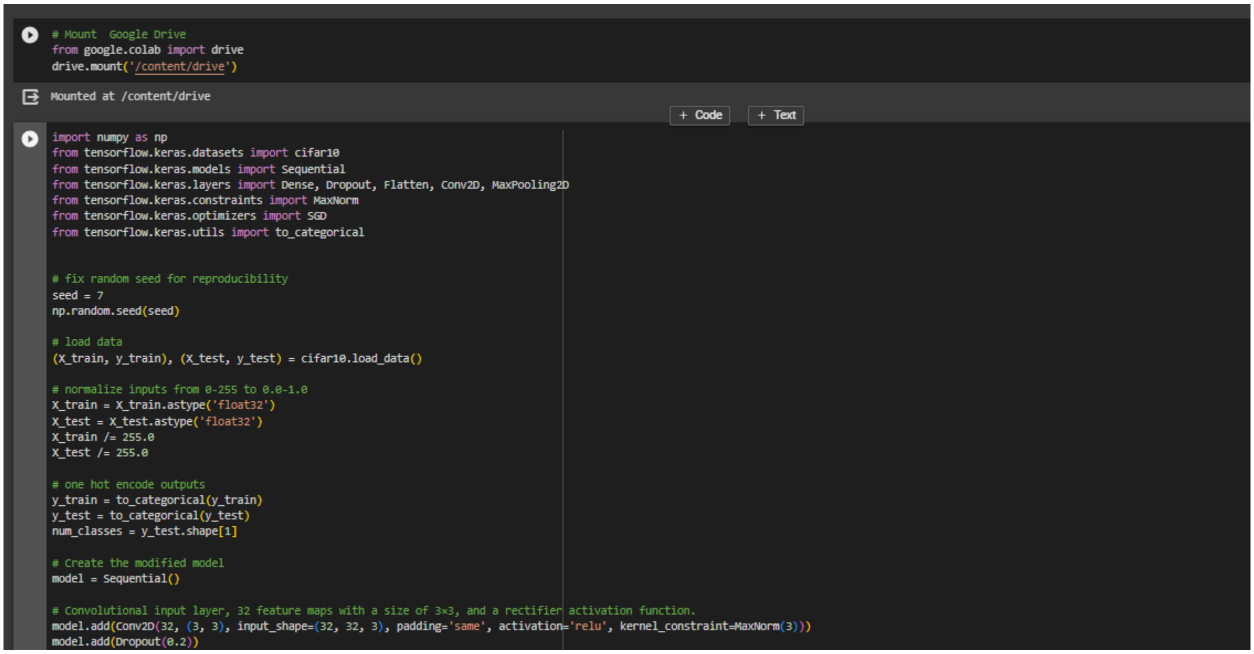
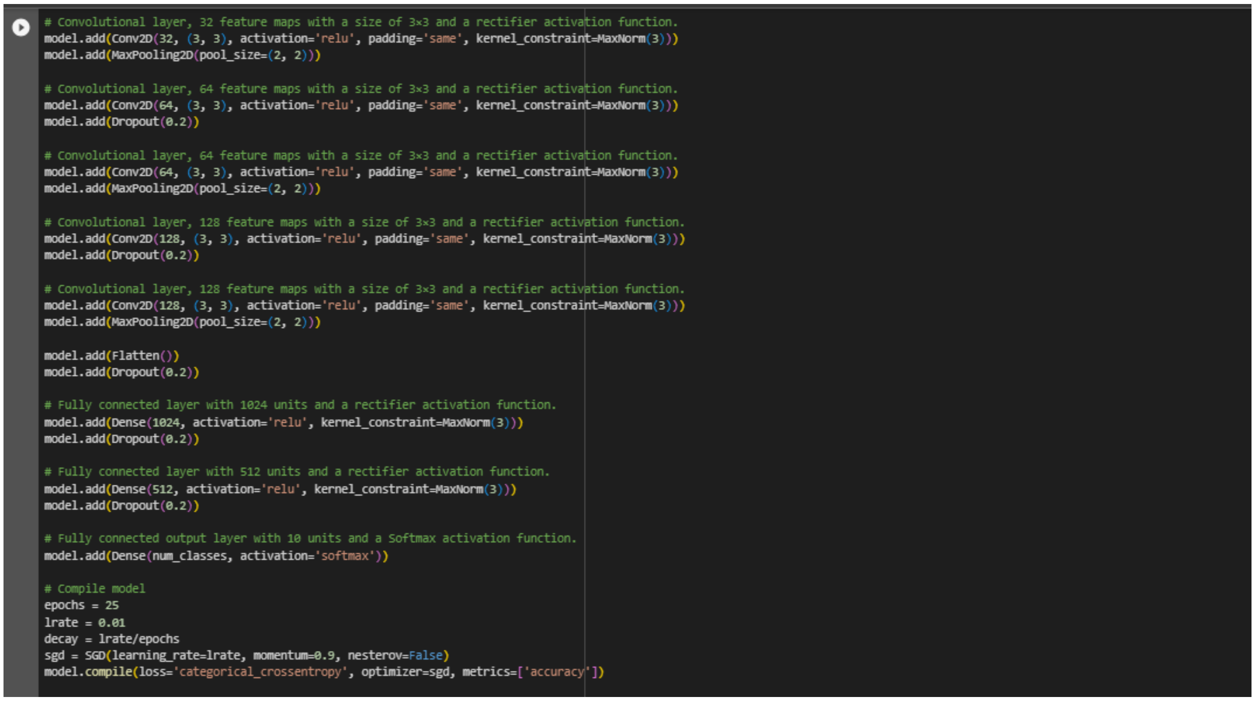
ICP 7 REPORT



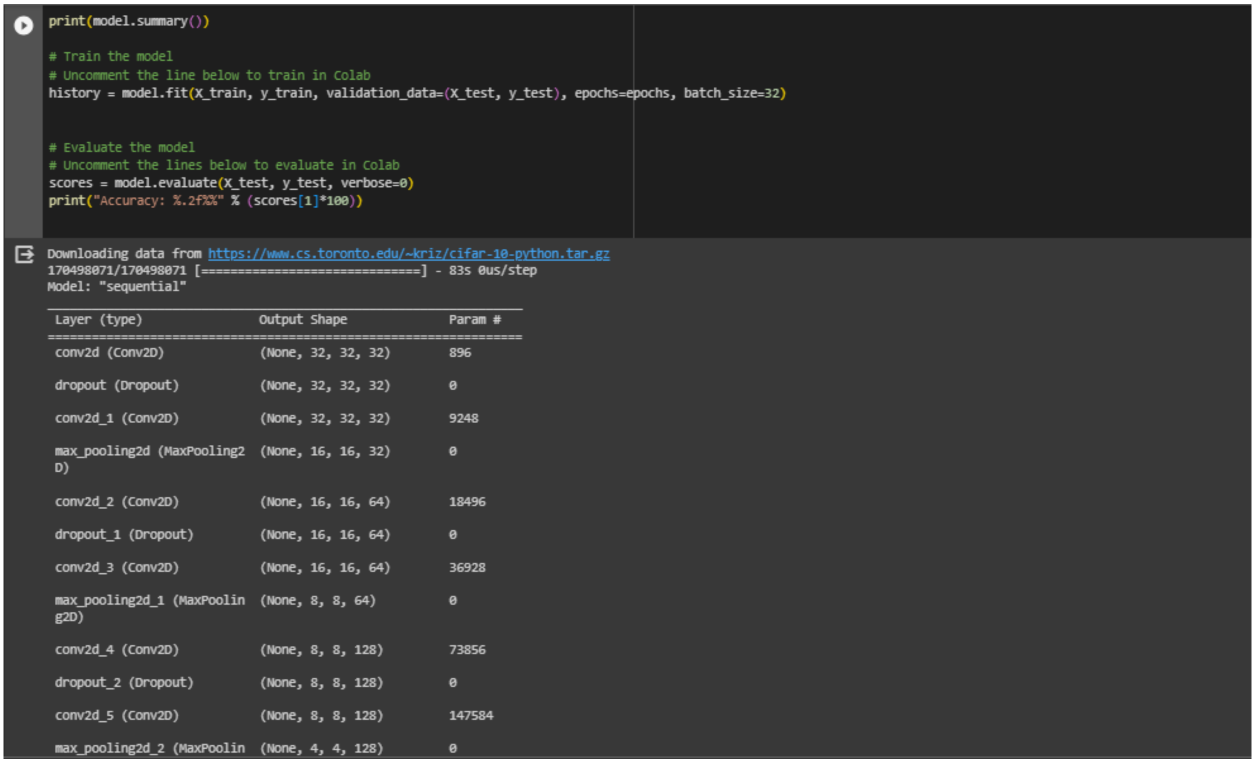
```
# Mount  Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.constraints import MaxNorm
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical


# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255.0
X_test /= 255.0

# one hot encode outputs
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]

# Create the modified model
model = Sequential()

# Convolutional input layer, 32 feature maps with a size of 3×3, and a rectifier activation function.
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
```

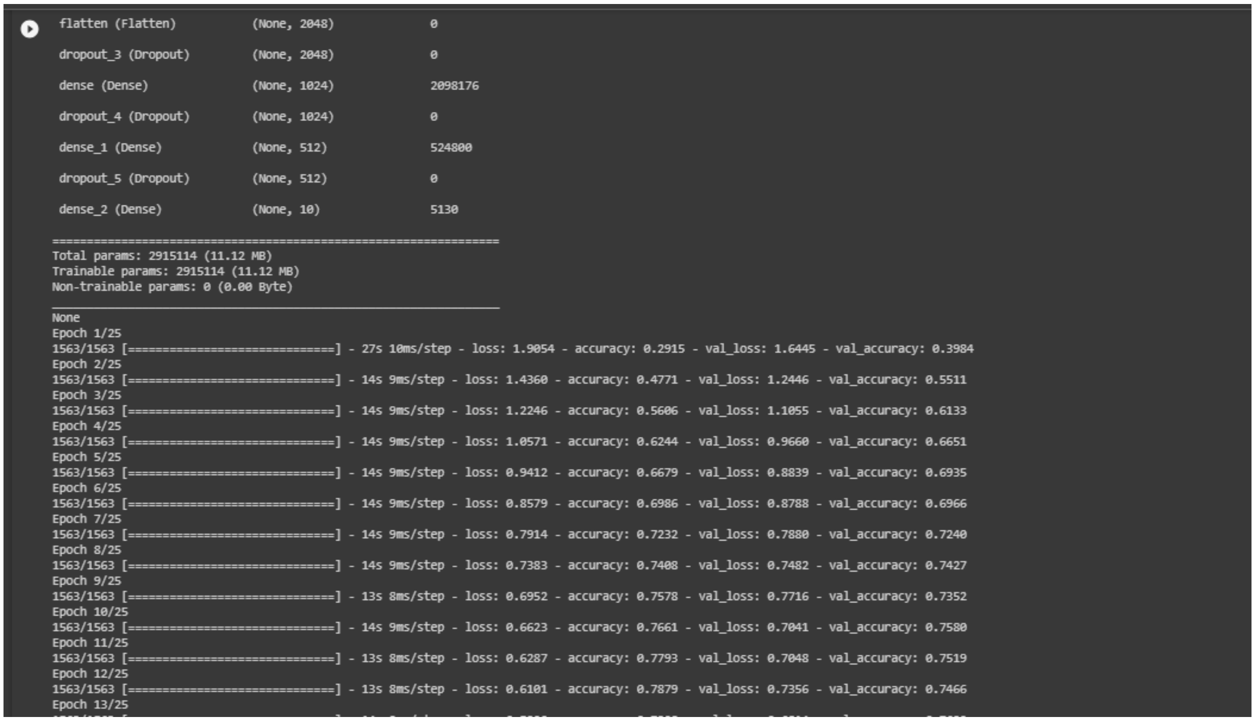


```
# Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))

# Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))

# Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dropout(0.2))

# Fully connected layer with 1024 units and a rectifier activation function.
model.add(Dense(1024, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))

# Fully connected layer with 512 units and a rectifier activation function.
model.add(Dense(512, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))

# Fully connected output layer with 10 units and a Softmax activation function.
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(learning_rate=lrate, momentum=0.9, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
```

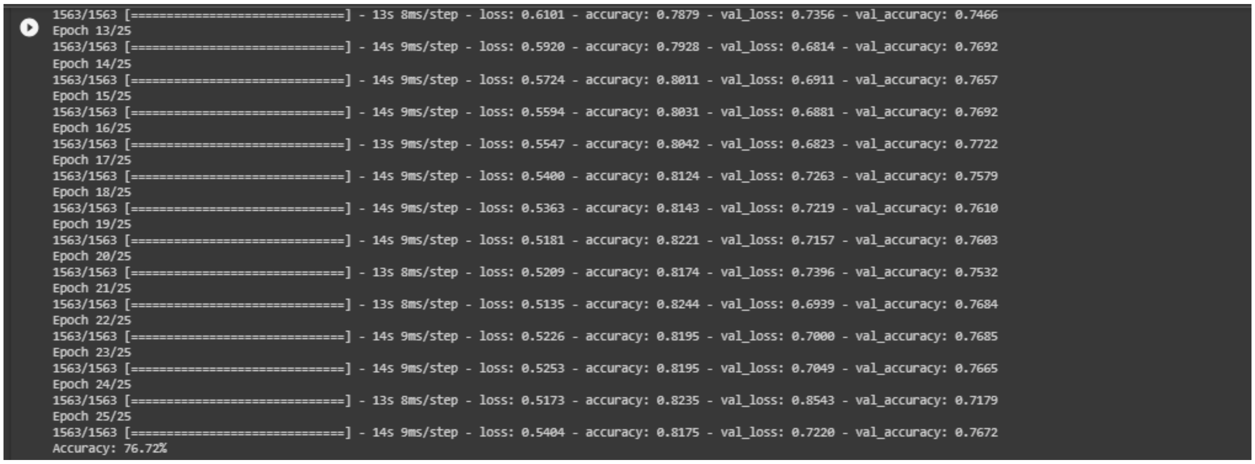


```
print(model.summary())

# Train the model
# Uncomment the line below to train in Colab
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)


# Evaluate the model
# Uncomment the lines below to evaluate in Colab
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

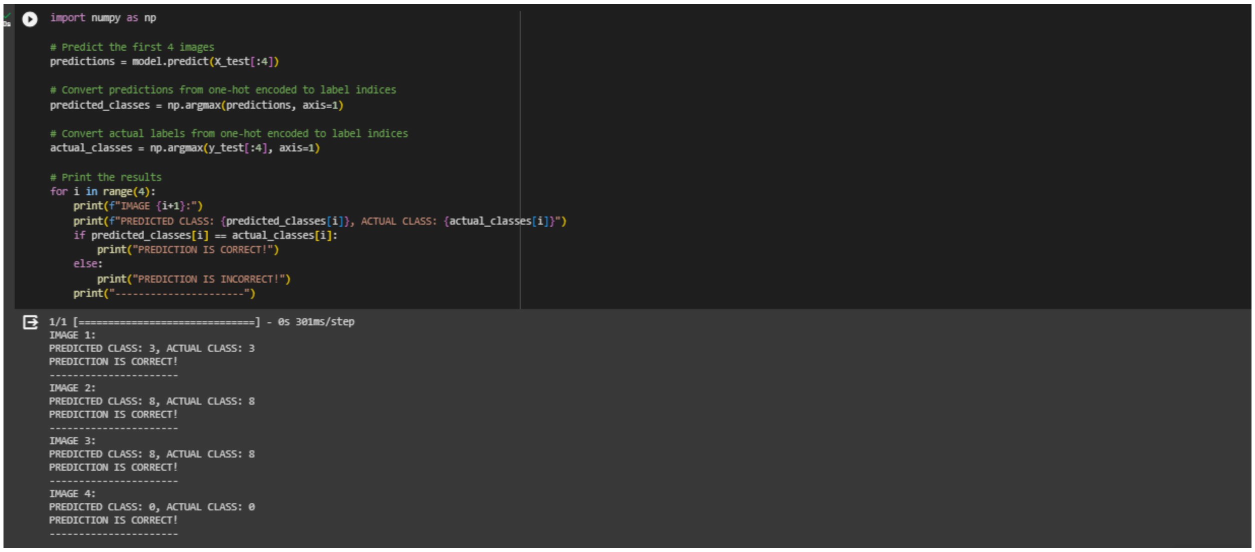
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 83s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 dropout (Dropout)           (None, 32, 32, 32)        0

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 dropout_1 (Dropout)         (None, 16, 16, 64)        0

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_1 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_4 (Conv2D)           (None, 8, 8, 128)         73856

 dropout_2 (Dropout)         (None, 8, 8, 128)         0

 conv2d_5 (Conv2D)           (None, 8, 8, 128)         147584

 max_pooling2d_2 (MaxPoolin  (None, 4, 4, 128)         0
```



```
 flatten (Flatten)           (None, 2048)              0

 dropout_3 (Dropout)         (None, 2048)              0

 dense (Dense)               (None, 1024)              2098176

 dropout_4 (Dropout)         (None, 1024)              0

 dense_1 (Dense)             (None, 512)               524800

 dropout_5 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 10)                5130

=================================================================
Total params: 2915114 (11.12 MB)
Trainable params: 2915114 (11.12 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
Epoch 1/25
1563/1563 [==============================] - 27s 10ms/step - loss: 1.9054 - accuracy: 0.2915 - val_loss: 1.6445 - val_accuracy: 0.3984
Epoch 2/25
1563/1563 [==============================] - 14s 9ms/step - loss: 1.4360 - accuracy: 0.4771 - val_loss: 1.2446 - val_accuracy: 0.5511
Epoch 3/25
1563/1563 [==============================] - 14s 9ms/step - loss: 1.2246 - accuracy: 0.5606 - val_loss: 1.1055 - val_accuracy: 0.6133
Epoch 4/25
1563/1563 [==============================] - 14s 9ms/step - loss: 1.0571 - accuracy: 0.6244 - val_loss: 0.9660 - val_accuracy: 0.6651
Epoch 5/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.9412 - accuracy: 0.6679 - val_loss: 0.8839 - val_accuracy: 0.6935
Epoch 6/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.8579 - accuracy: 0.6986 - val_loss: 0.8788 - val_accuracy: 0.6966
Epoch 7/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.7914 - accuracy: 0.7232 - val_loss: 0.7880 - val_accuracy: 0.7240
Epoch 8/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.7383 - accuracy: 0.7408 - val_loss: 0.7482 - val_accuracy: 0.7427
Epoch 9/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.6952 - accuracy: 0.7578 - val_loss: 0.7716 - val_accuracy: 0.7352
Epoch 10/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.6623 - accuracy: 0.7661 - val_loss: 0.7041 - val_accuracy: 0.7580
Epoch 11/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.6287 - accuracy: 0.7793 - val_loss: 0.7048 - val_accuracy: 0.7519
Epoch 12/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.6101 - accuracy: 0.7879 - val_loss: 0.7356 - val_accuracy: 0.7466
Epoch 13/25
```

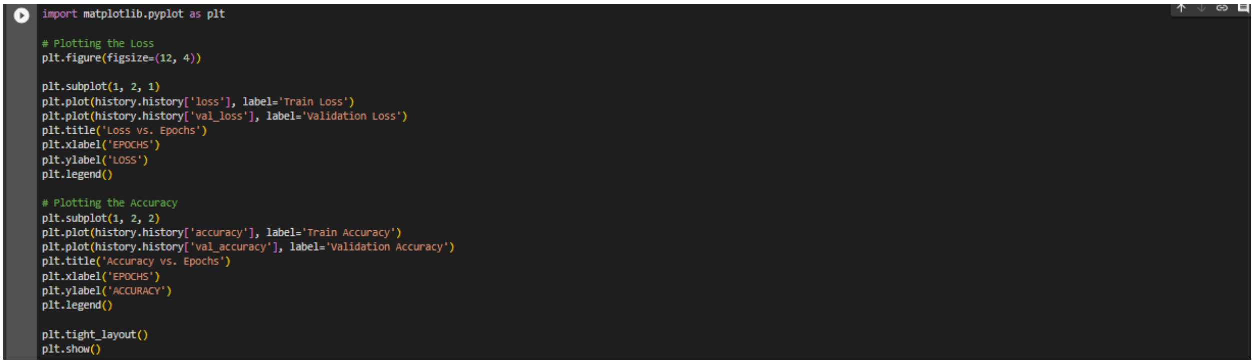


```
1563/1563 [==============================] - 13s 8ms/step - loss: 0.6101 - accuracy: 0.7879 - val_loss: 0.7356 - val_accuracy: 0.7466
Epoch 13/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5920 - accuracy: 0.7928 - val_loss: 0.6814 - val_accuracy: 0.7692
Epoch 14/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5724 - accuracy: 0.8011 - val_loss: 0.6911 - val_accuracy: 0.7657
Epoch 15/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5594 - accuracy: 0.8031 - val_loss: 0.6881 - val_accuracy: 0.7692
Epoch 16/25
1563/1563 [==============================] - 13s 9ms/step - loss: 0.5547 - accuracy: 0.8042 - val_loss: 0.6823 - val_accuracy: 0.7722
Epoch 17/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5400 - accuracy: 0.8124 - val_loss: 0.7263 - val_accuracy: 0.7579
Epoch 18/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5363 - accuracy: 0.8143 - val_loss: 0.7219 - val_accuracy: 0.7610
Epoch 19/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5181 - accuracy: 0.8221 - val_loss: 0.7157 - val_accuracy: 0.7603
Epoch 20/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5209 - accuracy: 0.8174 - val_loss: 0.7396 - val_accuracy: 0.7532
Epoch 21/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5135 - accuracy: 0.8244 - val_loss: 0.6939 - val_accuracy: 0.7684
Epoch 22/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5226 - accuracy: 0.8195 - val_loss: 0.7000 - val_accuracy: 0.7685
Epoch 23/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5253 - accuracy: 0.8195 - val_loss: 0.7049 - val_accuracy: 0.7665
Epoch 24/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5173 - accuracy: 0.8235 - val_loss: 0.8543 - val_accuracy: 0.7179
Epoch 25/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5404 - accuracy: 0.8175 - val_loss: 0.7220 - val_accuracy: 0.7672
Accuracy: 76.72%
```

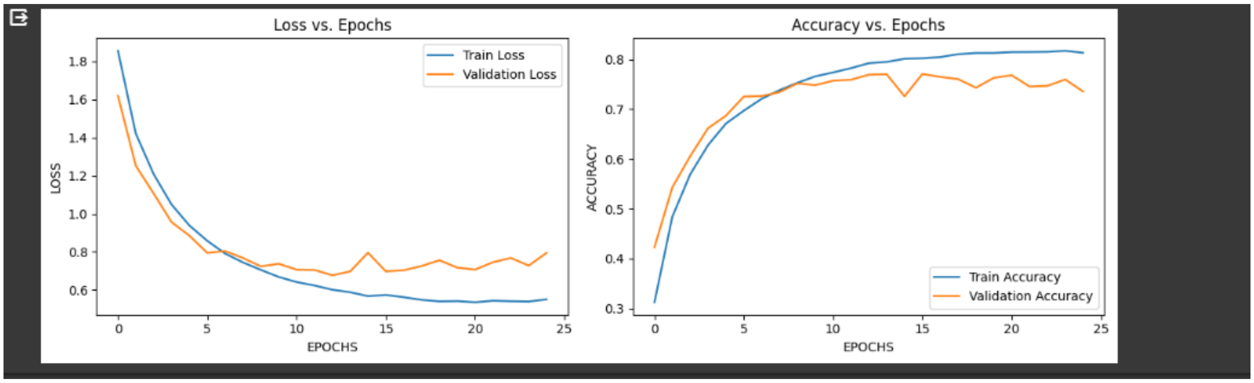


```python
import numpy as np

# Predict the first 4 images
predictions = model.predict(X_test[:4])

# Convert predictions from one-hot encoded to label indices
predicted_classes = np.argmax(predictions, axis=1)

# Convert actual labels from one-hot encoded to label indices
actual_classes = np.argmax(y_test[:4], axis=1)

# Print the results
for i in range(4):
    print(f"IMAGE {i+1}:")
    print(f"PREDICTED CLASS: {predicted_classes[i]}, ACTUAL CLASS: {actual_classes[i]}")
    if predicted_classes[i] == actual_classes[i]:
        print("PREDICTION IS CORRECT!")
    else:
        print("PREDICTION IS INCORRECT!")
    print("---------------------")
```

```
1/1 [==============================] - 0s 301ms/step
IMAGE 1:
PREDICTED CLASS: 3, ACTUAL CLASS: 3
PREDICTION IS CORRECT!
---------------------
IMAGE 2:
PREDICTED CLASS: 8, ACTUAL CLASS: 8
PREDICTION IS CORRECT!
---------------------
IMAGE 3:
PREDICTED CLASS: 8, ACTUAL CLASS: 8
PREDICTION IS CORRECT!
---------------------
IMAGE 4:
PREDICTED CLASS: 0, ACTUAL CLASS: 0
PREDICTION IS CORRECT!
---------------------
```



```python
import matplotlib.pyplot as plt

# Plotting the Loss
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss vs. Epochs')
plt.xlabel('EPOCHS')
plt.ylabel('LOSS')
plt.legend()

# Plotting the Accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy vs. Epochs')
plt.xlabel('EPOCHS')
plt.ylabel('ACCURACY')
plt.legend()

plt.tight_layout()
plt.show()
```



GITHUB Repo: https://github.com/sxk7912/Bigdata