Ex.No: 4 Roll.No: 210701182

# PYTHON PROGRAM TO VISUALIZE AND DESIGN CNN WITH TRANSFER LEARNING

Aim:

To visualize and design CNN with transfer learning in python.

# Procedure:

1. Import TensorFlow, CIFAR-10 dataset, VGG16 model, and necessary Keras layers for building the model.
2. Load the CIFAR-10 dataset, consisting of 60,000 images, and divide it into training and test sets.
3. Normalize the training and test images by scaling pixel values to the range [0, 1].
4. One-hot encode the training and test labels to convert them into categorical format.
5. Load the pre-trained VGG16 model with ImageNet weights, excluding the top fully connected layers.
6. Freeze the layers of the pre-trained VGG16 model to prevent them from being trained.
7. Initialize a Sequential model and add the pre-trained VGG16 as the base.
8. Add a Flatten layer followed by a Dense layer with 256 units and ReLU activation.
9. Add a ﬁnal Dense layer with 10 units and softmax activation for CIFAR-10 classiﬁcation.
10. Compile the model with Adam optimizer, train for 10 epochs, and evaluate it on test data to print the accuracy.

# Code:

import tensorﬂow as tf

from tensorﬂow.keras.datasets import cifar10 from tensorﬂow.keras.applications import VGG16 from tensorﬂow.keras.models import Sequential from tensorﬂow.keras.layers import Flatten, Dense from tensorﬂow.keras.utils import to\_categorical

# Load CIFAR-10 dataset

(itrain, ltrain), (itest, ltest) = cifar10.load\_data()

# Preprocess the data itrain = itrain / 255.0 itest = itest / 255.0

ltrain = to\_categorical(ltrain) ltest = to\_categorical(ltest)

# Load pre-trained VGG16 model (excluding the top fully-connected layers) basem = VGG16(weights='imagenet', include\_top=False, input\_shape=(32, 32, 3))

# Freeze the pre-trained layers for layer in basem.layers:

layer.trainable = False

# Create a new model on top semodel = Sequential() semodel.add(basem) semodel.add(Flatten())

semodel.add(Dense(256, activation='relu'))

semodel.add(Dense(10, activation='softmax')) # CIFAR-10 has 10 classes

# Compile the model

semodel.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

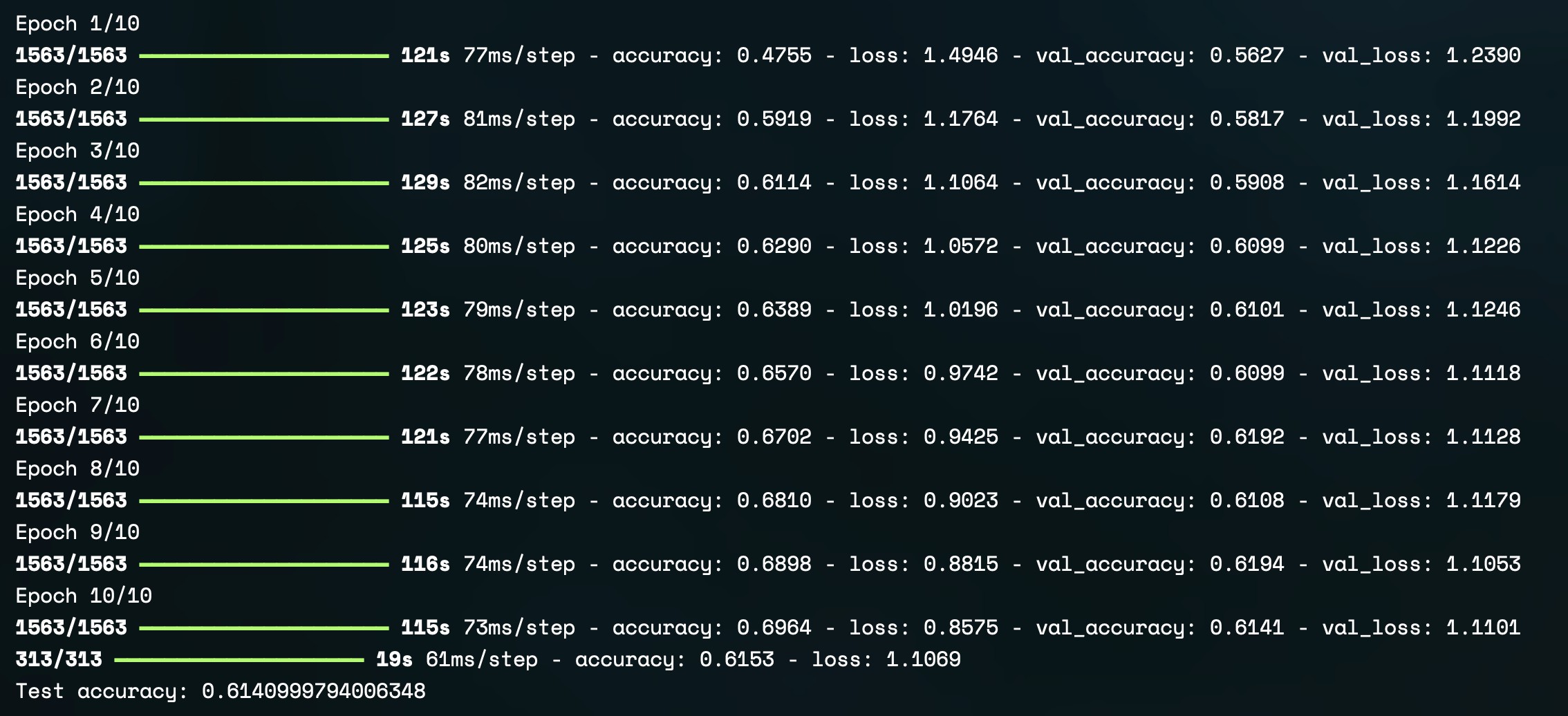
# Train the model

semodel.ﬁt(itrain, ltrain, epochs=10, batch\_size=32, validation\_data=(itest, ltest))

# Evaluate the model on test data

ltest, atest = semodel.evaluate(itest, ltest) print("Test accuracy:", atest)

# Output:



Result:

Thus, to visualize and design CNN with transfer learning has been completed successfully.