**Hyperparameter tuning with Neural Networks**

**Due: 11-07-2021 at 11:59:59 pm**

**Last updated: 10/25/2021**

**Objectives:**

1. Learn about different hyperparameters and their impact on the neural network performance
2. Learn about Keras and its ability to structurize the complicated steps of neural network
3. Visualize and tune the hyperparameters
4. Grid search on hyperparameters and find out the best set of neural networks for any task
5. Transfer your neural network hyperparameter optimization knowledge to real-world projects.

**Instructions:**

1. For this task, you must use the Keras (with TensorFlow backend) platform
2. We will use the MNIST hand-written digit recognition dataset for our task. Keras has this dataset in its *datasets* library.
3. A *.ipynb* notebook file is provided with the instructions. Please read it carefully and perform the tasks as specified
4. A simple neural network has many Hyperparameters. To list a few, the number of layers, unit size in each layer, batch size, learning rate, number of epochs, choice of the optimizer, choice of loss functions, dropout rate, regularization, and so on.
5. We will choose accuracy as the performance metric for the sake of simplicity. Please note, for an imbalanced dataset, it’s also important to report the F1-score, recall, and precision. But we discard them for this task.
6. Also for simplicity, we attempt to find the best set of hyperparameters based on the performance of the validation set. The validation set is a randomly chosen 20% split from the training set.
7. Neural networks have many stochasticities. In the given *.ipynb* notebook, we provided a SPECIFIC seed value so that your experiment is reproducible. You must not choose a different seed.
8. We recommend you use google collab for this task. It’s easy and less complicated and doesn’t require any installation. However, just for downloading *.html* file, you still may have to install anaconda in your local computer. If you use your local computer or anaconda installation, please try to reproduce the experiments provided in the *.ipynb*  notebook and make sure you have exactly the same result as ours.

**General Note on Deliverable:**

There should be only one .*ipynb* file and one *.html* file in a zipped file. Name the folder as *Lastname\_Firstaname\_task3*. That means all three subtasks will be in One code. But there will be 3 pdfs, one for each task. Name the pdfs as *sub1.pdf, sub2.pdf, and sub3.pdf.*

1. **Your code**: your code must be in the *.ipynb*  form. It must show all the outputs generated.

2. **The .*html* file**: From your *.ipynb*  notebook, you can also download the .*html* file. Please include that with your submission. If you use google collab, you may not be able to download the *.html*  directly. You have to download the *.ipynb* in your local computer and upload it in the local computer’s Jupyter notebook, and then you can download the *.html* from there. The *.html* should show all your outputs as well.

**3. The pdf file:**  For each subtask, you have to answer the given questions and explain.

**Subtask 1:**

For subtask1, you have to “tune” one hyperparameter and pick a fixed specific set of 4 hyperparameters. The detailed instructions are provided in the .*ipynb* file. You will visualize the performance by plotting them.

**Deliverables for Subtask1:**

**A pdf on the discussion**: Submit a pdf describing how your code works and how you handled the task. Your discussion should have the following:

* With the increase of the batch size, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification
* With the increase of the layer size, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification
* With the increase of the unit size per layer, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification
* With the decrease of the learning rate, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification
* With the increase of the epoch number, do you see the accuracy improve or deteriorate? Is it consistent for all cases? Attach the screenshot of the graphs with your justification

**Subtask 2:**

You will perform a grid search of hyperparameters. In the *.ipynb* notebook, we provided the specific values of hyperparameters; you have to find the accuracy for all the settings, make a table (pandas data frame) of that and sort the table in descending order. In the end, you also comment on the five best and five worst hyperparameter settings.

**Deliverables for Subtask2:**

**A pdf on the discussion**: Submit a pdf describing how your code works and how you handled the task. Your discussion should have the following:

* Provide a screenshot of the first five rows and last 5 rows of the panda’s data frame table that you outputted from subtask2. Summarize your findings in 2-3 sentences.
* interpret the content of the generated data frame with respect to settings of the five parameters that are associated with higher accuracy (e.g. Do 4-layer neural networks have higher accuracies than 2/3-layer networks?)
* statements about parameter settings that do not have much impact on accuracy (e.g. different chosen batch size parameters settings may not seem to have much impact on accuracy). Note that for some hyperparameters the reported findings might be inconclusive; in this case, state if that’s inconclusive.

**Subtask3:**

You will perform a grid search of some advanced hyperparameters, namely, Dropout rate, Kernel Initializer, and Optimizer. In the *.ipynb* notebook, we provided the specific values of these advanced hyperparameters; you have to find the accuracy for all the settings, make a table (pandas data frame) of that and sort the table in descending order. In the end, you also comment on the three best and three worst hyperparameter settings. For other parameters, use the best ones you found from subtask2.

**Deliverable for Subtask 3:**

**A pdf on the discussion**: Submit a pdf describing how your code works and how you handled the task. Your discussion should have the following:

* What is the meaning of “dropout rate”? Explain in 2 sentences how dropout is done in the training process.
* Does an increased dropout rate have any impact on performance? Is it consistent?
* Explain the results you found with ‘zero kernel initializer’. Would you have gotten the same result if you used “-1” instead of “0” for the kernel initializer?
* Explain the different results obtained with different Optimizers. Is there any winner? And if there is a winner, does it outperform the other optimizers in all the cases?