PMDS603P Deep Learning Lab Experiment 8

September 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.

Today we will look at how we can define a simple CNN model to fit CIFAR 10 dataset and prepare a classification model for the same. Here we have a 10-class classification problem. Then, use the known trained models, such as VGG16, ResNet 50, to predict the same by using its pre-trained weights from IMAGENET. This concept or approach is regarded as transfer learning. Note that our image dimension is $32 \times 32 \times 3$.

Question 1: First try to fit a basic CNN model to accomplish the same task. Import necessary modules and classes first. Load the CIFAR 10 dataset and normalize the data and reshape. Now you can check the shape of xtrain and xtest etc which you are loading. we need to provide $32 \times 32 \times 3$ images to the the CNN. Next compile the model and fit the model and check the performance. Also include early stopping in your model. Since we have a classification problem we have to use the softmax activation in the final layer.

Question 2: Next we will see how you can define a new model incorporating the pre-trained model VGG16 in your model as its part. For that first you have to import the respective model details along with necessary modules and classes.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Resizing, Dropout
from tensorflow.keras.applications import VGG16
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from keras.layers import GlobalAveragePooling2D
```

Then load the CIFAR10 dataset. And then do necessary preprocessing and define the base_model as VGG16 as given below.

```
base_model = VGG16(include_top=False, weights="imagenet", input_shape=(224,224,3))
for layer in base_model.layers:
    layer.trainable = False
```

Here we remove the final fully connected layers and only use only until the convolution layers in the model and the same Imagenet competition weights are used. So we are not training the weights in those layers. That is why its given as those layers as non trainable.

Next we can define the model with the base model as the VGG16 model and define the architecture of the model. The input size for VGG16 model is $224 \times 224 \times 3$, so we add a resizing layer which takes care of the resizing initially that is from $32 \times 32 \times 3$ to $224 \times 224 \times 3$. Then we have our base model and then further layers. Finally two dense layers are also added

```
# Build model
model = Sequential()
model.add(Resizing(224, 224, input_shape=(32,32,3)))  # resize CIFAR-10 images
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))

model.summary()
model.compile(
    loss="categorical_crossentropy",
    optimizer="adam",
    metrics=["accuracy"]
)
```

with dropout regularization.

Now as in the previous case employ early stopping and fit the model to see whether you get better results.

Question 3: Try to use the pre-trained model ResNet50 and design a new model in similar way and report the results obtained.