**Honor code.** This assignment is individual work. The goal of this assignment is for you to put in practice the concepts we learned in the video recordings, as well as explore complementary concepts. As mentioned in the synchronous lecture, academic integrity will be strictly enforced. If for any reason you are tempted to cheat (i.e., because you are facing personal hardship), contact the instructor immediately by email.

**Instructions.** You need to submit 2 things:

1. On Crowdmark, answers to each question:
    - for questions without code, as usual (you can take a screenshot of your answer in the colab cell or write it down on a separate medium)
    - for questions with code, take a screenshot of the code and upload it to Crowdmark
2. On Quercus, submit a py file with your entire source code

To facilitate grading, please continue to follow the following guidelines when uploading your assignment to Crowdmark:

- On Crowdmark, you will upload a *separate* image for each question.
- You can use any tool you want to generate these answers (e.g., word, LaTeX, scan your handwriting), but each image should be easy to read and oriented properly.
- If you decide to handwrite your assignment rather than typeset it, ensure your handwriting is readable otherwise TAs will have the discretion to not grade your answer.
- Graphs produced should be clearly interpretable. Include labels on axes and a legend.

**Assignment structure.** The assignment contains 17 questions worth a total of 27 points. There are four parts to this assignment:

Part 1: Building a CNN

Part 2: Training and Tuning a CNN

Part 3: Try Out a New Dataset

Part 4: Open Exploration

A starter code written in Objax is provided to you here:
`https://colab.research.google.com/drive/1aZD5c7Jfm5JdQdPzYl4Vi2PvtK5UiFAS?usp=sharing`

You will proceed to run and/or modify this code throughout the assignment. A few preliminaries before running the experiments:

- You are required to know what a convolutional neural network is, and how it works.
- Some basic familiarity with Numpy and Python classes. Some basic familiarity with Jax. Recall that tutorials were organized earlier this semester to help you prepare for this assignment.
- Prior experience with Pytorch, Keras, Tensorflow or Objax is helpful but not necessary.

**Before running any experiments, make sure that you have GPU enabled for the notebook. This setting can be found under:** `Tools` → `Settings`

**Online resources.** You may use online resources such as Objax's tutorials, but if you are unsure if it constitutes plagiarism, please reach out to the instructor and in any case provide a link to any references you used to write your solution.

**Part 1. Building a CNN.** The starter code provides you with a data loading procedure and a base convolutional network model (`Base Model`). Study the code carefully and answer the following questions.

1. (1 point) Your data has been split into training, validation and test set. Examine the ratio of the split and number of examples in each set. Suppose you were to train on batches of 32 examples each. That is, in each step of gradient descent, you randomly select 32 examples from the training set, compute your average loss on these examples, and then compute the gradient of this average loss with respect to the model parameters.

   How many iterations will it take to go through the entire training set given the number of training examples yielded by the data split? How many iterations are there in 30 epochs? Recall that one epoch is the number of iterations needed to train over the entire dataset.

2. (2 points) Fill in the code for your custom convolution filter and show that it returns the same output as Objax's own convolution routine.

3. (1 point) Fill in the code for your linear layer, and show that it returns the same output as passing through Objax's own linear layer.

4. (1 point) Explain in a short paragraph what is the difference between the training and validation set.

**Part 2. Training and Tuning a CNN.** For this part, you have been given a starter code. Navigate to the portion where you define your optimizer.

1. (1 point) Complete the optimizer by using the definition of (stochastic) gradient descent: $w_{k+1} = w_k - \epsilon \nabla L(w_k)$. Note that you need to update `params.value`, which are the values of the trainable variables of your model.

2. (1 point) Complete the batch sampling code in the train function by specifying a batch of examples. You should make use the lists `train_indices` and `val_indices`.

3. (1 point) Train the model for a few epochs, and observe the training/validation loss and training/validation accuracy plots. Include these plots within the PDF you hand in.

   You should observe that the validation accuracy is low and stagnates after a few epochs. Next we will go through a rudimentary way of adjusting the hyperparameters of the model which we created.

4. (1 point) In one sentence, define the meaning of a "hyperparameter". Explain in a short paragraph why it is important not to evaluate the accuracy on the test set until all hyperparameters have been tuned.

5. (2 points) Select 4 hyperparameters associated with your network, one of the hyperparameter must involve your CNN architecture, and come up with two different sets of hyperparameters.

   For example, suppose my set of hyperparameters are defined as (yours might be different) $H = \{$batch size, learning rate, number of outputs of conv layer 1, number of conv layers$\}$, where conv layer is defined as a composition between filter, activation and pooling, then two sets of hyperparameters may be,

   $$H_1 = \{32,\ 0.001,\ 16,\ 2\} \quad H_2 = \{64,\ 0.0001,\ 32,\ 3\}$$

   The hyperparameters that you tune does not need to be specified as a numerical value. For instance, you can specify the optimizer you are using to train the network, or the activation function you are using. You may wish to consult: `https://objax.readthedocs.io/en/latest/notebooks/Custom_Networks.html`

6. (3 points) Create two additional networks $M_1, M_2$, each with the set of hyperparameter $H_1$, $H_2$ that you have selected above. Train each model. Report the best validation accuracy as well as the corresponding epoch for which this occurs for the `Base Model` and your two additional models. For example,

   `Base model`: 20% at epoch 22 $\quad M_1$: 30% at epoch 18 $\quad M_2$: 50% at epoch 24

   Which model performs the best in terms of validation accuracy? These new models do not need to outperform the base model, however, if you are unsatisfied with the validation accuracy, adjust your hyperparameters in the previous part and train until you are satisfied. Don't forget to report your hyperparameters.

7. (2 points) Based on your answer, which model should you pick as your final model and why? Then evaluate your model on the test set and report final test accuracy.

**Part 3. Trying Out a New Dataset.** To solidify your knowledge of training and tuning of a CNN, you will run another set of experiments on a new dataset. It should closely follow what you have done in Part 2. You will develop your own network to classify the data included in the dataset you picked. You may re-use any part of the starter code (e.g., for dataloading).

Pick a dataset of your own choosing from the following link (under "Image classification"): `https://www.tensorflow.org/datasets/catalog/overview`

1. (1 point) Import and partition your data.
2. (1 point) Create a base model to start out with
3. (3 points) Pick several hyperparameters you would like to tune and train a model until its validation accuracy is 5-10% better than the base model. Provide a succinct discussion on your design procedure: which hyperparameters you tuned, what is the new validation accuracy.
4. (1 point) Select your final model and report test accuracy.

**Part 4. Open-Ended Exploration.** In this final part, you will explore one of several questions introducing ways to improve your classification performance through tuning. You may build your experiment on top of either Part 2 or Part 3. Pick **ONE** of the questions below, perform a small set of **NEW** experiments that address that question, and provide a discussion.

1. **Additional hyperparameter tuning** Come up with one or several hyperparameters that you have not tried in the previous parts (e.g., explore arguments of the method `objax.nn.Cov2d`), tune them on the validation set until you see (at-least) 5%-10% increase in the validation accuracy as compared to a base model. Discuss whether it performs well on the test set as compared to your base model.
2. **How do hyperparameters interact?** Can you demonstrate for at-least two hyperparameters, where each independently increases the performance, but does not increase the performance or produce worse performance when used together? Alternatively, can you demonstrate for at-least two hyperparameters, where each independently increases the performance, and also increase the performance when used together? *You may interpret "performance" as performance on the validation set.* Report the final performance on the test set.
3. **How do optimizers compare?** Try out at least two optimizers (other than the SGD routine you have implemented in Part 2), clearly state their optimization routine and the rationale behind their implementations, and all hyperparameters associated with each optimizer. Implement them in Objax, then compare their performances on the validation set after some tuning. (You may find `objax.optimizer` useful) Report the final performance on the test set.
4. **Is it better just to grab an off-of-the-shelf model?** Import at-least one pre-trained model, explain its architecture, adapt the model(s) to your application and compare the performance of the model(s) to your own CNN model. (You may find `objax.zoo` useful. Some well-known models include VGG19 and Wide Residual Network.) Report the final performance on the test set.

Your grade will be allocated on the quality of the discussion, which involves the following components:

1. (4 points) A succinct discussion of your problem that involve the following: What was the question? Which hyperparameter(s) did you tune? How many models did you try and what are their performances in terms of validation accuracy? How did you choose your final model?

   Your short discussion should be written in a way that can be understood by a fellow classmate. This means you should define technical terms used in your discussion when relevant, and possibly introduce equations to explain concepts clearly. You do not need to tune hyperparameter(s) extensively and only need to include the results associated with your final tuned value. You may wish to include table and figures to support your arguments.

2. (1 point) Select your final model and report its test accuracy.

Make sure to clearly indicate which of the questions above you are trying to answer. You are only expected to answer one of the questions, but multiple experiments are welcome!

*
* *