In [15]:
```
import tensorflow as tf
from tensorflow.keras import layers, Model
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.losses import BinaryCrossentropy, Huber
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import MeanIoU

# Define the number of epochs
epochs = 3

# Load the CIFAR-10 dataset
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.ci

# Normalize pixel values to range [0, 1]
train_images, test_images = train_images / 255.0, test_images / 255.0

# Define the backbone ResNet50 model
backbone = ResNet50(include_top=False, weights='imagenet', input_shape=(32, 32,

# Define classification and regression heads
classification_head = layers.GlobalAveragePooling2D()(backbone.output)
classification_head = layers.Dense(10, activation='softmax', name='classificati

regression_head = layers.GlobalAveragePooling2D()(backbone.output)
regression_head = layers.Dense(4, name='regression_output')(regression_head)

# Combine backbone with classification and regression heads
model = Model(inputs=backbone.input, outputs=[classification_head, regression_h

# Define loss functions
classification_loss = BinaryCrossentropy()
regression_loss = Huber()

# Define metrics
classification_metrics = [MeanIoU(num_classes=10)]
regression_metrics = ['mae']

# Define optimizer
optimizer = Adam()

# Compile the model
model.compile(optimizer=optimizer, loss=[classification_loss, regression_loss],

# One-hot encode the target labels for classification
num_classes = 10
train_labels_categorical = tf.keras.utils.to_categorical(train_labels, num_clas
test_labels_categorical = tf.keras.utils.to_categorical(test_labels, num_classe

# Train the model
model.fit(train_images, [train_labels_categorical, train_labels], epochs=epochs

# Evaluate the model
results = model.evaluate(test_images, [test_labels_categorical, test_labels])

# Make predictions
predictions = model.predict(test_images)
```

```
Epoch 1/3
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 555s 345ms/step - classification_output_mean_i
o_u_14: 0.4500 - loss: 2.2533 - regression_output_mae: 2.3814 - val_classific
ation_output_mean_io_u_14: 0.4500 - val_loss: 2.2919 - val_regression_output_
mae: 2.4112
Epoch 2/3
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 540s 346ms/step - classification_output_mean_i
o_u_14: 0.4500 - loss: 1.7105 - regression_output_mae: 1.8663 - val_classific
ation_output_mean_io_u_14: 0.4500 - val_loss: 1.7751 - val_regression_output_
mae: 1.9157
Epoch 3/3
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 542s 347ms/step - classification_output_mean_i
o_u_14: 0.4500 - loss: 1.3973 - regression_output_mae: 1.5648 - val_classific
ation_output_mean_io_u_14: 0.4500 - val_loss: 1.4551 - val_regression_output_
mae: 1.6089
313/313 ━━━━━━━━━━━━━━━━━━━━ 13s 41ms/step - classification_output_mean_io_u_
14: 0.4500 - loss: 1.4523 - regression_output_mae: 1.6049
313/313 ━━━━━━━━━━━━━━━━━━━━ 14s 41ms/step
```

In [ ]: