PMDS603P Deep Learning Lab Experiment 4

August 2025

1 Work to do today

Note: Make a single PDF file of the work you are doing in a Jupyter notebook. Upload with the proper format. Please mention your name and roll no properly with the Experiment number on the first page of your submission.

Question1. cifar10 dataset is also an inbuilt dataset which contains 10 classes of images, mainly, 0-airplane, 1-automobile, 2-bird, 3-cat, 4-deer, 5-dog, 6-frog, 7-horse, 8-ship, 9-truck.

Load the inbuilt dataset cifar10 as you did in last lab by replacing mnist.load_datae() as cifar10.load_data(). First, try to import it from keras.datasets as you did for mnist.

Now, identify the size of the images you have first of all. You can now see 32*32*3 images that is 32*32 pixel images with 3 channels that give the RGB values since we have a color image.

Try to print the shape of each image and see. you will see it's stored like 32*32*3 arrays. Now, try to visualize certain images using appropriate functions.

Check the size of x_train and x_test and reshape them into one-dimensional arrays as done in the case of mnist dataset.

Do necessary pre-processing and split the data into training, validation, and testing sets.

Create a new model using a sequential class with appropriate hidden layers and output layer neurons. Choose appropriate activation functions like sigmoid and relu, etc. And also an appropriate one in the output layer.

Choose the error function appropriately. Include early stopping technique in your model and run the model for 500 epochs. Try to come up with a better model with decent accuracy.

The choice we have taken in the model here may not be the appropriate one. But you can see the accuracy you are able to come up with without having overfitting happen there.

Question 2. Next **from keras.regularizers import l2**

Now we will try to include L2 regularization in out modeling part for performing regularization. You can add this weight decay in any of your hidden layers like.

```
model.add(Dense(256, activation='relu', kernel_regularizer=l2(0.001)))
```

The parameter 0.001 is an arbitrary choice for the regularization parameter α that we saw in the class.

Now try to fit your same model as above with this change and check for any improvements.

Question 3: Now, let's see how we can proceed to do perform some hyperparameter tuning and find out the appropriate parameter value. The following part is done for a very simple model with one hidden layer and an output layer. The number of neurons and the dropout parameter is being tuned to find appropriate ones.

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.utils import to_categorical
from keras.optimizers import SGD
import keras_tuner as kt

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(-1, 28 * 28).astype('float32') / 255.0
x_test = x_test.reshape(-1, 28 * 28).astype('float32') / 255.0

print(x_train.shape)

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)


def build_model(hp):
    model = Sequential()
    model.add(Flatten(input_shape=(28*28,)))

    units = hp.Int('units', min_value=64, max_value=512, step=64)
    model.add(Dense(units, activation='relu'))

    dropout_rate = hp.Float('dropout', min_value=0.0, max_value=0.5, step=0.1)
    model.add(Dropout(dropout_rate))

    model.add(Dense(10, activation='softmax'))

    model.compile(
        optimizer=SGD(),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

What you see above is that we have loaded and pre-processed the data and created a model that would be re-used for the tuning part.

Next, we will set up the keras_tuner that can be used for the tuning part using tuner.search() So our objective is to maximize val_accuracy. We are using randomsearch with a maximum

```
tuner = kt.RandomSearch(
    build_model,
    objective='val_accuracy',
    max_trials=10,
    executions_per_trial=1,
    directory='mnist_tuning',
    project_name='dense_dropout_tune'
)

tuner.search(x_train, y_train,
             epochs=10,
             validation_split=0.2,
             batch_size=128,
             callbacks=[keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)])

best_model = tuner.get_best_models(num_models=1)[0]

test_loss, test_acc = best_model.evaluate(x_test, y_test)
print("Test accuracy:", test_acc)

best_hps = tuner.get_best_hyperparameters(1)[0]
print("Best units:", best_hps.get('units'))
print("Best dropout:", best_hps.get('dropout'))
```

10 combinations of our hyperparameters with each exceution_per_trail as 1. A directory is created on the disk with the name mnist_tuning which stores logs and information.

Now using tuner.search method, we can try to get the best model. Here I have only given 10 epochs. you can increase as well.

Now if you want to re-tune your model by adding few more layers etc then you have to first delete the already stored information in the directory mnist_tuning. For that you can use this

```
import shutil
shutil.rmtree('mnist_tuning/dense_dropout_tune', ignore_errors=True)
```

Then again you can re-run.

So your work is to include L2 regularization and tune the above model to get better accuracy with mnist.

Then later go back to the cifar10 dataset problem and come up with your best model.