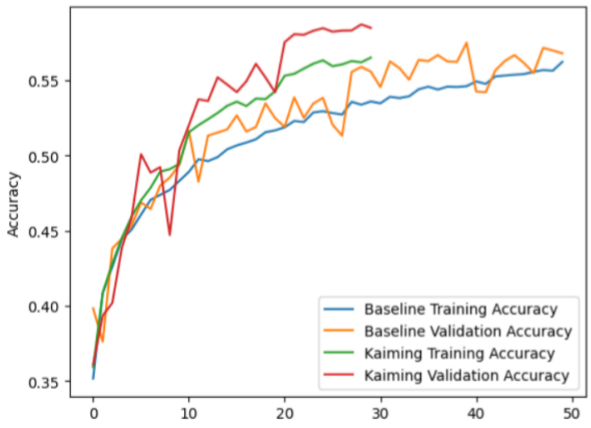
PROGRAM

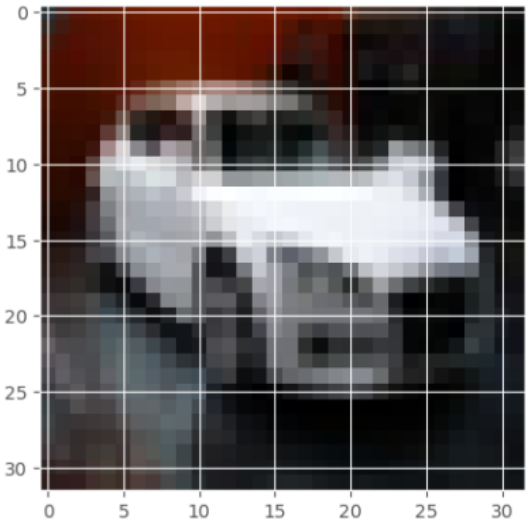
```
import tensorflow as tf
from tensorflow.keras import layers, models, initializers
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.callbacks import LearningRateScheduler,
EarlyStopping
import matplotlib.pyplot as plt
train_images, train_labels), (test_images, test_labels) =
cifar10.load_data()
train_images, test_images = train_images / 255.0, test_images
/ 255.0
train_labels = to_categorical(train_labels, 10)
test_labels = to_categorical(test_labels, 10)
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
)
datagen.fit(train_images)
model_kaiming = models.Sequential()
model_kaiming.add(layers.Flatten(input_shape=(32, 32, 3)))
model_kaiming.add(layers.Dense(512, activation='relu',
kernel_initializer='he_normal'))
model_kaiming.add(layers.BatchNormalization())
model_kaiming.add(layers.Dense(256, activation='relu',
kernel_initializer='he_normal'))
model_kaiming.add(layers.BatchNormalization())
model_kaiming.add(layers.Dense(128, activation='relu',
kernel_initializer='he_normal'))
model_kaiming.add(layers.BatchNormalization())
model_kaiming.add(layers.Dense(10, activation='softmax',
kernel_initializer='he_normal'))
model_kaiming.compile(optimizer='adam',loss='categorical_cross
entropy',
                      metrics=['accuracy'])

def lr_schedule(epoch):
    if epoch < 10:
        return 0.001
    elif epoch < 20:
        return 0.0005
    elif epoch < 30:
```

```
        return 0.0001
    else:
        return 0.00005
lr_scheduler = LearningRateScheduler(lr_schedule)
early_stopping = EarlyStopping(monitor='val_accuracy',
patience=10, restore_best_weights=True)

history_kaiming = model_kaiming.fit(datagen.flow(train_images,
train_labels,
batch_size=64),epochs=30,validation_data=(test_images,
test_labels),callbacks=[lr_scheduler, early_stopping])

test_loss_kaiming, test_acc_kaiming =
model_kaiming.evaluate(test_images, test_labels)
print(f'Test accuracy with Kaiming initialization:
{test_acc_kaiming * 100:.2f}%')

plt.plot(history.history['accuracy'], label='Baseline Training
Accuracy')
plt.plot(history.history['val_accuracy'], label='Baseline
Validation Accuracy')
plt.plot(history_kaiming.history['accuracy'], label='Kaiming
Training Accuracy')
plt.plot(history_kaiming.history['val_accuracy'],
label='Kaiming Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

import numpy as np
predictions = model.predict(test_images)
predicted_labels = np.argmax(predictions, axis=1)
plt.figure(figsize=(10, 5))
for i in range(5):
    plt.subplot(1, 5, i + 1)
    plt.imshow(test_images[i])
    plt.title(f"Actual:
{np.argmax(test_labels[i])}\nPredicted:
{predicted_labels[i]}")
    plt.axis('off')
plt.show()
```

OUTPUT

```
782/782 [==============================] - 36s 46ms/step - loss: 1.2259 - accuracy: 0.5651 - val_loss: 1.1717 - val_accuracy: 0.5849 - lr: 1.0000e-04
313/313 [==============================] - 1s 3ms/step - loss: 1.1717 - accuracy: 0.5849
Test accuracy with Kaiming initialization: 58.49%
```





```
313/313 [==============================] - 3s 10ms/step
Predicted Label: 1
```