Advanced topics in Deep Learning - Mid Semester Project

Roi Tzadok 212136618, shahar itzhaki 211492277

Submitted as mid-semester project report for Advanced topics in Deep Learning course, Colman, 2023

1 Introduction

In this project we attempt to prove the effectiveness of cnn. This is done by solving a multi-class classification problem. We have trained three different models with an increasing success rate. Each one of the models contains convolution layers, pooling (with stride) and neural layers.

1.1 Data

we were given a zip containing images of the size (224, 224, 3). every image was labels as one of the following categories:

- **Dark**
- **Green**
- **Light**
- **Medium**

1.2 Problem

There are four different types of coffee beans.

Given an image of unknown type of coffee bean we would like to be able to identify and predict the bean type out of the four options.

2 Solution

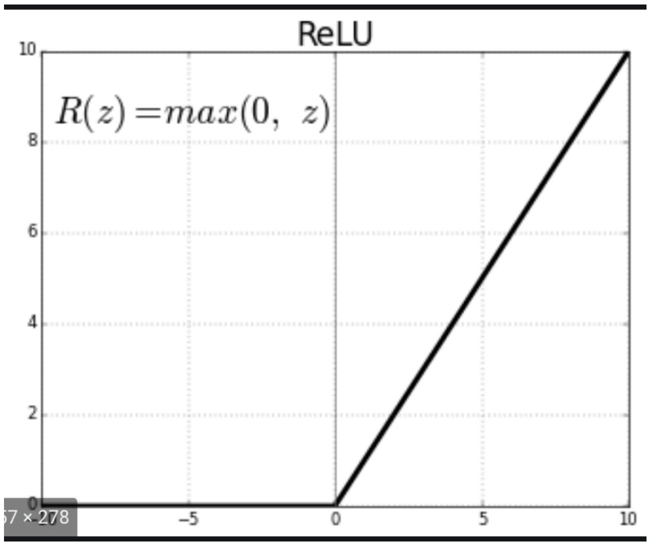
2.1 General approach

We have created a convolutional neural network in order to solve the problem at hand.

2.2 Design

The code is written in a python notebook and is cross platform. It was trained extremely quickly (a matter of minutes for each of the models) using google collab's gpu.
relevant concepts that were used in our code:

- The Adam optimization algorithm is an extension to stochastic gradient descent.
- the Relu loss function is a linear function that outputs the input directly if it is positive. otherwise, it outputs zero.



- cross entropy - similar to the binary entropy loss function we used in the previous exercise. However, in this exercise there are several classes.
- stride - in order to reduce the amount of data we have to process, we could use stride to help us reduce redundant data. Using stride means shifting the filter a few pixels forward before applying it again.
- pooling - taking groups of pixels (a fixed predefined kernel size) and perform an aggregation over them. Often we assume that pooling is combined with stride equal to the kernel size.
- max pooling - an instance of pooling. it's the operation of taking the largest value in a set of pixels.

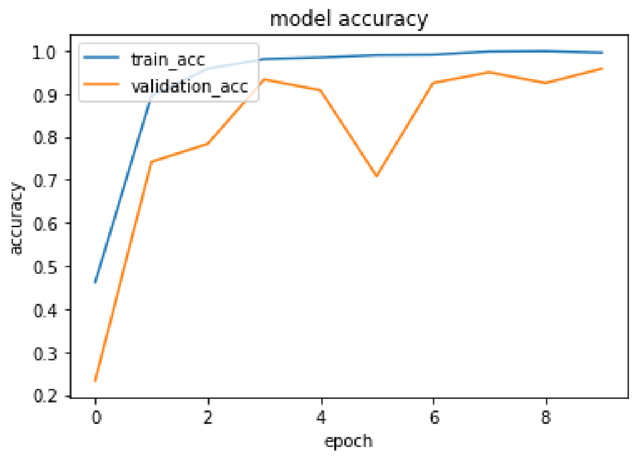
All of the models shared the same base concept:
Convolution layer(s) followed by max pooling. Then flattening the output to a single long vector which is then used in a fully connected layer(s).

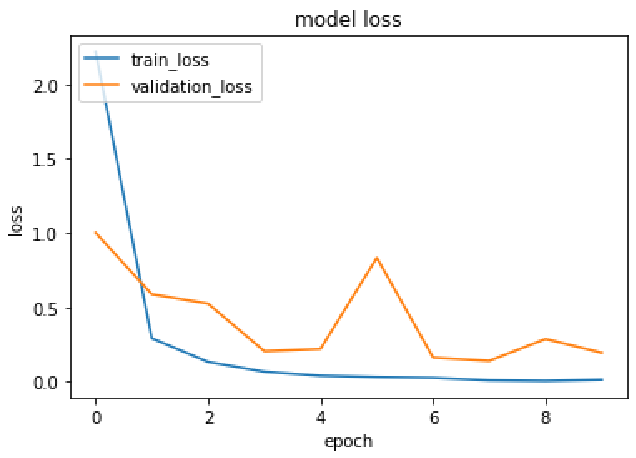
2.3 Base Model

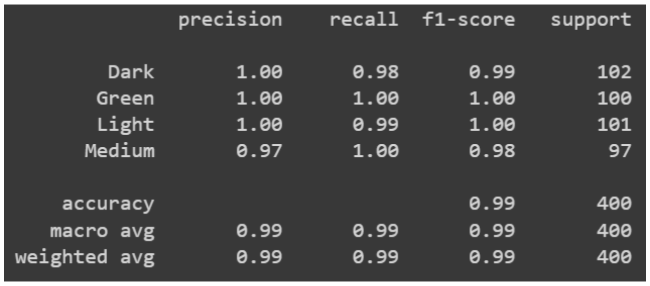
The base model is built from the following layers:

- convolution layer with sliding window of 3X3 and 32 outputs
- max pooling with a 2X2 windows and stride of 2 as well
- convolution layer with sliding window of 3X3 and 64 outputs
- max pooling with a 2X2 windows and stride of 2 as well
- convolution layer with sliding window of 3X3 and 64 outputs
- fully connected layer with 64 inputs (as the last convolution layer had 64 outputs)
- output layer, with four outputs - one probability for each class

The model was able to achieve an accuracy score of 0.9925.



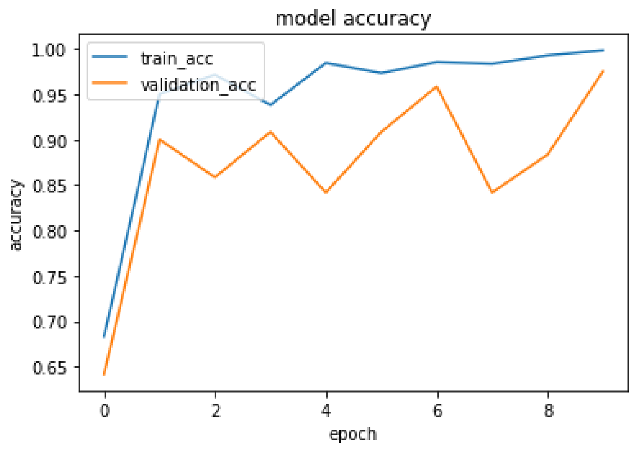


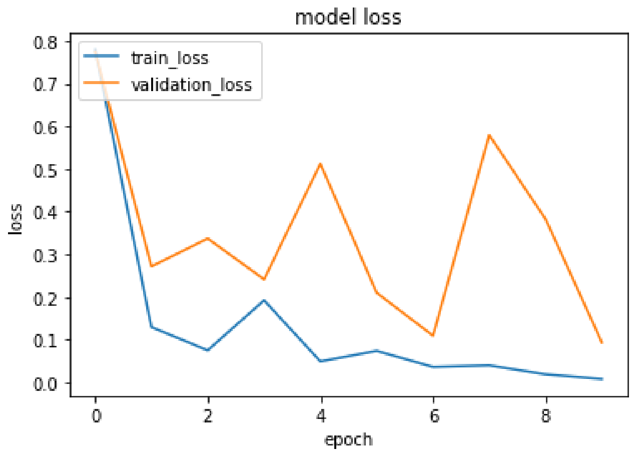


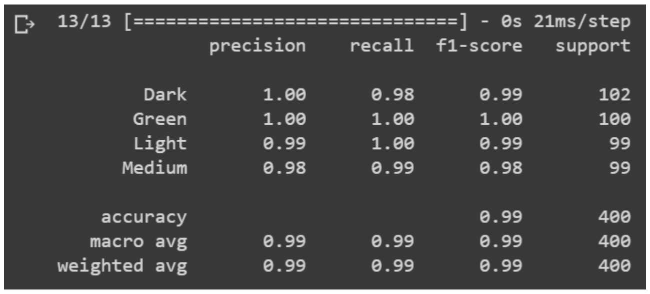
```
              precision    recall  f1-score   support

        Dark       1.00      0.98      0.99       102
       Green       1.00      1.00      1.00       100
       Light       1.00      0.99      1.00       101
      Medium       0.97      1.00      0.98        97

    accuracy                           0.99       400
   macro avg       0.99      0.99      0.99       400
weighted avg       0.99      0.99      0.99       400
```

2.4 First Experiment

In addition to the base model's layers, we have added another convolution layer and another fully connected layer. The new convolution layer had a kernel of 3X3 with 128 outputs. Therefore, the new fully connected layer had 128 inputs. The accuracy here was 0.9925 slightly better than the base model.







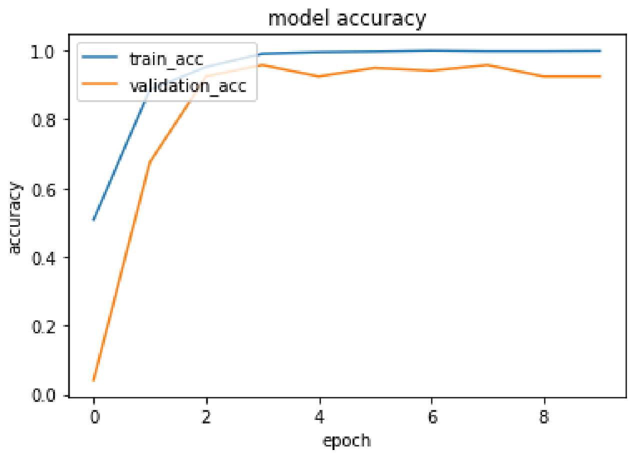
```
13/13 [==============================] - 0s 21ms/step
              precision    recall  f1-score   support

        Dark       1.00      0.98      0.99       102
       Green       1.00      1.00      1.00       100
       Light       0.99      1.00      0.99        99
      Medium       0.98      0.99      0.98        99

    accuracy                           0.99       400
   macro avg       0.99      0.99      0.99       400
weighted avg       0.99      0.99      0.99       400
```

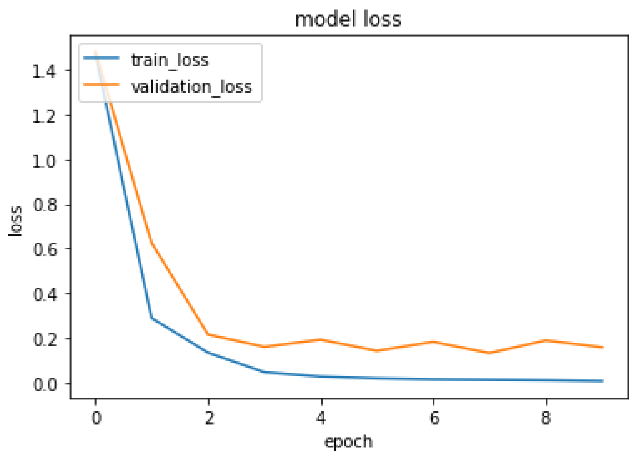
2.5 Second Experiment

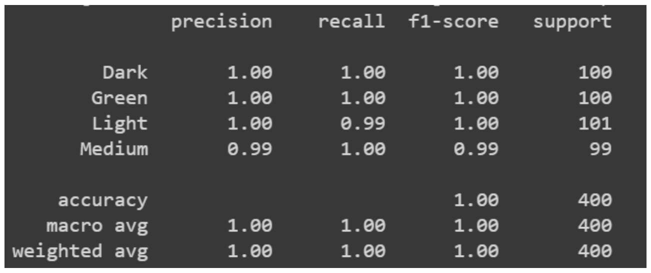
This was the best model of the three. In addition to the base model's cnn, we changed the learning rate to become dynamic. we created a callback function that decreases the learning rate at the end of each epoch. we tried two functions to do so.
The first sets the learning rate to be equal to 0.9 of itself every epoch.
The second sets the learning rate to be equal to the learning rate times 0.9 at the power of the current epoch.

we discovered that the former was able to achieve better results.

This model was able to achieve an accuracy score of 0.9975!

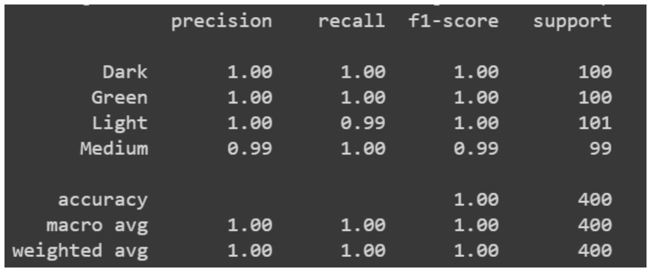





```
              precision    recall  f1-score   support

        Dark       1.00      1.00      1.00       100
       Green       1.00      1.00      1.00       100
       Light       1.00      0.99      1.00       101
      Medium       0.99      1.00      0.99        99

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```

2.6 Best model Results and Metrics

The second experiment has clearly shown the most promise with an accuracy rate of 0.9975.
model summary (layers and parameters wise) as displayed in keras's summary:



```
              precision    recall  f1-score   support

        Dark       1.00      1.00      1.00       100
       Green       1.00      1.00      1.00       100
       Light       1.00      0.99      1.00       101
      Medium       0.99      1.00      0.99        99

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```

the rest of the details (and graphs) are in the second experiment section. (the one just above this one)

3 Discussion

We have discovered the effectiveness of cnns. It is safe to conclude that the networks were able to achieve high level of accuracy even with relatively scarce

data set.
We have seen that a dynamic learning rate can be extremely effective. Thus, we can infer that other deep learning models could benefit from this discovery as well. Since the basic concept of finding a minimum value for a loss function is a common ground for the vast majority of deep learning algorithms.
Furthermore, from the first experiment we can see that there ought to be a limit for the number of layers in a model. The first experiment showed a slight improvement in comparison to the base model, nevertheless we are certain that any additional layers would have caused over-fitting.

4 Code

https://colab.research.google.com/drive/1mdxkBVxHSbFv6tS7dezkQ4yxqvVtjhk0?usp=sharing