# Project 2 - Computer Vision

Student Name: ido bistry

Submission Date: 24/07/2024

## **Question 1**

|  |  |
| --- | --- |
| Plot of train/validation accuracy per epoch: |  |
| Plot of train/validation loss per epoch: |  |

Question 2

Based on the plots,

1. Do you think that training for more epochs will improve the results? Explain.

No, and there are two reasons :

* + - 1. as we can see the validation and training are drifting away, both for loss and  
         for accuracy.   
         this points to us about an **overfitting** issue, when the **training** is getting better   
         but **validations** is not showing same progress and just drifting away.
      2. We are using *base\_model* that is aggregation of *MobileNet* , and within the code we set ***freeze*** on the weights (*base\_model.trainable = false*). This means that even if there was a productive learning from the training data – it wouldn’t had happened

1. Do you think that increasing the dataset will improve the results?

Since we have an overfitting is not only a case of lack of data, but also a lack of model complexity. I guess that model is not complex enough for this data and we might need also to add not only data – but also some layers to improve learning

## **Question 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Training  Time (sec) | Test  Accuracy % | Explain why accuracy has or has not improved | Explain why the training time is slower, faster or hasn’t changed |
| **-** | 97.6 | 0.10 | – | – |
| **A** | 351.32 | 0. 49 | accuracy improved since we added learning to the training phase, | Slower since the training also added learning. Before the training was false so no *back propagation* was done |
| **B** | 361.07 | 0.45 | accuracy decreased since Learning rate increased X10, and each time the loss was calculated it was calculated with worse granularity – hance more risk of missing in tuning calculation | Time increased (a bit) since each time back propagation was executed the calculation of weights has bigger differentiation hance more weight to process I guess not so much time – since dataset is not huge |
| **C** | 360.86 | 0.43 | Accuracy decreased Since dropout – we have less complex  neurons to process learning | Time decreased since we had a dropout – less neurons to calculate. Alas, I would expect much less time |
| **D** | 379.33 | 0.4730 | Adding augmentation flip helped a bit – since we have more augmented data to measure  we try to prevent overfitting, but having progress just a bit , might suggest we don’t have enough complex model, after dropping 0.5 neurons before, we wanted to prevent overfitting, but now have inadequate model | Time increased since now we have another layer to calculate (and much more synthetic data) more neurons layers to calculate , meaning more backward propagation calculation (and on more data…. ) |
| **E** | 375.43 | 0.1 | Resnet accuracy is worse, maybe because we need some more hyperparameters set up or maybe some fine tuning will help.   we are looking at a nerwork that is overfitted, even before and we might not have enough data for his complexity added | Time decreased but not much.  as I read , we are using more convolutional power and less regular computing.  mobile net has 16 layers of convolutional calculation (according to CHAT GPT) and other 13 regular calculation – total 29, while resnet has only 18 layers of convolutional computation   so if I use the right hardware with GPU – the convolutional calculation might take me faster  tensorflow/core/grappler/optimizers/custom\_graph\_optimizer\_registry.cc:114] Plugin optimizer for device\_type GPU is enabled. |

Application architecture:

1. application can load with parameters and without parameters :

$ python main.py -–help

usage: main.py [-h] [--execution\_set EXECUTION\_SET] [--log\_level {CRITICAL,FATAL,ERROR,WARNING,WARN,INFO,DEBUG,NOTSET}] [--execution\_path EXECUTION\_PATH] [--use\_gpu {True,False,true,false,TRUE,FALSE}]

options:

-h, --help show this help message and exit

--execution\_set EXECUTION\_SET

execution list of profiles to execute (default 3,3.A,3.B,3.C,3.D,3.E

--log\_level {CRITICAL,FATAL,ERROR,WARNING,WARN,INFO,DEBUG,NOTSET}

log level to use (either one of : DEBUG, INFO, WARNING, ERROR, CRITICAL) default INFO

--execution\_path EXECUTION\_PATH

path for logs and plots default ./resources/executions/

--use\_gpu {True,False,true,false,TRUE,FALSE}

use gpu if available or cpu only (default : True )

1. default execution will execute all of the profiles (exercise 3 with all sub letters )
2. after execution, by default – al plots and logs of execution will be saved at ./resources/date/time/ directory
3. all of the profiles to be executed are saved in ./profiles.py file

profiles.py has a json that describe profile inheritance :

first profile – 3 is inheriting the configuration frombase , and then each profile inherit its previous , with override of the requested change according to the exercise demand   
  
on start the application check which profiles to execute, according to selected parameter “execution\_set”

and then run on the profiles described in profiles.py file and if this profile is selected to execute – it will be executed,

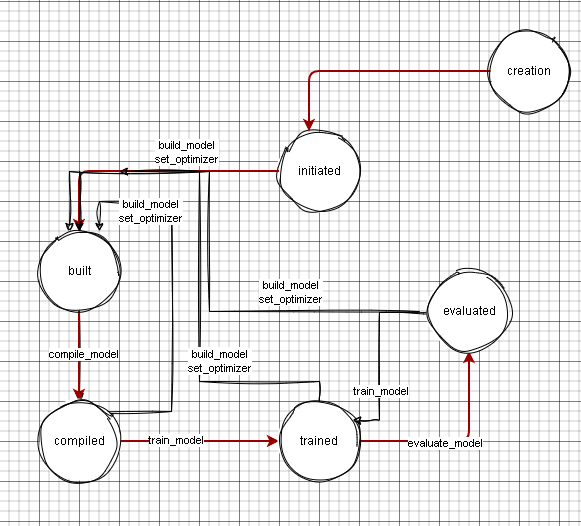
then load next profile and override the previous configuration

1. in he application a Profile is abstract class that manipulate the DNN model to be executed   
   I hold a factory class : *ProfileModelFactory* that according to the configuration *model* and *sub\_model* attributes – loads the derived class that maintain specifications for loading the base relevant model(*MobileNetModel/ResNet18Model*)



1. the profile class has a restricted life-Cycle, regulated by specific statuses – that change after each of the instance actions

once in a specific state – instance can not do specific actions.



1. pfa log of all executions one by the other



1. pfa plots crated for each of the executions :

3.

