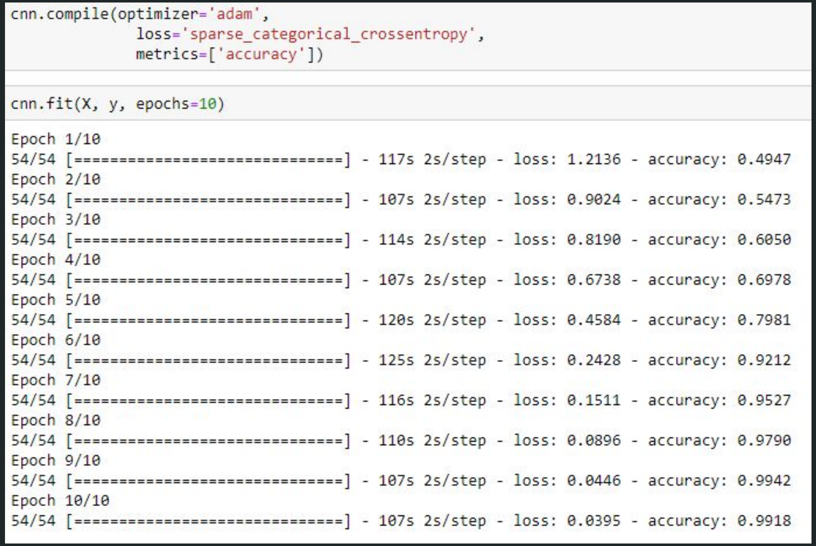
Now we are going to do it with deep learning, using convolutional neural networks. What will this network have?

- Two convolutional layers with a size of 3x3 each, 32 filter on one and 64 filter on the other and both using activation relu
- Two maxPool for each convolutional layers of 2x2
- And then our neural network as we saw before but changing the activation to softmax to get the predictions normalized

We will compile it using adam a popular optimizer for this kind of networks, with the same loss as before and metrics



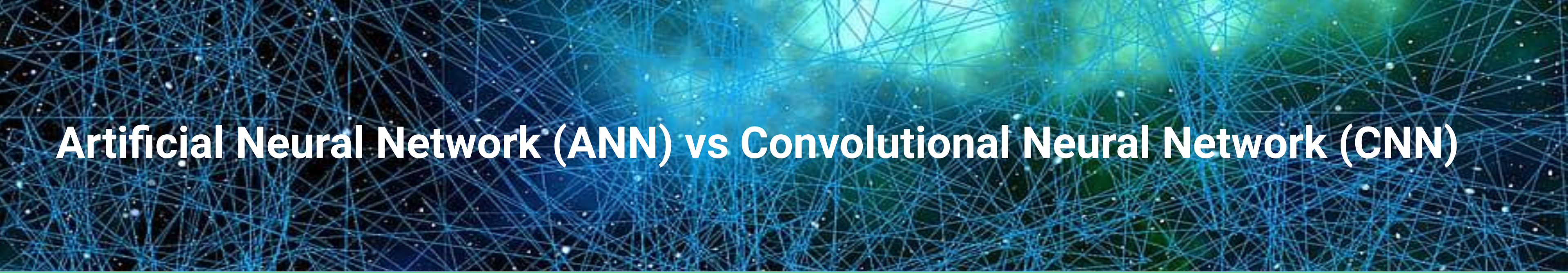
```
cnn = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(332, 332, 3)),
    MaxPooling2D((2, 2)),

    Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Flatten(),
    Dense(64, activation='relu'),
    Dense(3, activation='softmax')
])
```



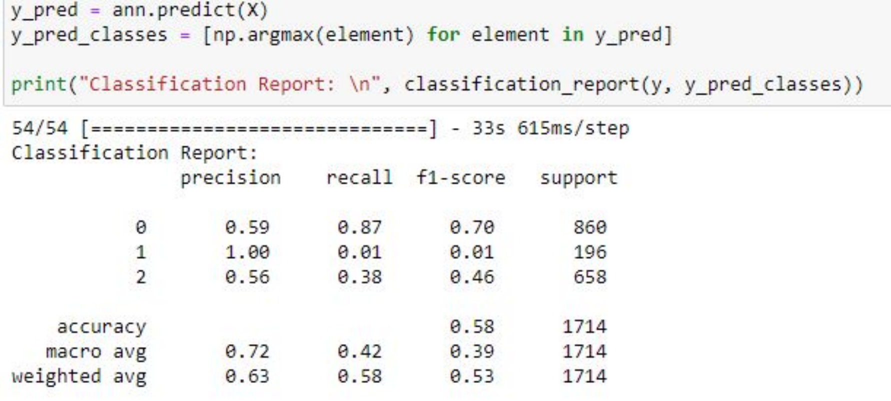
```
cnn.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
cnn.fit(X, y, epochs=10)
```

```
Epoch 1/10
54/54 [==============================] - 117s 2s/step - loss: 1.2136 - accuracy: 0.4947
Epoch 2/10
54/54 [==============================] - 107s 2s/step - loss: 0.9024 - accuracy: 0.5473
Epoch 3/10
54/54 [==============================] - 114s 2s/step - loss: 0.8190 - accuracy: 0.6050
Epoch 4/10
54/54 [==============================] - 107s 2s/step - loss: 0.6738 - accuracy: 0.6978
Epoch 5/10
54/54 [==============================] - 120s 2s/step - loss: 0.4584 - accuracy: 0.7981
Epoch 6/10
54/54 [==============================] - 125s 2s/step - loss: 0.2428 - accuracy: 0.9212
Epoch 7/10
54/54 [==============================] - 116s 2s/step - loss: 0.1511 - accuracy: 0.9527
Epoch 8/10
54/54 [==============================] - 110s 2s/step - loss: 0.0896 - accuracy: 0.9790
Epoch 9/10
54/54 [==============================] - 107s 2s/step - loss: 0.0446 - accuracy: 0.9942
Epoch 10/10
54/54 [==============================] - 107s 2s/step - loss: 0.0395 - accuracy: 0.9918
```

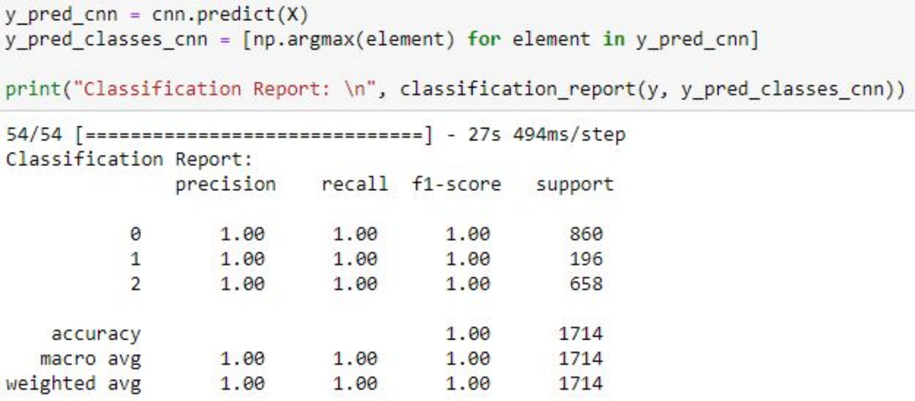


Artificial Neural Network (ANN) vs Convolutional Neural Network (CNN)



```
y_pred = ann.predict(X)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y, y_pred_classes))
```

```
54/54 [==============================] - 33s 615ms/step
Classification Report:
               precision    recall  f1-score   support

           0       0.59      0.87      0.70       860
           1       1.00      0.01      0.01       196
           2       0.56      0.38      0.46       658

    accuracy                           0.58      1714
   macro avg       0.72      0.42      0.39      1714
weighted avg       0.63      0.58      0.53      1714
```



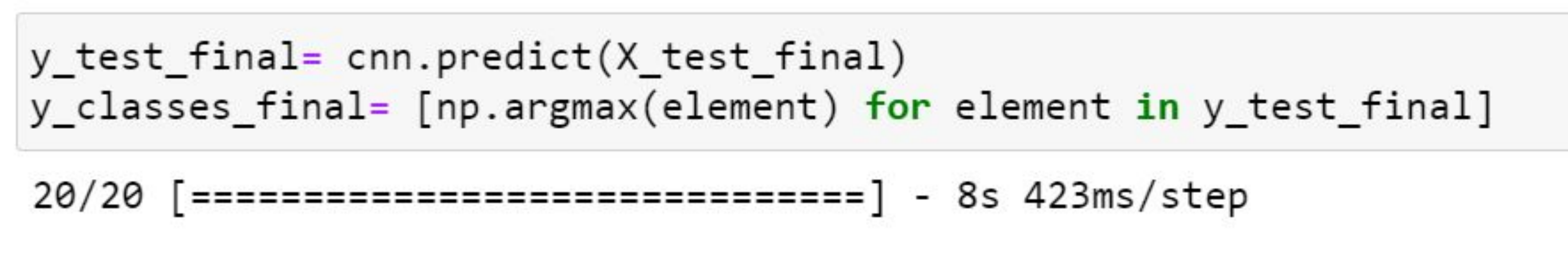
```
y_pred_cnn = cnn.predict(X)
y_pred_classes_cnn = [np.argmax(element) for element in y_pred_cnn]

print("Classification Report: \n", classification_report(y, y_pred_classes_cnn))
```

```
54/54 [==============================] - 27s 494ms/step
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       860
           1       1.00      1.00      1.00       196
           2       1.00      1.00      1.00       658

    accuracy                           1.00      1714
   macro avg       1.00      1.00      1.00      1714
weighted avg       1.00      1.00      1.00      1714
```

As we can see, the CNN is much better because it gives us a 100% of precision unlike the ANN that only reaches 58%

3.Final Prediction

We will predict with the data from the test csv.
Because we used softmax it will gives us the probability normalized of each type of deforestation. We are only interested in the one with the biggest probability so we will also take the maximum value of the probability of the three types for each prediction.



```
y_test_final= cnn.predict(X_test_final)
y_classes_final= [np.argmax(element) for element in y_test_final]
```

```
20/20 [==============================] - 8s 423ms/step
```

4.CONCLUSION

CNN IS THE WINNER

What we have learnt today is that CN networks are way much more efficient that AN networks. Leading us the first one to an accuracy of almost **100%**.

Because of that, in our predictions we used the CNN model and we were available to predict all of the test images.