Advanced topics in Deep Learning - Mid Semester Project

Ori Malca (ID. 315150599),Kfir Sitalkil ID. 208722660)

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

1 Introduction

In our project, we asked to build a model that will be able to classify 4 different classes. Each class represents a different type of coffee beans while they're main difference is the color (dark, light, etc)
Our motivation for facing that task was the opportunity to classify a multi class problem, after facing only binary classifying problems.

1.1 Data

The data we been received was a directory which contains a train and test sub directories.
Each of the sub directories contains 4 sub directories when each directory contains images of coffee beans of one of the classes.
The shape each image is 224X224X3.

1.2 Problem

Our mission was to build a classifier that gets an input image of coffee beans and outputs the class the coffee beans belongs to, with high accuracy rate.

2 Solution

2.1 General approach

Our approach was to divide the training data to training set(80 percents) and validation set (20 percents), augment and re-scale the images.
The names of the directories presented the name of the coffee beans class, we will use it to encode the labels.
To do it we used ImageDataGenerator which helped us to pool the data while making the division, augmentation and re-scale. We also gave him the 'categorical' attribute to class mode parameter to get the labels as 2D one hot encoded

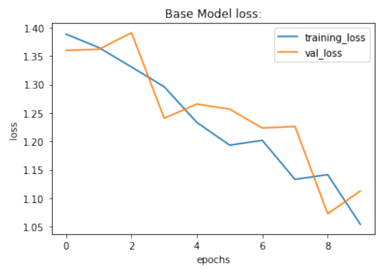
labels.
We also pulled the data in batches of 32 images in each batch.

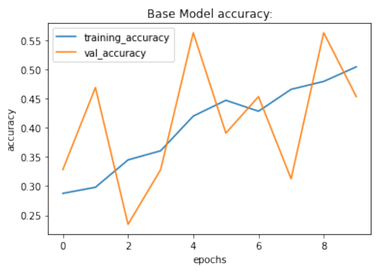
2.2 Design

Our model implemented with keras library.
The training time was not that long because there's no big amount of data, also we didn't need to train the model with big amount of epochs.
We cant think of any significant technical challenege because the library is very convinient for use, also the documantion is very detailed so we just needed to find the relevant section to the problem we were facing.
the loss function we chose to use was categorical cross entropy, which is a convex loss function that can fix the model's weights in multi class problems.
Our final model's architecture contains 7 layers, the first 4 layers are convolutional layers and pooling layers connected to a flatten layer which outputs a flatten that we send to the 2 fully connected layers.

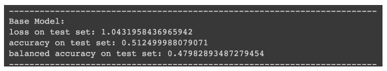
2.3 Base Model

To build the base model, we used an architecture we learned at class at ex.2 to get some base architecture and try to improve it with the performance we get from the experiments.
Of course we could not use the architecture as is, we needed to change the input shape to fit our data, also the activation function at the output layer was sigmoid which is good for binary classification, we changed it to softmax so the network could handle multi class classifying.
Our intuition was that this model with our adjustments will be good enough as base model, because we're not dealing with a too complicated problem. The hyper parameters we chose are the following:
batch size: 32 (arribitary)
number of epochs: 10 (we're not facing a too complex problem, 10 should be enough)
optimizer: SGD
loss function: categorical cross entropy (convex loss function which can handle multi class problem)
The base model results and evaluation metrics:

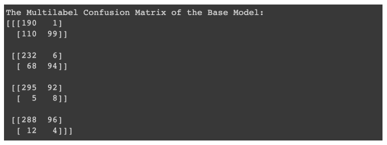




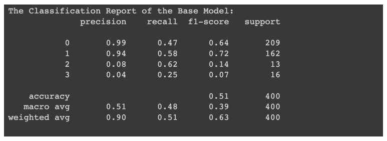
Base model evaluation metrics:



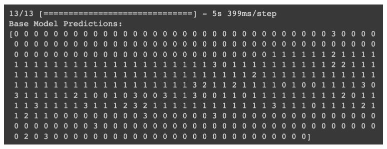
```
---------------------------------------------------------------------------
Base Model:
loss on test set: 1.0431958436965942
accuracy on test set: 0.512499988079071
balanced accuracy on test set: 0.47982893487279454
---------------------------------------------------------------------------
```



```
The Multilabel Confusion Matrix of the Base Model:
[[[190   1]
  [110  99]]

 [[232   6]
  [ 68  94]]

 [[295  92]
  [  5   8]]

 [[288  96]
  [ 12   4]]]
```



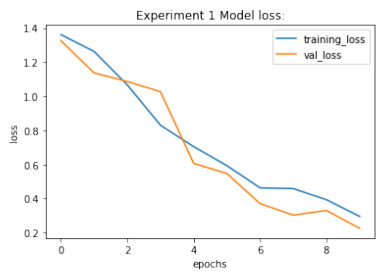
```
The Classification Report of the Base Model:
              precision    recall  f1-score   support

           0       0.99      0.47      0.64       209
           1       0.94      0.58      0.72       162
           2       0.08      0.62      0.14        13
           3       0.04      0.25      0.07        16

    accuracy                           0.51       400
   macro avg       0.51      0.48      0.39       400
weighted avg       0.90      0.51      0.63       400
```

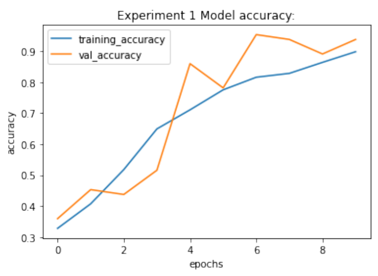


```
13/13 [==============================] - 5s 399ms/step
Base Model Predictions:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 2 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 0 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 2 1 1 2 1 1 1 1 0 1 0 0 1 1 1 1 3 0
 3 1 1 1 1 1 2 1 0 0 1 0 3 0 0 3 1 1 3 0 0 1 1 0 1 1 1 1 1 1 1 1 1 2 0 1 1
 1 1 3 1 1 1 1 3 1 1 1 2 3 2 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 0 1 1 1 1 2 1
 1 2 1 1 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 2 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

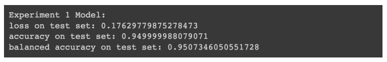
2.4 First Experiment

After testing the base model, we decided to increase the number of neurons at the first dense layer from 32 neurons to 64 neurons. At first we wanted our model to be light as possible, after the unsatisfying results, We decided to give our model more learn-able parameters to learn and maybe be able to solve our problem more efficiently. We chose to stick with one change so we will know exactly what's causing the change in the new model results.
The change we made increased our model accuracy while avoiding over fitting situation.
First experiment results and evaluation metrics:

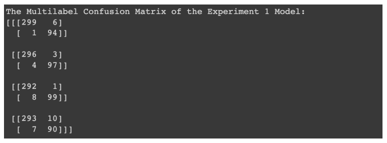




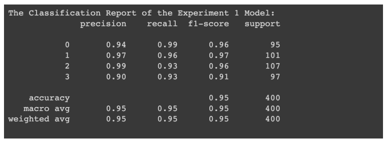
First experiment evaluation metrics:



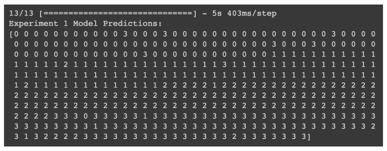
```
Experiment 1 Model:
loss on test set: 0.17629779875278473
accuracy on test set: 0.949999988079071
balanced accuracy on test set: 0.9507346050551728
```



```
The Multilabel Confusion Matrix of the Experiment 1 Model:
[[[299   6]
  [  1  94]]

 [[296   3]
  [  4  97]]

 [[292   1]
  [  8  99]]

 [[293  10]
  [  7  90]]]
```



```
The Classification Report of the Experiment 1 Model:
              precision    recall  f1-score   support

           0       0.94      0.99      0.96        95
           1       0.97      0.96      0.97       101
           2       0.99      0.93      0.96       107
           3       0.90      0.93      0.91        97

    accuracy                           0.95       400
   macro avg       0.95      0.95      0.95       400
weighted avg       0.95      0.95      0.95       400
```



```
13/13 [==============================] - 5s 403ms/step
Experiment 1 Model Predictions:
[0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 3 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 3 3 3 0 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2
 3 1 3 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3]
```

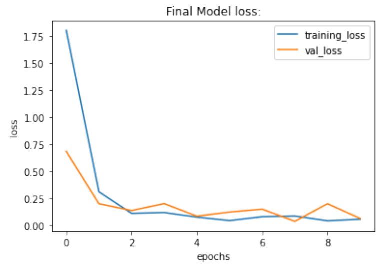
2.5 Second Experiment

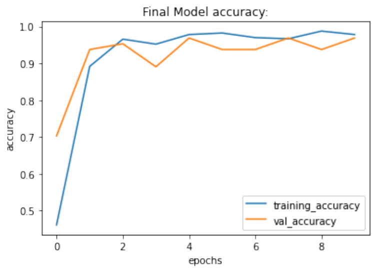
After the first experiment results, we decided to stick with the approach of making only one change so we will be able to know exactly what caused the difference in the model performance.
We chose to change the optimizer from SGD optimizer to Adam optimizer.
Adam is an extension of SGD that includes support for learning rates that are

adaptive to the parameters. Adam uses a moving average of the parameters, which helps to eliminate oscillations and dampen noise in the gradients, making the optimization process more stable.
After we changed the optimizer, we had significant improvement in our model, and we got our precision rates, also without getting to over fitting situation.
we chose this model to be our final model, and we will show the results in the next section.

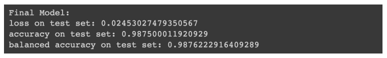
2.6 Best model Results and Metrics

As mentioned at the above section, in our second experiment we got significant improvement which led us to our desired model performance. We chose the second experiment model to be our final model because his performance was satisfying enough.
Final model accuracy and loss:

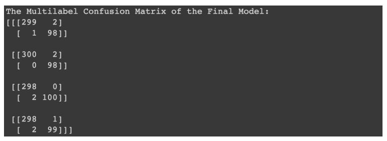




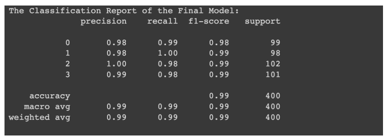
Final model evaluation metrics:



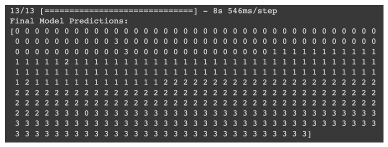
```
Final Model:
loss on test set: 0.02453027479350567
accuracy on test set: 0.987500011920929
balanced accuracy on test set: 0.9876222916409289
```



```
The Multilabel Confusion Matrix of the Final Model:
[[[299   2]
  [  1  98]]

 [[300   2]
  [  0  98]]

 [[298   0]
  [  2 100]]

 [[298   1]
  [  2  99]]]
```



```
The Classification Report of the Final Model:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98        99
           1       0.98      1.00      0.99        98
           2       1.00      0.98      0.99       102
           3       0.99      0.98      0.99       101

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



```
13/13 [==============================] - 8s 546ms/step
Final Model Predictions:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 3 3 3 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
```

3 Discussion

Provide some final words and summarize what you have found from running the experiments you described above. Provide some high level insights. In conclusion, after implementing our first neural network with numpy only, we really enjoined using keras frame work and see how easy it is to build a model if you understand the way that neural network works, the different meanings of the network architecture, hyper parameters, etc.
Our insights from this project: Deep learning can be used to solve complex classification tasks, such as identifying the type of coffee bean. Convolutional neural networks (CNNs) are well-suited for image classification tasks, and can be enhanced with techniques such as data augmentation to improve performance.

It is important to have a diverse and well-labeled dataset in order to train a successful deep learning model. Evaluating the model's performance on a holdout test set is crucial for getting a sense of how the model will generalize to unseen data.

4 Code

Link to colab notebook:
https://colab.research.google.com/drive/1p21X9Jtbo5phK7JThm7suKE-9E7W469m?usp=sharing