CSCI 495/595

Final Exam

Instructions: This exam will require some research to answer a few questions. There are some questions for which you have to show your code implementation, e.g., show the snapshot of program run in this file. You are still required to turn in a Python based notebook of your code separately with this exam copy. Make sure there is a single python notebook file that contains separate code sections for each coding question.

1. What are regularization techniques for DL framework? Discuss dropout technique for deep learning (3-5 sentences). **(3 points)**

   Regularization techniques are used to reduce overfitting and improve model performance by increasing its ability to generalize new data. Some examples include early stopping (stopping training when the validation score reaches a minimum), max-norm (constrains the magnitude of the weight vectors), and ℓ1 / ℓ2 regularization (lasso and ridge regression).

   Dropout is another important regularization technique. In each training step, each neuron has a chance of being temporarily disabled and not considered during the forward/backward passes. This encourages each neuron to be more useful on its own and prevents the model from relying too much on a single neuron or path.

2. Discuss cross-validation for deep learning algorithms (3-5 sentences). **(2 points)**

   It is an evaluation technique that randomly splits the training set into k non-overlapping folds. Then it will train and evaluate the model's performance k times (one for each fold). In each iteration, one fold will become the validation set and the remaining folds will form the training set. Each fold will be used once as the validation set. This is a useful technique for assessing model performance with unseen data that doesn't require the use of the test set.

3. Consider the following DL network configuration (discussed for image classification of MNIST dataset). Your task is to construct a DL network with the following specifications. Attach the snapshot of program run in this exam file (you don't need to provide snapshot of your code here; that has to be submitted in a separate python notebook file)- **(7 points)**

Configuration	Hidden activation	Hidden Initializer	Output activation	Output initializer	Loss Function	Optimizer	Batch Size
1	tanh	Uniform 0.1	Sigmoid	Uniform 0.1	MSE	SGD Lr = 0.01	1

CE: Cross Entropy
MSE: Mean Squared error
SGD: stochastic gradient descent

Configuration details:

1. The network takes input image of 28 x 28 pixels, one hidden layer with 50 neurons, and the output layer with 10 outputs. It should initialize other DL parameters as shown in the configuration above.
2. The network code should have **10 EPOCHS** with a **batch size of 2.**
3. The network code should standardize the data.
4. The network creates hidden and output layers for the input data (MNIST dataset).
5. The network performs a Batch Normalization after the activation function in the hidden layer.
6. The network should use 'accuracy' for the metrics to output evaluation results.

Things to turn in:

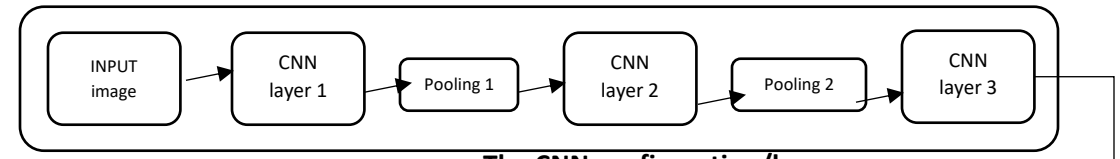
a) Snapshot of model run (i.e., snapshot when you would call a model.fit() function)

```
Epoch 1/10
30000/30000 - 51s - 2ms/step - accuracy: 0.5709 - loss: 0.0821 - val_accuracy: 0.8273 - val_loss: 0.0398
Epoch 2/10
30000/30000 - 51s - 2ms/step - accuracy: 0.7099 - loss: 0.0472 - val_accuracy: 0.8526 - val_loss: 0.0382
Epoch 3/10
30000/30000 - 51s - 2ms/step - accuracy: 0.7617 - loss: 0.0387 - val_accuracy: 0.8648 - val_loss: 0.0384
Epoch 4/10
30000/30000 - 51s - 2ms/step - accuracy: 0.7941 - loss: 0.0335 - val_accuracy: 0.8720 - val_loss: 0.0370
Epoch 5/10
30000/30000 - 51s - 2ms/step - accuracy: 0.8132 - loss: 0.0300 - val_accuracy: 0.8759 - val_loss: 0.0368
Epoch 6/10
30000/30000 - 52s - 2ms/step - accuracy: 0.8267 - loss: 0.0276 - val_accuracy: 0.8829 - val_loss: 0.0348
Epoch 7/10
30000/30000 - 50s - 2ms/step - accuracy: 0.8365 - loss: 0.0258 - val_accuracy: 0.8797 - val_loss: 0.0350
Epoch 8/10
30000/30000 - 50s - 2ms/step - accuracy: 0.8441 - loss: 0.0245 - val_accuracy: 0.8820 - val_loss: 0.0332
Epoch 9/10
30000/30000 - 50s - 2ms/step - accuracy: 0.8503 - loss: 0.0233 - val_accuracy: 0.8835 - val_loss: 0.0328
Epoch 10/10
30000/30000 - 50s - 2ms/step - accuracy: 0.8553 - loss: 0.0224 - val_accuracy: 0.8859 - val_loss: 0.0316
```

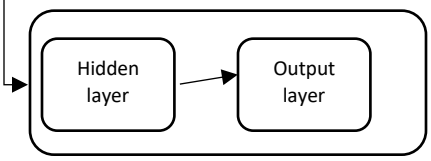
4. Consider the following CNN network configuration (discussed for image classification of CIFAR-10 dataset). Your task is to construct a CNN network with the following specifications. Attach

the snapshot of program run in this exam file (you don't need to provide snapshot of your code here; that has to be submitted in a separate python notebook file)- **(8 points)**

The CNN and DL layers for this question:



The CNN configuration/layer



The DL layers

Configurations (The CNN Layer):

a. The CNN part has 3 layers with 64 neurons in each layer and each followed by a pooling technique). Use MaxPooling technique with a size of 2 and a stride of size 2. (Research on how to implement Max pooling in Keras)
b. The CNN layer-1 should use kernel size of 5 and a stride size of 2. The CNN layer-2 and layer-3 should have a kernel size of 3 and a stride size of 2.
c. After the pooling layer-1, add a dropout of 20%. Similarly, add a dropout of 20% after the pooling layer-2. (Research on how to implement dropout in Keras)
d. For the CNN layers, use 'relu' as activation function, keep the padding parameter as 'same', kernel_initializer as 'he_normal', and bias initializer to be 'zeros'.

Configurations (The DL Layer):

a. The hidden layer should have 1024 neurons, activation function to be 'relu', kernel initializer to be 'he_normal'.
b. The output layer to have 10 neurons, activation function to be 'softmax', kernel initializer to be 'glorot uniform'. The bias initializer can be 'zeros'.
c. The loss function should be categorical cross entropy, optimizer should be 'adam', the evaluation metrics should be 'accuracy'.

Other details:

7. The network code should **have 128 EPOCHS** with a **batch size of 32.**
8. The network code should standardize the data.
9. The network creates the CNN and DL layers for the input data (CIFAR-10 dataset).

Things to turn in:

b) Snapshot of model run (i.e., snapshot when you would call a model.fit() function)

```
Epoch 111/128
391/391 - 2s - 4ms/step - accuracy: 0.8033 - loss: 0.5433 - val_accuracy: 0.7162 - val_loss: 0.9344
Epoch 112/128
391/391 - 3s - 7ms/step - accuracy: 0.8032 - loss: 0.5454 - val_accuracy: 0.7084 - val_loss: 0.9426
Epoch 113/128
391/391 - 2s - 4ms/step - accuracy: 0.8029 - loss: 0.5504 - val_accuracy: 0.7133 - val_loss: 0.9214
Epoch 114/128
391/391 - 2s - 4ms/step - accuracy: 0.8051 - loss: 0.5404 - val_accuracy: 0.7128 - val_loss: 0.9317
Epoch 115/128
391/391 - 3s - 7ms/step - accuracy: 0.8042 - loss: 0.5428 - val_accuracy: 0.7145 - val_loss: 0.9307
Epoch 116/128
391/391 - 3s - 6ms/step - accuracy: 0.8053 - loss: 0.5419 - val_accuracy: 0.7095 - val_loss: 0.9266
Epoch 117/128
391/391 - 2s - 4ms/step - accuracy: 0.8045 - loss: 0.5418 - val_accuracy: 0.7177 - val_loss: 0.9142
Epoch 118/128
391/391 - 3s - 6ms/step - accuracy: 0.8070 - loss: 0.5414 - val_accuracy: 0.7072 - val_loss: 0.9354
Epoch 119/128
391/391 - 2s - 6ms/step - accuracy: 0.8061 - loss: 0.5444 - val_accuracy: 0.7149 - val_loss: 0.9231
Epoch 120/128
391/391 - 2s - 4ms/step - accuracy: 0.8064 - loss: 0.5404 - val_accuracy: 0.7146 - val_loss: 0.9308
Epoch 121/128
391/391 - 2s - 6ms/step - accuracy: 0.8096 - loss: 0.5332 - val_accuracy: 0.7122 - val_loss: 0.9195
Epoch 122/128
391/391 - 3s - 7ms/step - accuracy: 0.8085 - loss: 0.5341 - val_accuracy: 0.7110 - val_loss: 0.9356
Epoch 123/128
391/391 - 2s - 4ms/step - accuracy: 0.8074 - loss: 0.5381 - val_accuracy: 0.7092 - val_loss: 0.9489
Epoch 124/128
391/391 - 2s - 6ms/step - accuracy: 0.8043 - loss: 0.5401 - val_accuracy: 0.7116 - val_loss: 0.9364
Epoch 125/128
391/391 - 3s - 7ms/step - accuracy: 0.8097 - loss: 0.5370 - val_accuracy: 0.7153 - val_loss: 0.9357
Epoch 126/128
391/391 - 2s - 4ms/step - accuracy: 0.8121 - loss: 0.5255 - val_accuracy: 0.7139 - val_loss: 0.9317
Epoch 127/128
391/391 - 2s - 6ms/step - accuracy: 0.8086 - loss: 0.5282 - val_accuracy: 0.7102 - val_loss: 0.9503
Epoch 128/128
391/391 - 2s - 5ms/step - accuracy: 0.8080 - loss: 0.5315 - val_accuracy: 0.7076 - val_loss: 0.9386
```

NOTE: You should keep all your code to a separate python notebook file for questions 3 and 4. However, only include the snapshot of model run in this file. Submit both python notebook and the exam in D2L before the due date.