In [1]:
```python
# sgd_warm_restarts.py

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
import seaborn as sns
import time
```

In [2]:
```python
# 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]:
```python
# 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

# SGD with Warm Restarts optimizer
optimizer = SGD(learning_rate=tf.keras.optimizers.schedules.CosineDecayRestarts(
    0.001,
    100,
    t_mul=1.0,
    m_mul=1.0,
    alpha=0.0,
    name='SGDRDecay'
))
```

In [4]:
```python
from tensorflow.keras.callbacks import EarlyStopping


# Train and evaluate model
```

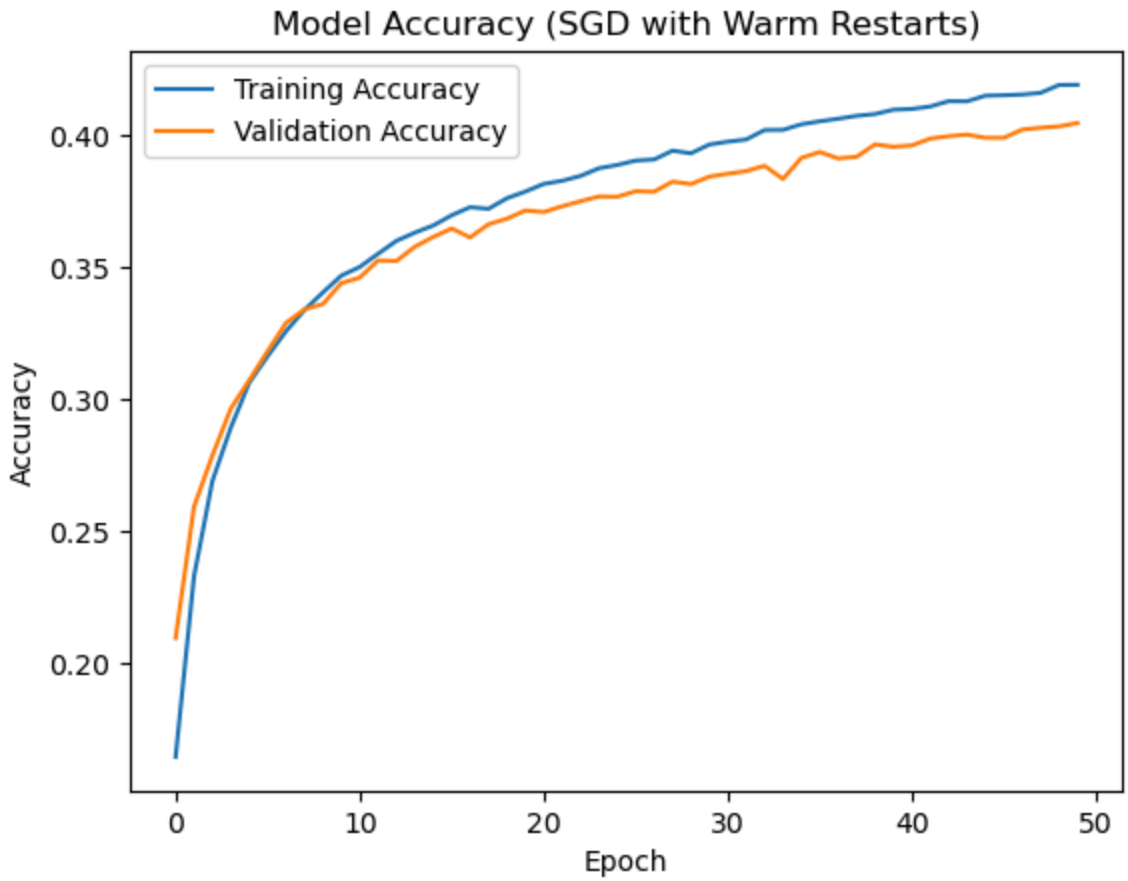
```
start_time = time.time()
model = create_model(optimizer)
early_stop = EarlyStopping(monitor='val_accuracy', patience=5, restore_best_weights

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

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

```
Epoch 1/50
625/625 [==============================] - 3s 4ms/step - loss: 2.2456 - accuracy: 0.
1647 - val_loss: 2.1726 - val_accuracy: 0.2097
Epoch 2/50
625/625 [==============================] - 2s 4ms/step - loss: 2.1304 - accuracy: 0.
2331 - val_loss: 2.0893 - val_accuracy: 0.2594
Epoch 3/50
625/625 [==============================] - 5s 8ms/step - loss: 2.0643 - accuracy: 0.
2691 - val_loss: 2.0373 - val_accuracy: 0.2788
Epoch 4/50
625/625 [==============================] - 4s 6ms/step - loss: 2.0193 - accuracy: 0.
2893 - val_loss: 2.0002 - val_accuracy: 0.2965
Epoch 5/50
625/625 [==============================] - 2s 3ms/step - loss: 1.9850 - accuracy: 0.
3063 - val_loss: 1.9687 - val_accuracy: 0.3072
Epoch 6/50
625/625 [==============================] - 2s 4ms/step - loss: 1.9570 - accuracy: 0.
3163 - val_loss: 1.9471 - val_accuracy: 0.3182
Epoch 7/50
625/625 [==============================] - 2s 4ms/step - loss: 1.9353 - accuracy: 0.
3257 - val_loss: 1.9268 - val_accuracy: 0.3289
Epoch 8/50
625/625 [==============================] - 2s 3ms/step - loss: 1.9170 - accuracy: 0.
3338 - val_loss: 1.9107 - val_accuracy: 0.3340
Epoch 9/50
625/625 [==============================] - 3s 5ms/step - loss: 1.9016 - accuracy: 0.
3404 - val_loss: 1.8971 - val_accuracy: 0.3360
Epoch 10/50
625/625 [==============================] - 3s 5ms/step - loss: 1.8873 - accuracy: 0.
3467 - val_loss: 1.8834 - val_accuracy: 0.3439
Epoch 11/50
625/625 [==============================] - 3s 4ms/step - loss: 1.8756 - accuracy: 0.
3500 - val_loss: 1.8722 - val_accuracy: 0.3460
Epoch 12/50
625/625 [==============================] - 3s 4ms/step - loss: 1.8648 - accuracy: 0.
3550 - val_loss: 1.8623 - val_accuracy: 0.3525
Epoch 13/50
625/625 [==============================] - 3s 4ms/step - loss: 1.8546 - accuracy: 0.
3599 - val_loss: 1.8536 - val_accuracy: 0.3523
Epoch 14/50
625/625 [==============================] - 2s 3ms/step - loss: 1.8457 - accuracy: 0.
3630 - val_loss: 1.8455 - val_accuracy: 0.3577
Epoch 15/50
625/625 [==============================] - 3s 4ms/step - loss: 1.8371 - accuracy: 0.
3657 - val_loss: 1.8361 - val_accuracy: 0.3614
Epoch 16/50
625/625 [==============================] - 4s 6ms/step - loss: 1.8292 - accuracy: 0.
3696 - val_loss: 1.8286 - val_accuracy: 0.3646
Epoch 17/50
625/625 [==============================] - 2s 3ms/step - loss: 1.7090 - accuracy: 0.
4128 - val_loss: 1.7195 - val_accuracy: 0.3995
Epoch 44/50
625/625 [==============================] - 2s 3ms/step - loss: 1.7065 - accuracy: 0.
4127 - val_loss: 1.7177 - val_accuracy: 0.4001
Epoch 45/50
625/625 [==============================] - 3s 4ms/step - loss: 1.7033 - accuracy: 0.
```

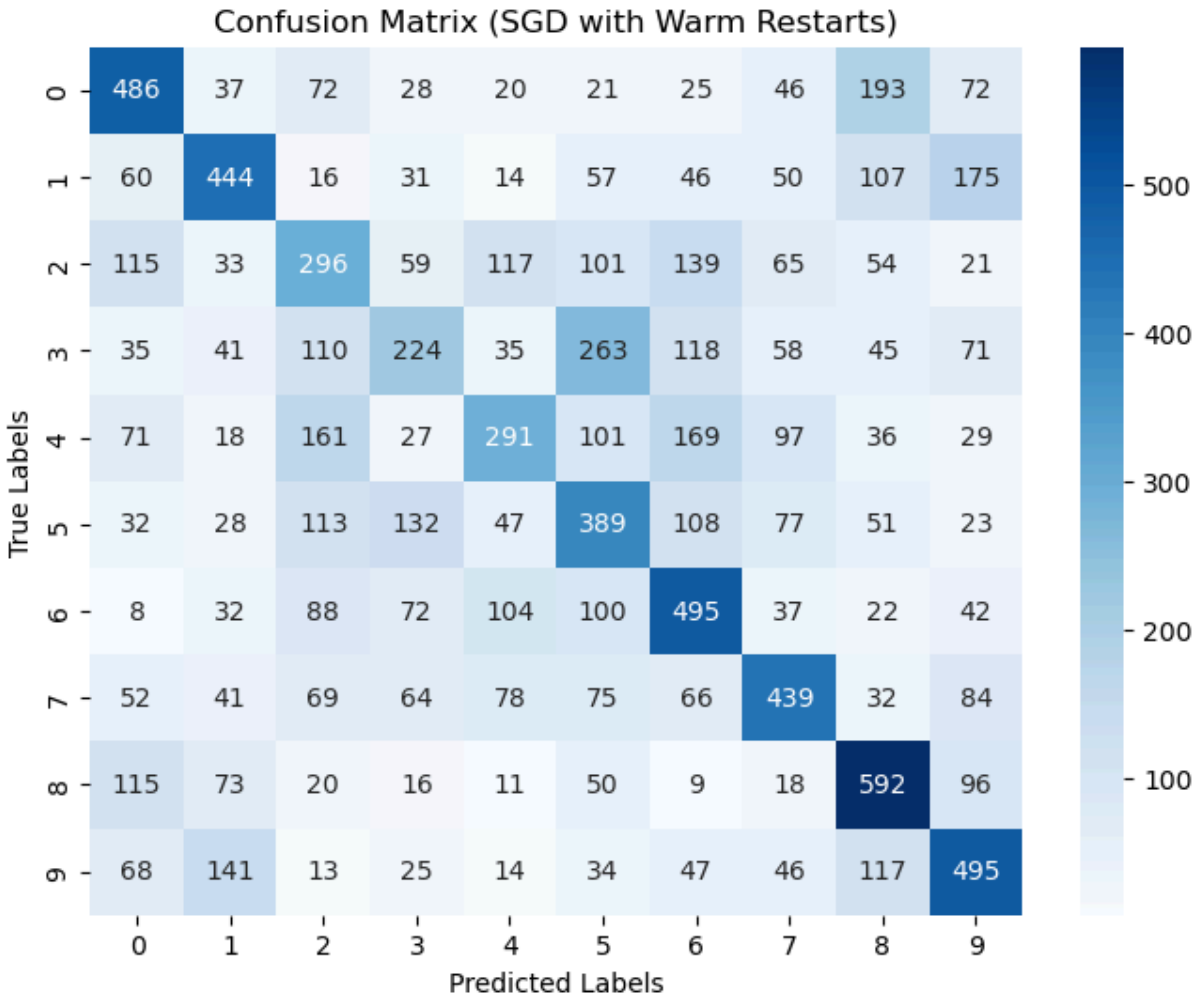
```
4148 - val_loss: 1.7148 - val_accuracy: 0.3989
Epoch 46/50
625/625 [==============================] - 3s 4ms/step - loss: 1.7003 - accuracy: 0.
4150 - val_loss: 1.7125 - val_accuracy: 0.3989
Epoch 47/50
625/625 [==============================] - 6s 9ms/step - loss: 1.6973 - accuracy: 0.
4153 - val_loss: 1.7084 - val_accuracy: 0.4020
Epoch 48/50
625/625 [==============================] - 2s 3ms/step - loss: 1.6946 - accuracy: 0.
4159 - val_loss: 1.7068 - val_accuracy: 0.4027
Epoch 49/50
625/625 [==============================] - 2s 4ms/step - loss: 1.6920 - accuracy: 0.
4189 - val_loss: 1.7046 - val_accuracy: 0.4032
Epoch 50/50
625/625 [==============================] - 3s 4ms/step - loss: 1.6893 - accuracy: 0.
4189 - val_loss: 1.7005 - val_accuracy: 0.4045
Training time: 132.54 seconds
```

In [5]:

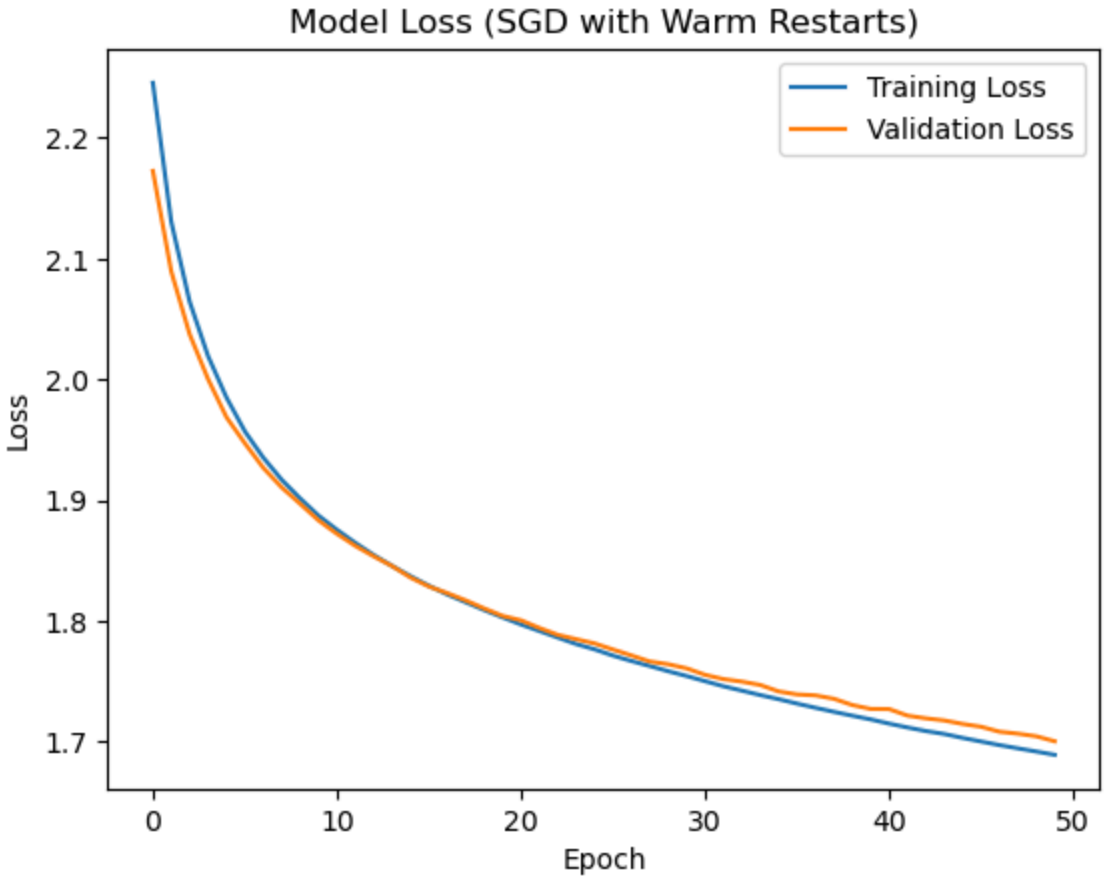
```python
# Plot training and validation accuracy
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
plt.plot(accuracy, label='Training Accuracy')
plt.plot(val_accuracy, label='Validation Accuracy')
plt.title('Model Accuracy (SGD with Warm Restarts)')
plt.xlabel('Epoch')
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 Warm Restarts)')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```




```
313/313 [==============================] - 2s 7ms/step
```



In [6]:
```python
# 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 Warm Restarts)')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```


Model Loss (SGD with Warm Restarts)
Training Loss
Validation Loss
Loss
2.2
2.1
2.0
1.9
1.8
1.7
0
10
20
30
40
50
Epoch

In [ ]: