```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.applications import VGG16, ResNet50, MobileNetV2
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, GlobalAveragePooling2D

# Load CIFAR-10 data
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalize data
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# One-hot encode labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Function to build and compile models
def build_and_compile_model(base_model, model_name):
    x = Flatten()(base_model.output)
    x = Dense(256, activation='relu')(x)
    x = Dense(10, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=x)
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=10, batch_size=64, validation_split=0.2)
    model.save(f'{model_name}_model.h5')

# VGG16 Model
vgg16_base = VGG16(weights=None, include_top=False, input_shape=(32, 32, 3))
build_and_compile_model(vgg16_base, 'vgg16')

# ResNet50 Model
resnet50_base = ResNet50(weights=None, include_top=False, input_shape=(32, 32, 3))
build_and_compile_model(resnet50_base, 'resnet50')

# MobileNetV2 Model
mobilenetv2_base = MobileNetV2(weights=None, include_top=False, input_shape=(32, 32, 3))
build_and_compile_model(mobilenetv2_base, 'mobilenetv2')

# Simplified YOLO-like Model
def build_yolo_like_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=10, batch_size=64, validation_split=0.2)
    model.save('yolo_like_model.h5')

build_yolo_like_model()
```

```
625/625 ━━━━━━━━━━━━━━━━━━━━ 11s 18ms/step - accuracy: 0.6405 - loss: 1.0188 - val_accuracy: 0.1829 - val_loss: 2.2009
Epoch 8/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 10s 16ms/step - accuracy: 0.6721 - loss: 0.9377 - val_accuracy: 0.4584 - val_loss: 1.6215
Epoch 9/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy: 0.6896 - loss: 0.8874 - val_accuracy: 0.5699 - val_loss: 1.3109
Epoch 10/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 11s 16ms/step - accuracy: 0.7056 - loss: 0.8488 - val_accuracy: 0.5602 - val_loss: 2.2126
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `in
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - accuracy: 0.3741 - loss: 1.7297 - val_accuracy: 0.5540 - val_loss: 1.2482
Epoch 2/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.5859 - loss: 1.1732 - val_accuracy: 0.6173 - val_loss: 1.0996
Epoch 3/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.6561 - loss: 0.9860 - val_accuracy: 0.6500 - val_loss: 1.0146
Epoch 4/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6981 - loss: 0.8748 - val_accuracy: 0.6608 - val_loss: 0.9891
Epoch 5/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.7359 - loss: 0.7564 - val_accuracy: 0.6829 - val_loss: 0.9496
Epoch 6/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.7671 - loss: 0.6703 - val_accuracy: 0.6904 - val_loss: 0.9195
Epoch 7/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.7996 - loss: 0.5799 - val_accuracy: 0.6938 - val_loss: 0.9222
Epoch 8/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8319 - loss: 0.4900 - val_accuracy: 0.7107 - val_loss: 0.9046
Epoch 9/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8606 - loss: 0.4080 - val_accuracy: 0.7071 - val_loss: 0.9722
Epoch 10/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8910 - loss: 0.3284 - val_accuracy: 0.7070 - val_loss: 1.0154
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file
```

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical

# Load CIFAR-10 test dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalize data
x_test = x_test.astype('float32') / 255.0

# One-hot encode labels
y_test = to_categorical(y_test, 10)

# Load the trained models
vgg16_model = load_model('vgg16_model.h5')
resnet50_model = load_model('resnet50_model.h5')
mobilenetv2_model = load_model('mobilenetv2_model.h5')
yolo_like_model = load_model('yolo_like_model.h5')

# Select a few test images
num_images = 5
test_images = x_test[:num_images]
test_labels = y_test[:num_images]

# Make predictions
vgg16_preds = vgg16_model.predict(test_images)
resnet50_preds = resnet50_model.predict(test_images)
mobilenetv2_preds = mobilenetv2_model.predict(test_images)
yolo_like_preds = yolo_like_model.predict(test_images)

# Convert predictions to class labels
vgg16_labels = np.argmax(vgg16_preds, axis=1)
resnet50_labels = np.argmax(resnet50_preds, axis=1)
mobilenetv2_labels = np.argmax(mobilenetv2_preds, axis=1)
yolo_like_labels = np.argmax(yolo_like_preds, axis=1)

# CIFAR-10 class names
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# Visualize predictions
plt.figure(figsize=(12, 10))

for i in range(num_images):
    plt.subplot(num_images, 1, i + 1)
    plt.imshow(test_images[i])
    plt.title(f"True Label: {class_names[np.argmax(test_labels[i])]}\n"
              f"VGG16: {class_names[vgg16_labels[i]]} | "
              f"ResNet50: {class_names[resnet50_labels[i]]} | "
              f"MobileNetV2: {class_names[mobilenetv2_labels[i]]} | "
              f"YOLO-Like: {class_names[yolo_like_labels[i]]}")
    plt.axis('off')
```

```
plt.tight_layout()
plt.show()
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be e
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be e
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be e
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be e
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 328ms/step
```