```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.initializers import HeNormal, GlorotNormal
from tensorflow.keras import regularizers
from tensorflow.keras.layers import Dropout


# Step 1: Load and preprocess CIFAR-10 dataset
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()
train_images = train_images / 255.0
test_images = test_images / 255.0
train_labels = to_categorical(train_labels, num_classes=10)
test_labels = to_categorical(test_labels, num_classes=10)


# Step 2: Function to create a model with specified configurations
def create_model(initializer, regularizer, dropout_rate=None):
    model = Sequential([
        Flatten(input_shape=(32, 32, 3)),
        Dense(512, activation='relu', kernel_initializer=initializer, kernel_regularizer=regularizer),
        Dense(256, activation='relu', kernel_initializer=initializer, kernel_regularizer=regularizer),
        Dense(10, activation='softmax', kernel_initializer=initializer)
    ])
    if dropout_rate:
        model.add(Dropout(dropout_rate))
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model


# Step 3: Initialize models using Xavier/Glorot and Kaiming/He initializers
xavier_model = create_model(GlorotNormal(), regularizers.l2(0.001))
kaiming_model = create_model(HeNormal(), regularizers.l2(0.001))
```

```
C:\Users\csconda2\AppData\Roaming\Python\Python311\site-packages\keras\src\initializers\initializers.py:120: UserWarning: The initia
  warnings.warn(
C:\Users\csconda2\AppData\Roaming\Python\Python311\site-packages\keras\src\initializers\initializers.py:120: UserWarning: The initia
  warnings.warn(
```

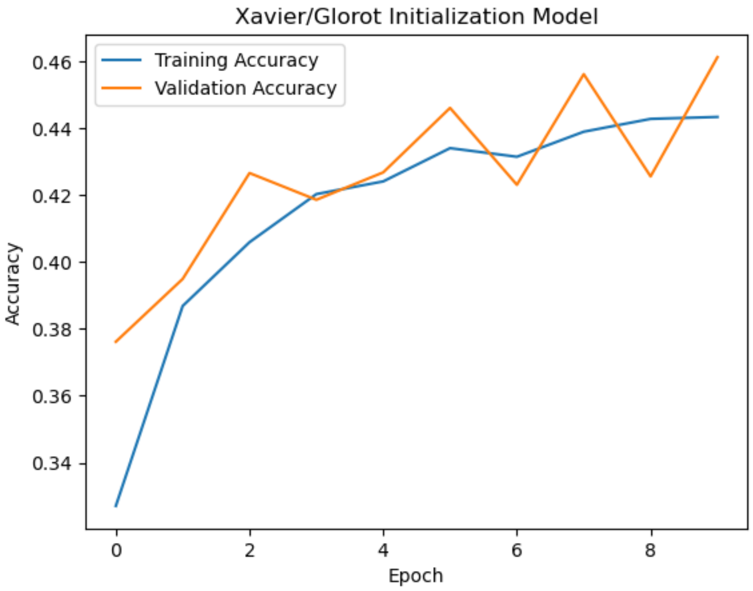
```
# Step 4: Train both models and store the training history
xavier_history = xavier_model.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))
kaiming_history = kaiming_model.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))
```

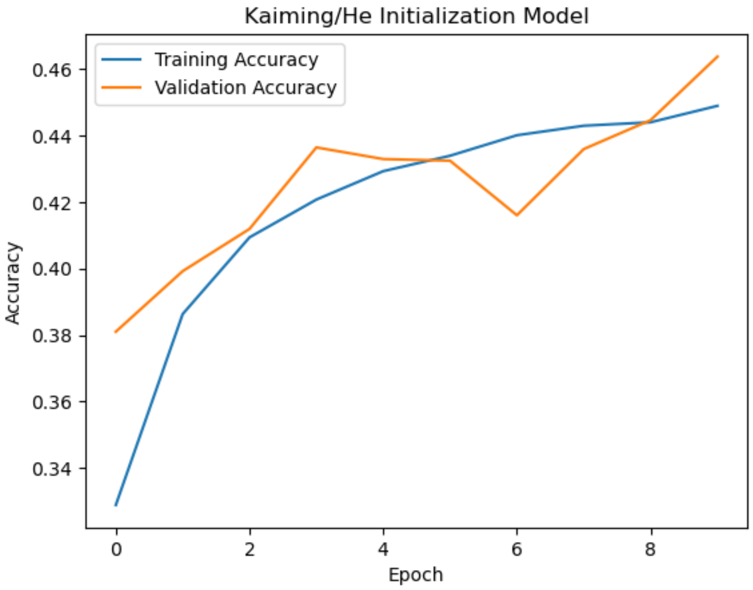
```
Epoch 1/10
1563/1563 [==============================] - 109s 68ms/step - loss: 2.1057 - accuracy: 0.3270 - val_loss: 1.8176 - val_accuracy: 0.3
Epoch 2/10
1563/1563 [==============================] - 101s 65ms/step - loss: 1.7911 - accuracy: 0.3869 - val_loss: 1.7770 - val_accuracy: 0.3
Epoch 3/10
1563/1563 [==============================] - 82s 53ms/step - loss: 1.7296 - accuracy: 0.4060 - val_loss: 1.7036 - val_accuracy: 0.42
Epoch 4/10
1563/1563 [==============================] - 44s 28ms/step - loss: 1.6905 - accuracy: 0.4204 - val_loss: 1.7053 - val_accuracy: 0.41
Epoch 5/10
1563/1563 [==============================] - 48s 31ms/step - loss: 1.6765 - accuracy: 0.4242 - val_loss: 1.6573 - val_accuracy: 0.42
Epoch 6/10
1563/1563 [==============================] - 46s 30ms/step - loss: 1.6558 - accuracy: 0.4342 - val_loss: 1.6267 - val_accuracy: 0.44
Epoch 7/10
1563/1563 [==============================] - 45s 29ms/step - loss: 1.6569 - accuracy: 0.4316 - val_loss: 1.6915 - val_accuracy: 0.42
Epoch 8/10
1563/1563 [==============================] - 44s 28ms/step - loss: 1.6453 - accuracy: 0.4391 - val_loss: 1.6145 - val_accuracy: 0.45
Epoch 9/10
1563/1563 [==============================] - 44s 28ms/step - loss: 1.6399 - accuracy: 0.4429 - val_loss: 1.6884 - val_accuracy: 0.42
Epoch 10/10
1563/1563 [==============================] - 44s 28ms/step - loss: 1.6317 - accuracy: 0.4435 - val_loss: 1.5996 - val_accuracy: 0.46
Epoch 1/10
1563/1563 [==============================] - 46s 29ms/step - loss: 2.2007 - accuracy: 0.3289 - val_loss: 1.8407 - val_accuracy: 0.38
Epoch 2/10
1563/1563 [==============================] - 45s 29ms/step - loss: 1.8053 - accuracy: 0.3863 - val_loss: 1.7920 - val_accuracy: 0.39
Epoch 3/10
1563/1563 [==============================] - 45s 29ms/step - loss: 1.7304 - accuracy: 0.4094 - val_loss: 1.7195 - val_accuracy: 0.41
Epoch 4/10
1563/1563 [==============================] - 45s 29ms/step - loss: 1.7004 - accuracy: 0.4207 - val_loss: 1.6925 - val_accuracy: 0.43
Epoch 5/10
1563/1563 [==============================] - 45s 29ms/step - loss: 1.6769 - accuracy: 0.4293 - val_loss: 1.6790 - val_accuracy: 0.43
Epoch 6/10
1563/1563 [==============================] - 48s 31ms/step - loss: 1.6680 - accuracy: 0.4339 - val_loss: 1.7122 - val_accuracy
Epoch 7/10
1563/1563 [==============================] - 47s 30ms/step - loss: 1.6504 - accuracy: 0.4401 - val_loss: 1.7083 - val_accuracy: 0.41
Epoch 8/10
1563/1563 [==============================] - 47s 30ms/step - loss: 1.6472 - accuracy: 0.4429 - val_loss: 1.6554 - val_accuracy: 0.43
Epoch 9/10
```

```
1563/1563 [==============================] - 47s 30ms/step - loss: 1.6425 - accuracy: 0.4440 - val_loss: 1.6429 - val_accuracy: 0.44
Epoch 10/10
1563/1563 [==============================] - 47s 30ms/step - loss: 1.6357 - accuracy: 0.4489 - val_loss: 1.6146 - val_accuracy: 0.46
```

```
# Step 5: Evaluate and visualize model performance
def plot_history(history, title):
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.title(title)
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

plot_history(xavier_history, 'Xavier/Glorot Initialization Model')
plot_history(kaiming_history, 'Kaiming/He Initialization Model')
```


Xavier/Glorot Initialization Model
Training Accuracy
Validation Accuracy
Accuracy
Epoch


Kaiming/He Initialization Model
Training Accuracy
Validation Accuracy
Accuracy
Epoch