```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import EfficientNetB0, ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
import numpy as np


(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()


x_train, x_test = x_train / 255.0, x_test / 255.0


y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)


img_height, img_width = 32, 32
num_classes = 10
batch_size = 32


train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)


train_generator = train_datagen.flow(x_train, y_train, batch_size=batch_size)


test_datagen = ImageDataGenerator()
test_generator = test_datagen.flow(x_test, y_test, batch_size=batch_size)


efficientnet_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))


for layer in efficientnet_model.layers:
    layer.trainable = False


efficientnet_based_model = models.Sequential()
efficientnet_based_model.add(efficientnet_model)
efficientnet_based_model.add(layers.GlobalAveragePooling2D())
efficientnet_based_model.add(layers.Dense(256, activation='relu'))
efficientnet_based_model.add(layers.Dropout(0.5))
efficientnet_based_model.add(layers.Dense(num_classes, activation='softmax'))


efficientnet_based_model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])


efficientnet_based_history = efficientnet_based_model.fit(
    train_generator,
    epochs=10,
    validation_data=test_generator
)
```

```
Epoch 1/10
1563/1563 [==============================] - 160s 96ms/step - loss: 2.3075 - accuracy: 0.0998 - val_loss: 2.3029 - val_accuracy: 0.1000
Epoch 2/10
1563/1563 [==============================] - 146s 94ms/step - loss: 2.3028 - accuracy: 0.0979 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 3/10
1563/1563 [==============================] - 149s 95ms/step - loss: 2.3027 - accuracy: 0.0973 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 4/10
1563/1563 [==============================] - 149s 95ms/step - loss: 2.3026 - accuracy: 0.0984 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 5/10
1563/1563 [==============================] - 148s 95ms/step - loss: 2.3026 - accuracy: 0.0972 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 6/10
1563/1563 [==============================] - 149s 95ms/step - loss: 2.3026 - accuracy: 0.0984 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 7/10
1563/1563 [==============================] - 143s 91ms/step - loss: 2.3026 - accuracy: 0.0953 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 8/10
```

```
1563/1563 [==============================] - 142s 91ms/step - loss: 2.3026 - accuracy: 0.0972 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 9/10
1563/1563 [==============================] - 149s 95ms/step - loss: 2.3026 - accuracy: 0.0986 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 10/10
1563/1563 [==============================] - 148s 95ms/step - loss: 2.3026 - accuracy: 0.0979 - val_loss: 2.3026 - val_accuracy: 0.1000
```

```
resnet50_model = ResNet50(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_no
94765736/94765736 [==============================] - 1s 0us/step
```

```
for layer in resnet50_model.layers:
    layer.trainable = False
```

```
resnet50_based_model = models.Sequential()
resnet50_based_model.add(resnet50_model)
resnet50_based_model.add(layers.GlobalAveragePooling2D())
resnet50_based_model.add(layers.Dense(256, activation='relu'))
resnet50_based_model.add(layers.Dropout(0.5))
resnet50_based_model.add(layers.Dense(num_classes, activation='softmax'))
```

```
resnet50_based_model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
resnet50_based_history = resnet50_based_model.fit(
    train_generator,
    epochs=10,
    validation_data=test_generator
)
```

```
Epoch 1/10
1563/1563 [==============================] - 363s 229ms/step - loss: 2.2589 - accuracy: 0.1646 - val_loss: 2.1028 - val_accuracy: 0.2623
Epoch 2/10
1563/1563 [==============================] - 318s 204ms/step - loss: 2.1303 - accuracy: 0.2155 - val_loss: 2.0398 - val_accuracy: 0.2789
Epoch 3/10
1563/1563 [==============================] - 352s 225ms/step - loss: 2.0828 - accuracy: 0.2327 - val_loss: 1.9770 - val_accuracy: 0.2986
Epoch 4/10
1563/1563 [==============================] - 356s 228ms/step - loss: 2.0495 - accuracy: 0.2467 - val_loss: 1.9496 - val_accuracy: 0.3087
Epoch 5/10
1563/1563 [==============================] - 318s 203ms/step - loss: 2.0317 - accuracy: 0.2560 - val_loss: 1.9273 - val_accuracy: 0.3191
Epoch 6/10
1563/1563 [==============================] - 355s 227ms/step - loss: 2.0163 - accuracy: 0.2634 - val_loss: 1.9013 - val_accuracy: 0.3274
Epoch 7/10
1563/1563 [==============================] - 315s 202ms/step - loss: 2.0033 - accuracy: 0.2680 - val_loss: 1.8883 - val_accuracy: 0.3278
Epoch 8/10
1563/1563 [==============================] - 355s 227ms/step - loss: 1.9886 - accuracy: 0.2731 - val_loss: 1.8879 - val_accuracy: 0.3243
Epoch 9/10
1563/1563 [==============================] - 318s 204ms/step - loss: 1.9794 - accuracy: 0.2803 - val_loss: 1.8748 - val_accuracy: 0.3328
Epoch 10/10
1563/1563 [==============================] - 352s 225ms/step - loss: 1.9707 - accuracy: 0.2834 - val_loss: 1.8653 - val_accuracy: 0.3382
```

```
efficientnet_based_accuracy = efficientnet_based_model.evaluate(test_generator)[1]
resnet50_based_accuracy = resnet50_based_model.evaluate(test_generator)[1]
```

```
313/313 [==============================] - 20s 63ms/step - loss: 2.3026 - accuracy: 0.1000
313/313 [==============================] - 47s 150ms/step - loss: 1.8653 - accuracy: 0.3382
```

```
print('EfficientNet-based Model Test Accuracy: {:.2%}'.format(efficientnet_based_accuracy))
print('ResNet50-based Model Test Accuracy: {:.2%}'.format(resnet50_based_accuracy))
```

```
EfficientNet-based Model Test Accuracy: 10.00%
ResNet50-based Model Test Accuracy: 33.82%
```

```
def predict_image(model, img_array, class_names):
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    predictions = model.predict(img_array)
    predicted_class = np.argmax(predictions)
    predicted_class_name = class_names[predicted_class]
    return predicted_class_name, predictions
```

```
image_path = 'dog_pic.jpg'
img = tf.keras.preprocessing.image.load_img(image_path, target_size=(img_height, img_width))
img_array = tf.keras.preprocessing.image.img_to_array(img)


class_names = [
    'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'
]
efficientnet_predicted_class, efficientnet_predictions = predict_image(efficientnet_based_model, img_array, class_names)
resnet50_predicted_class, resnet50_predictions = predict_image(resnet50_based_model, img_array, class_names)
```

```
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 58ms/step
```

```
print('\nPredicted Class (EfficientNet-based Model):', efficientnet_predicted_class)
print('Predictions (EfficientNet-based Model):', efficientnet_predictions)

print('\nPredicted Class (ResNet50-based Model):', resnet50_predicted_class)
print('Predictions (ResNet50-based Model):', resnet50_predictions)
```

```
Predicted Class (EfficientNet-based Model): horse
Predictions (EfficientNet-based Model): [[0.09998834 0.09995509 0.09995919 0.09996554 0.09999426 0.09989298
  0.09994327 0.1001451  0.10004739 0.10010879]]

Predicted Class (ResNet50-based Model): deer
Predictions (ResNet50-based Model): [[0.03710438 0.00719727 0.18897031 0.09318926 0.3411967  0.06149691
  0.20198525 0.03880529 0.02338013 0.00667445]]
```