

```
In [1]: import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers import SGD
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers.schedules import PiecewiseConstantDecay
import seaborn as sns
import time
```

```
In [2]: # Load CIFAR-10 dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# Normalize the images
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# Ensure labels are integers (no one-hot encoding)
y_train = y_train.astype('int')
y_test = y_test.astype('int')

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,
print(X_train.shape)
print(X_test.shape)

(40000, 32, 32, 3)
(10000, 32, 32, 3)
```

```
In [3]: # Training parameters
total_steps = 31250 # Total steps (50 epochs x 625 steps per epoch)
initial_lr = 0.001 # Starting Learning rate

# Define decay points and values
boundaries = [int(total_steps * 0.3), int(total_steps * 0.6), int(total_steps * 0.9)]
values = [initial_lr, 0.0005, 0.0001, 0.00005] # Learning rates for each interval

# Create Step Decay scheduler
step_decay_schedule = PiecewiseConstantDecay(boundaries=boundaries, values=values)

# Define optimizer
optimizer = SGD(learning_rate=step_decay_schedule)
```

```
In [4]: # Create model function
def create_model(optimizer):
    model = models.Sequential()
    model.add(layers.Flatten(input_shape=(32, 32, 3)))
    model.add(layers.Dense(128, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))
    model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metr
    return model
```

```
In [5]: from tensorflow.keras.callbacks import EarlyStopping



















# Train and evaluate model
start_time = time.time()
model = create_model(optimizer)
early_stop = EarlyStopping(monitor='val_accuracy', patience=5, restore_best_weights=True)

history = model.fit(X_train, y_train, epochs=50, batch_size=64, validation_data=(X_test, y_test))
end_time = time.time()

# Record training time
training_time = end_time - start_time
print(f"Training time: {training_time:.2f} seconds")
```

```
c:\Users\Omar Wessam\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras
\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an `Input(shape)` object
as the first layer in the model instead.
  super().__init__(**kwargs)
```

Epoch 1/50
625/625 ————— 2s 3ms/step - accuracy: 0.1709 - loss: 2.2454 - val_acc
uracy: 0.2557 - val_loss: 2.0807
Epoch 2/50
625/625 ————— 2s 2ms/step - accuracy: 0.2676 - loss: 2.0547 - val_acc
uracy: 0.2858 - val_loss: 1.9955
Epoch 3/50
625/625 ————— 1s 2ms/step - accuracy: 0.3021 - loss: 1.9791 - val_acc
uracy: 0.3139 - val_loss: 1.9438
Epoch 4/50
625/625 ————— 2s 2ms/step - accuracy: 0.3271 - loss: 1.9337 - val_acc
uracy: 0.3289 - val_loss: 1.9118
Epoch 5/50
625/625 ————— 1s 2ms/step - accuracy: 0.3382 - loss: 1.8995 - val_acc
uracy: 0.3383 - val_loss: 1.8841
Epoch 6/50
625/625 ————— 1s 2ms/step - accuracy: 0.3481 - loss: 1.8738 - val_acc
uracy: 0.3478 - val_loss: 1.8625
Epoch 7/50
625/625 ————— 1s 2ms/step - accuracy: 0.3578 - loss: 1.8510 - val_acc
uracy: 0.3546 - val_loss: 1.8447
Epoch 8/50
625/625 ————— 1s 2ms/step - accuracy: 0.3671 - loss: 1.8369 - val_acc
uracy: 0.3613 - val_loss: 1.8286
Epoch 9/50
625/625 ————— 1s 2ms/step - accuracy: 0.3708 - loss: 1.8230 - val_acc
uracy: 0.3615 - val_loss: 1.8190
Epoch 10/50
625/625 ————— 1s 2ms/step - accuracy: 0.3781 - loss: 1.8025 - val_acc
uracy: 0.3707 - val_loss: 1.8047
Epoch 11/50
625/625 ————— 1s 2ms/step - accuracy: 0.3827 - loss: 1.7886 - val_acc
uracy: 0.3694 - val_loss: 1.7949
Epoch 12/50
625/625 ————— 1s 2ms/step - accuracy: 0.3875 - loss: 1.7829 - val_acc
uracy: 0.3801 - val_loss: 1.7832
Epoch 13/50
625/625 ————— 1s 2ms/step - accuracy: 0.3901 - loss: 1.7686 - val_acc
uracy: 0.3818 - val_loss: 1.7748
Epoch 14/50
625/625 ————— 1s 2ms/step - accuracy: 0.3912 - loss: 1.7613 - val_acc
uracy: 0.3877 - val_loss: 1.7656
Epoch 15/50
625/625 ————— 1s 2ms/step - accuracy: 0.3909 - loss: 1.7637 - val_acc
uracy: 0.3870 - val_loss: 1.7614
Epoch 16/50
625/625 ————— 1s 2ms/step - accuracy: 0.4018 - loss: 1.7422 - val_acc
uracy: 0.3923 - val_loss: 1.7552
Epoch 17/50
625/625 ————— 1s 2ms/step - accuracy: 0.3967 - loss: 1.7446 - val_acc
uracy: 0.3932 - val_loss: 1.7499
Epoch 18/50
625/625 ————— 1s 2ms/step - accuracy: 0.4015 - loss: 1.7381 - val_acc
uracy: 0.3947 - val_loss: 1.7469
Epoch 19/50
625/625 ————— 1s 2ms/step - accuracy: 0.4026 - loss: 1.7323 - val_acc

uracy: 0.3939 - val_loss: 1.7445
Epoch 20/50
625/625  1s 2ms/step - accuracy: 0.4116 - loss: 1.7206 - val_acc
uracy: 0.3985 - val_loss: 1.7399
Epoch 21/50
625/625  1s 2ms/step - accuracy: 0.4092 - loss: 1.7252 - val_acc
uracy: 0.3984 - val_loss: 1.7361
Epoch 22/50
625/625  1s 2ms/step - accuracy: 0.4095 - loss: 1.7172 - val_acc
uracy: 0.4014 - val_loss: 1.7336
Epoch 23/50
625/625  1s 2ms/step - accuracy: 0.4108 - loss: 1.7201 - val_acc
uracy: 0.4029 - val_loss: 1.7308
Epoch 24/50
625/625  1s 2ms/step - accuracy: 0.4115 - loss: 1.7182 - val_acc
uracy: 0.4025 - val_loss: 1.7271
Epoch 25/50
625/625  1s 2ms/step - accuracy: 0.4151 - loss: 1.7109 - val_acc
uracy: 0.4049 - val_loss: 1.7268
Epoch 26/50
625/625  1s 2ms/step - accuracy: 0.4145 - loss: 1.7103 - val_acc
uracy: 0.4035 - val_loss: 1.7210
Epoch 27/50
625/625  1s 2ms/step - accuracy: 0.4173 - loss: 1.7000 - val_acc
uracy: 0.4051 - val_loss: 1.7187
Epoch 28/50
625/625  1s 2ms/step - accuracy: 0.4124 - loss: 1.7043 - val_acc
uracy: 0.4024 - val_loss: 1.7183
Epoch 29/50
625/625  1s 2ms/step - accuracy: 0.4191 - loss: 1.6971 - val_acc
uracy: 0.4068 - val_loss: 1.7132
Epoch 30/50
625/625  1s 2ms/step - accuracy: 0.4196 - loss: 1.6884 - val_acc
uracy: 0.4075 - val_loss: 1.7110
Epoch 31/50
625/625  1s 2ms/step - accuracy: 0.4148 - loss: 1.6957 - val_acc
uracy: 0.4099 - val_loss: 1.7089
Epoch 32/50
625/625  1s 2ms/step - accuracy: 0.4220 - loss: 1.6911 - val_acc
uracy: 0.4096 - val_loss: 1.7089
Epoch 33/50
625/625  1s 2ms/step - accuracy: 0.4199 - loss: 1.6932 - val_acc
uracy: 0.4098 - val_loss: 1.7079
Epoch 34/50
625/625  1s 2ms/step - accuracy: 0.4168 - loss: 1.7010 - val_acc
uracy: 0.4113 - val_loss: 1.7070
Epoch 35/50
625/625  1s 2ms/step - accuracy: 0.4193 - loss: 1.6898 - val_acc
uracy: 0.4103 - val_loss: 1.7070
Epoch 36/50
625/625  1s 2ms/step - accuracy: 0.4129 - loss: 1.6968 - val_acc
uracy: 0.4107 - val_loss: 1.7060
Epoch 37/50
625/625  1s 2ms/step - accuracy: 0.4190 - loss: 1.6883 - val_acc
uracy: 0.4120 - val_loss: 1.7059
Epoch 38/50

```

625/625 ————— 1s 2ms/step - accuracy: 0.4228 - loss: 1.6893 - val_acc
uracy: 0.4120 - val_loss: 1.7052
Epoch 39/50
625/625 ————— 1s 2ms/step - accuracy: 0.4203 - loss: 1.6871 - val_acc
uracy: 0.4122 - val_loss: 1.7045
Epoch 40/50
625/625 ————— 1s 2ms/step - accuracy: 0.4241 - loss: 1.6837 - val_acc
uracy: 0.4113 - val_loss: 1.7044
Epoch 41/50
625/625 ————— 1s 2ms/step - accuracy: 0.4223 - loss: 1.6896 - val_acc
uracy: 0.4130 - val_loss: 1.7038
Epoch 42/50
625/625 ————— 1s 2ms/step - accuracy: 0.4244 - loss: 1.6838 - val_acc
uracy: 0.4134 - val_loss: 1.7030
Epoch 43/50
625/625 ————— 1s 2ms/step - accuracy: 0.4180 - loss: 1.6943 - val_acc
uracy: 0.4133 - val_loss: 1.7027
Epoch 44/50
625/625 ————— 1s 2ms/step - accuracy: 0.4255 - loss: 1.6787 - val_acc
uracy: 0.4123 - val_loss: 1.7020
Epoch 45/50
625/625 ————— 1s 2ms/step - accuracy: 0.4219 - loss: 1.6884 - val_acc
uracy: 0.4124 - val_loss: 1.7015
Epoch 46/50
625/625 ————— 1s 2ms/step - accuracy: 0.4271 - loss: 1.6750 - val_acc
uracy: 0.4131 - val_loss: 1.7013
Epoch 47/50
625/625 ————— 1s 2ms/step - accuracy: 0.4269 - loss: 1.6789 - val_acc
uracy: 0.4129 - val_loss: 1.7010
Training time: 65.69 seconds

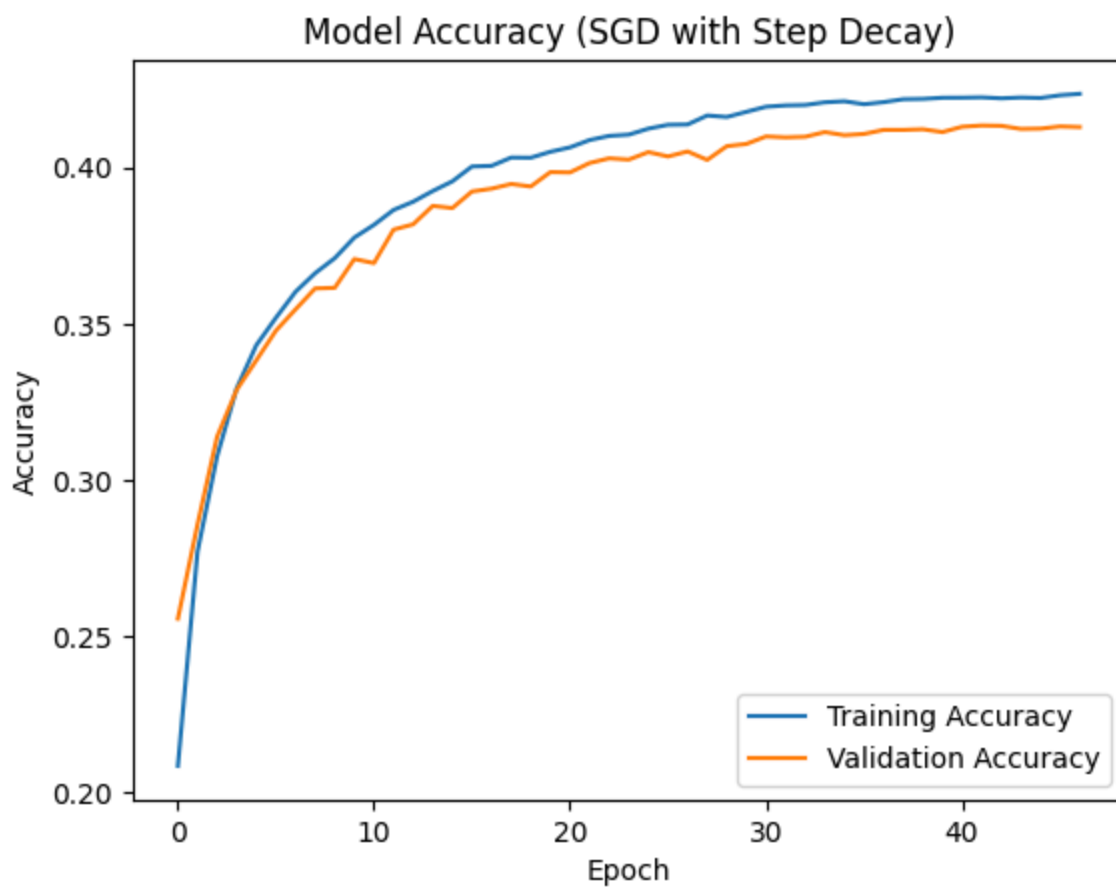
```

```

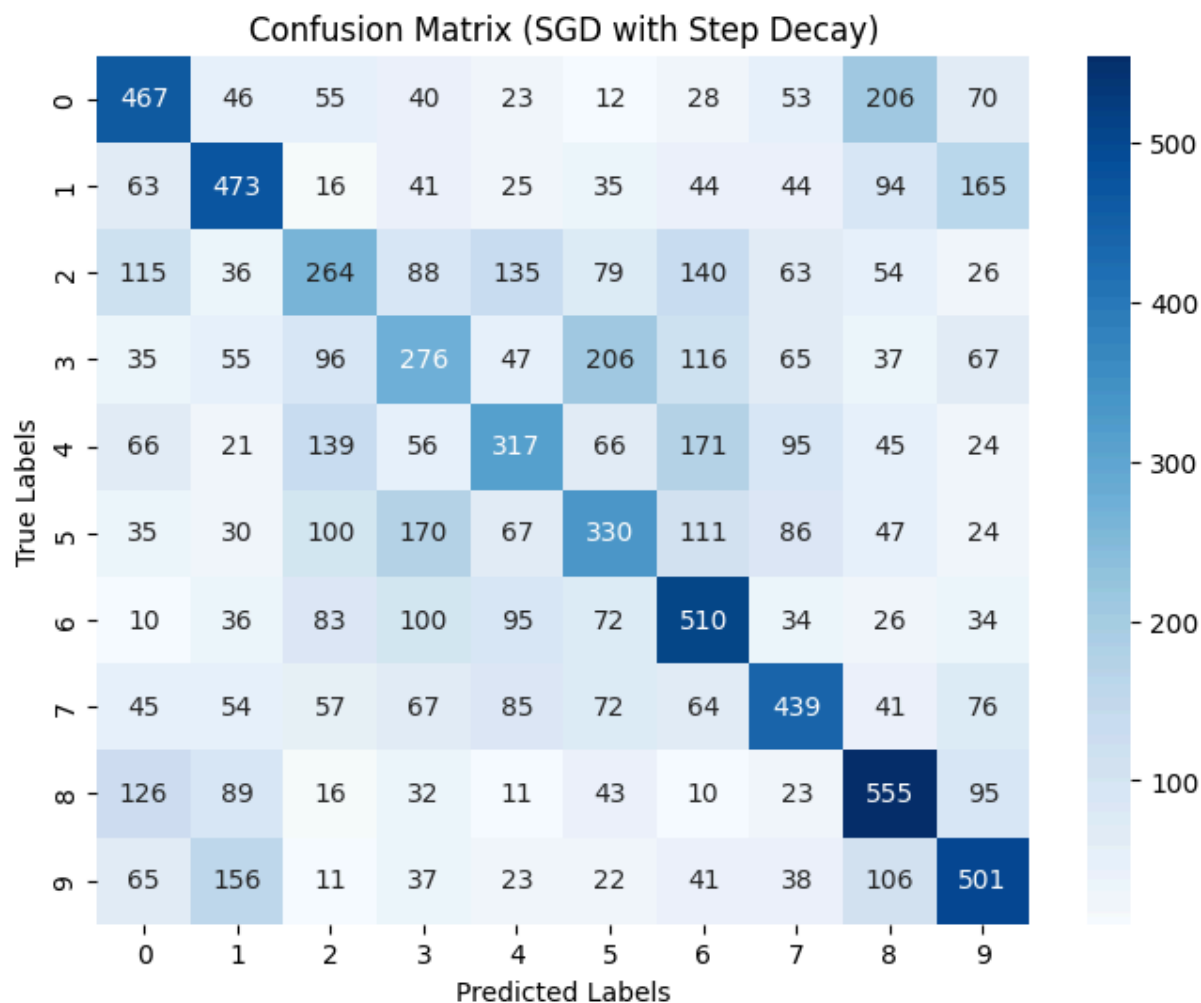
In [6]: # Plot training and validation accuracy
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
plt.plot(accuracy, label='Training Accuracy')
plt.plot(val_accuracy, label='Validation Accuracy')
plt.title('Model Accuracy (SGD with Step Decay)')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Confusion Matrix
y_pred = np.argmax(model.predict(X_test), axis=1)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=np.arange(10), ytick
plt.title('Confusion Matrix (SGD with Step Decay)')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()

```



313/313 — 0s 1ms/step



```
In [7]: # Plot training and validation accuracy
accuracy = history.history['loss']
val_accuracy = history.history['val_loss']
plt.plot(accuracy, label='Training Loss')
plt.plot(val_accuracy, label='Validation Loss')
plt.title('Model Loss (SGD with Step Decay)')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Model Loss (SGD with Step Decay)

