```
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, LeakyReLU, MaxPooling2D, Layer
from tensorflow.keras.models import Model
import numpy as np

def generate_random_data(image_size=(416, 416, 3), num_samples=10):
    images = np.random.rand(num_samples, *image_size)
    labels = np.random.rand(num_samples, 13, 13, 3, 85)
    return images, labels

def conv_block(x, filters, size, strides=1):
    x = Conv2D(filters, size, strides=strides, padding='same', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.1)(x)
    return x

class YOLOReshaped(Layer):
    def __init__(self, num_classes, **kwargs):
        super(YOLOReshaped, self).__init__(**kwargs)
        self.num_classes = num_classes

    def call(self, inputs):
        # Reshape to (batch_size, 13, 13, 3, num_classes + 5)
        return tf.keras.layers.Reshape((13, 13, 3, self.num_classes + 5))(inputs)

    def get_config(self):
        config = super(YOLOReshaped, self).get_config()
        config.update({"num_classes": self.num_classes})
        return config

def yolo_v3(input_shape, num_classes):
    inputs = Input(input_shape)

    # Downsample to 208x208
    x = conv_block(inputs, 32, 3)
    x = MaxPooling2D(2)(x)

    # Downsample to 104x104
    x = conv_block(x, 64, 3)
    x = MaxPooling2D(2)(x)

    # Downsample to 52x52
    x = conv_block(x, 128, 3)
    x = MaxPooling2D(2)(x)

    # Downsample to 26x26
    x = conv_block(x, 256, 3)
    x = MaxPooling2D(2)(x)

    # Downsample to 13x13
    x = conv_block(x, 512, 3)
    x = MaxPooling2D(2)(x)

    # Additional convolutions at 13x13
    x = conv_block(x, 1024, 3)
    x = conv_block(x, 512, 1)
    x = conv_block(x, 1024, 3)

    # Output layer
    filters = 3 * (num_classes + 5)  # 3 anchors * (num_classes + 5)
    x = Conv2D(filters, (1, 1), padding='same')(x)

    # Use custom reshape layer instead of tf.reshape
    outputs = YOLOReshaped(num_classes)(x)

    model = Model(inputs, outputs)
    return model

def yolo_loss(y_true, y_pred):
    return tf.reduce_mean(tf.square(y_true - y_pred))

# Model parameters
input_shape = (416, 416, 3)
num_classes = 80

# Create and compile model
```

```
model = yolo_v3(input_shape, num_classes)
model.compile(optimizer='adam', loss=yolo_loss, metrics=['accuracy'])

# Generate random data
images, labels = generate_random_data(image_size=input_shape, num_samples=10)

# Train model
history = model.fit(images, labels, epochs=1, batch_size=2)

# Make predictions
predictions = model.predict(images)
print("Prediction shape:", predictions.shape)
print("Sample prediction output (first box):")
print(predictions[0, 0, 0, 0, :5])  # Print first 5 values of first prediction box
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 14s 2s/step - accuracy: 0.0110 - loss: 0.9003
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Prediction shape: (10, 13, 13, 3, 85)
Sample prediction output (first box):
[0.04064313 0.02663211 0.02915632 0.03943413 0.02215367]
```