Cairo University **Faculty of Computers and Artificial Intelligence**

Third year (2024-2025) **Supervised learning (Spring 2025)**

Project

Programming Language: Python

Teams

You will be in teams of **Five**, what is written in the report should be tested by all team members but only write one result in the report (you can choose whose result to write). This means that all team members should be able to run the code if asked to on google collab, furthermore all team members should have the code stored on google collab and ready to run or answer questions about it if asked to.

If a team member happens to not have any runnable code relating to this report (*The code should **not** contain errors, since a code can't really fully run with an error and hence no results could have been displayed and taken. Meaning the person with the error didn't try it*) or can't answer questions about parts of the code or can't elaborate on something on the report then that student will have grades deducted (all members should know what is written in the report and why they said that).

So, **one** report (include the name and ID of all members somewhere in the report), all students should have tried the code and understand what is written in the report (you can share opinions, this is essentially what a team is for).

General Description

You will be using keras (with the default tensorflow backend) to study the different effects of adding different CNN and ANN models . In this assignment you are required to **submit a report** on your study, you should include the code you implemented for the first study and for all the other implementations, you can either **include the part that you changed or the full code**. Use headings and titles to separate each part/study of your report. Submit it in both word and PDF formats. The purpose of this long description is to organize your study.

Report Requirement

For each study you should test training with different number and size of CNNs layers, different number and sizes of FC layers, dropout, try different batch sizes (use different numbers bigger than 30), different epochs, different activation functions and finally the different optimizers (**Continue reading to know how to go about testing all this**).

A. To implement/study the above you should go on about it as the following steps:

0- Include shuffling in your data (this is usually by setting a parameter to true)

1- YOU MUST START WITH SIMPLE **ANN MODEL AND SVM AND APPLY THEM TO THE DATA independently**

2- SECOND YOU START WITH CNN

3- Next you will **build with the settings of your choice**, be sure to start with **ReLU** as the activation function for each layer that you include (All your models should have at least 3 layers total). For the CNNs remember to use a **2D max pooling layer** after at least one CNN layer, your 2D max pool layer must have a configuration of 2x2 stride and 2x2 kernel size. For the first model, keep testing the number of epochs to find a suitable one your satisfied with. For the optimizer start with **SGD** (choose your own momentum) and you can start here with any learning rate. (don't use a dropout layer yet)

4- Next you will also test different learning rate.

5- Then you will test changing the number of CNN and parameters of the CNN (*beware CNNs are memory and computation heavy depending on the configuration you give them; do **NOT** using more than 3 CNNs in any model you test and **NOT** more than 4 FC layers*) along with changing the number and sizes of fully connected layers. Your objective in this point is to get a good model with less parameters and memory footprint (try to balance both out, you should not need a strong model). You can also work on choosing a good number of epochs. Also start with a small batch size like 32 or 64.

6- Now you will try 2 different batch sizes, if your batch size for the previous step is 'b', test batch size of 2*b and (4*b or 3*b).

7- Now try changing all the activations to something else and train the model. Use at least 3 other activations and observe the changes (info on what you should observe in all the steps is in segment 'C').

8- Now with the best settings you've reached, try 2 more optimizers.

9- Put a dropout layer in the model, anywhere you see fit, try 2 places and test at least 2 different dropout rates (that aren't 0% or 100% of course).

10- Your done, now you still have to read the below to know what to write (two parts), and what to use if you don't have computing power.

B. First, what will you write in each model you test:

0- The code of course as state in the general description section (read it)
1- Final accuracy of the model and the accuracy in the first 5 epoch
2- The number of parameters in the model (calculate this)
3- The average time to train in each epoch.
4- The average test time in each epoch.
5- The layers of each model (including activations).
6- The learning rate used and configuration of the optimizers
7- You are not required to test learning decays, if you wish to include it, then include it.
8- The optimizer used with its configuration.
9- If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it)
10- At the end of each step/series of test on a parameter. Write what you observed has changed due to the parameter you changed (according to the above part).
11- Which value for the parameter gave you the best result and why do think this happened.

C. **Second, the observation to focus on in each point 'A's** (This explains the goal for each step in Segment 'A', The points here in Segment 'C' follow the same order of Segment 'A', so point 1 refers to point 1 and so on [except point 0]).

0- Don't forget that for each model you test, you will write down what is in required in segment B (the list above).
1- Start with a good enough model, here you are testing the effect of number of epochs on your parameter. Find a good number of epochs here and stick with it for the next steps. Write how the epochs you tested affected the model and what epochs you found best. You should start with a batch size of 32 or 64.
2- In the second step you will use the best epoch you found (it should be at least 10, also 10-15 should be good, but depending on your model you may need a few more, **few here refers to numbers you can count on your hands**). And then try different learning rates. Write how each learning rate affected the learning of the model and what learning rate did you see fit better.
3- After finding an appropriate learning rate. Now it is time to test changing the model. Try adding or removing layers and see how it affected your model. You should try at least 4 more models and write down the model that you saw gave better results. Better here should be in terms of a mix of both more accuracy and less size/processing needed.
4- Now you have your good model (along with the learning rate and # of epochs), you want to test other batch sizes. In here you will test 2 other batch sizes and note the difference in training the model (In here you will need to focus on more points in segment 'B' while comparing). Write the differences you saw from using a different batch size.

5- From the last step you can fix the batch size to whatever you found to be better. In here you should change **ALL** the activation functions to another activation function of your choice. You will test 3 other activation functions of your choice, but at least one of them should be sigmoid (this is important). Write down all the differences you'll note in training.
6- Now revert to all activations being ReLU (or if you found a better one use it). And test using 2 different optimizers other than SGD. Write down the difference you'll see.
7- In here you are testing the effect of adding a dropout layer with different dropout rate. (remember to test in 2 different locations and 2 different rates as said above).
8- **Note that after each step, you will set the parameter to the better result you found and continue to test another parameter.** For example, in the first step say I found 13 epochs to be best, I'll use 13 epochs then test the learning rate, say I found 0.0001 to better, then I'll use it. In the next step I have epochs set to 13 and learning rate set to 0.0001 and will be checking for a better model (by trying different layers). And so on in the manner.The total models you'll test in minimum (in case you wanted to calculate it): 3+3+5+2+3+4 = 21 models.

Dataset properties

You will use the **MNIST** dataset, the shape of the single input should be 28x28 not 784 (since you'll use CNNs). The output layer size is 10, each one represents a class (you will find tools to automate a lot of the process).

Additional Note

- At the end of the report write the best model you've achieved and why you think it performed better than others
- The loss you'll be using is the cross-entropy loss (it is the standard loss in this problem)
- **An activation function/layer DOES NOT COUNT AS A LAYER**. They call it a layer in regard to their design pattern. The soft max is not counted as a layer.
- You are not required to say why you chose your starting model in step 1. But you will comment on the changes that will happen when trying different parameters in each step.
- Parameters or hyper parameters are parameters of the model that you can change and tweak (like the size of the model, the learning rate, # of epochs, etc..). Each step is concerned with testing each parameter finding the better one and then using it.
- You should make sure the code works first, then start modifying the models and studying different results (it should not take effort to do the modifying).
- You should print information of segment 'B' by default in each run so that you can copy it from the output each time easily. You will write this info for each model you will include.
- You will comment on the result of the single model, and difference of the results between models in a single step (when testing the same parameter).

- The **test batch size** can be bigger than the train batch size. There is nothing to study in different test batch sizes, because the model isn't learning anything new, and the variation only affects getting the accuracy results faster. Also test batches don't carry the burden of keeping gradients so they are more memory friendly to compute. **Note I am talking about the test batch size**, not the train batch size. Do note to add to the study anything about different **test batch sizes**.
- You will use keras for the assignment. Don't use pytorch or any other machine learning framework. If you already learned to use another framework, then it is time to learn to use keras

Don'ts

Don't use a batch size bigger than 250 or smaller than 32, a dropout bigger than 85%, a fully connected size bigger than 512, a CNN with kernel of size more than 5x5, a max pool layer **with** anything different than 2x2 stride and 2x2 kernel size. And most importantly, don't use more than 128 channels for the CNN ((عشان كده انت هتبقى مفتري(ة)).

These constraints on the study of hyper parameters, is to avoid reaching extreme conclusions due to (maybe) your limited knowledge. You can test these out later outside the assignment if you have the patience and wish to.

Processing power

Use google collab. There is a solution to use your own PC, but most of you may not have a GPU or may not be able patient enough to get the GPU to work (and this training of MNIST on CPU multiple times will take A LOT more time [could be days], than GPU).

So, use google collab as it already provides a GPU most likely better than what you would have and will work on the go (Even if you have an RTX it still is faster to get started on google collab, than to download, setup and debug what you will need to make keras work, unless you already know what to do).

Finally

- I did test a run of a model on MNIST on google collab. So, I know that aside from the time to get your code to work initially for the first time (didn't consider it), for the at least 21 models you'll test it should not take more than a few hours in total (all of them combined). Provided you follow the "**Don'ts**". And a plus side is if it takes time, you still can run it and focus on something else (productive I hope).

- **<u>Only</u> team leader should submit the report and the whole code(in one notebook) in this way:**

Ex code : id1_id2_id3_id4_id5.ipynp

Ex code: id1_id2_id3_id4_id5.pdf