Importing relevant libaries.

In [2]:

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        os.path.join(dirname, filename)
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
import pydicom
from skimage.transform import resize
import matplotlib.patches as patches
from tqdm import tqdm
import math

import tensorflow as tf
from tensorflow.keras.utils import plot_model
```

Loading labels and checking dimensions.

In [4]:

```
train_label = pd.read_csv(r'C:\Users\Sanan\Downloads\CompApp Project\rsna-pneumonia
train_label.shape
```

Out[4]:

```
(30227, 6)
```

Creating function designed to resize an image to fit within a fixed input_size while maintaining aspect ratio, while adjusting the bounding box accordingly.

In [5]:

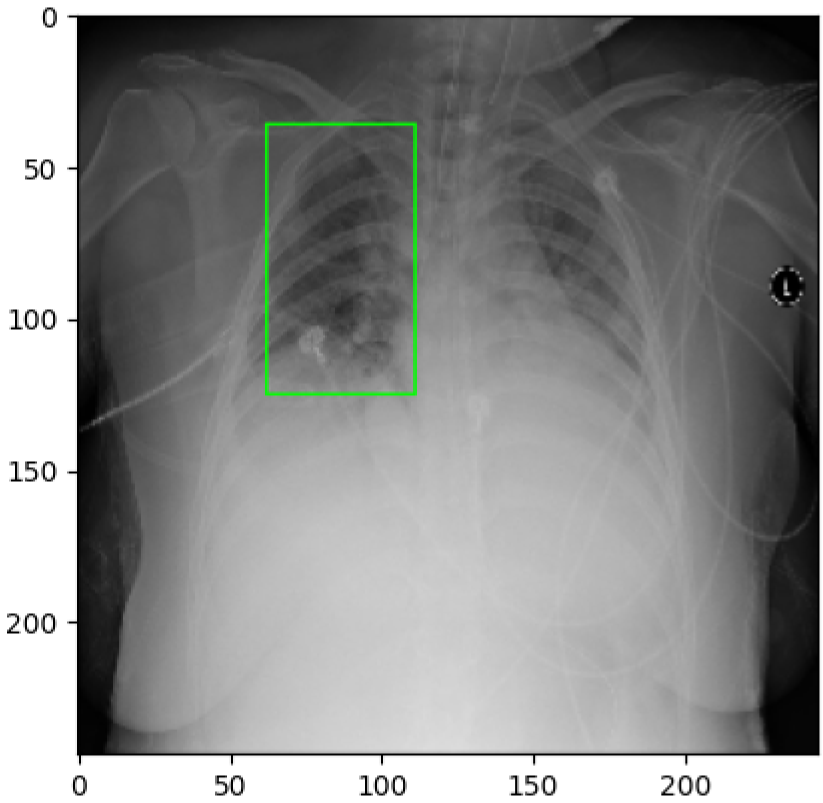
```
input_size = 244

def format_image(img, box):
    height, width = img.shape
    max_size = max(height, width)
    r = max_size / input_size
    new_width = int(width / r)
    new_height = int(height / r)
    new_size = (new_width, new_height)
    resized = cv.resize(img, new_size, interpolation= cv.INTER_LINEAR)
    new_image = np.zeros((input_size, input_size), dtype=np.uint8)
    new_image[0:new_height, 0:new_width] = resized

    x, y, w, h = (box[0], box[1], box[2], box[3]) if box[0] else (0.0,0.0,0.0,0.0)
    new_box = [int((x)/ r), int((y)/ r), int(w/ r), int(h/ r)] if box[0] else [0.0,

    return new_image, new_box
```

This function processes a DICOM medical image, resizes it, adjusts the bounding box, converts it to RGB, and visualizes it with the bounding box overlay.

In [6]:

```
datapath = 'rsna-pneumonia-detection-challenge\stage_2_train_images/00436515-870c-4
temp_img = pydicom.dcmread(datapath).pixel_array
temp_box = [264.0, 152.0, 213.0, 379.0]

temp_img_formated, box = format_image(temp_img, temp_box)
print(box)
temp_color_img = cv.cvtColor(temp_img_formated, cv.COLOR_GRAY2RGB)

cv.rectangle(temp_color_img, box, (0, 255, 0), 1)

plt.imshow(temp_color_img)
# plt.axis("off")
plt.show()
```

```
[62, 36, 50, 90]
```



Load labels and display first 5 rows.

In [7]:

```
train_labels = pd.read_csv('rsna-pneumonia-detection-challenge\stage_2_train_labels
train_labels.head()
```

Out[7]:

	patientId	x	y	width	height	Target
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	NaN	NaN	NaN	NaN	0
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	NaN	NaN	NaN	NaN	0
2	00322d4d-1c29-4943-afc9-b6754be640eb	NaN	NaN	NaN	NaN	0
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	NaN	NaN	NaN	NaN	0
4	00436515-870c-4b36-a041-de91049b9ab4	264.0	152.0	213.0	379.0	1

Creates and uses a function to load and process the DICOM images of Lung CTs and labels.

In [8]:

```
def data_load(dataset, batch_size=3, full_data_path=r"rsna-pneumonia-detection-chal
    X = []
    Y = []

    for index, row in tqdm(dataset.iterrows(), total=len(dataset), desc="Loading da
        filename = row['patientId']

        temp_img = pydicom.dcmread(os.path.join(full_data_path, filename + image_ex

        temp_box = [row['x'], row['y'], row['width'], row['height']] if not math.is

        img, box = format_image(temp_img, temp_box)

        img = img.astype(float) / 255.
        box = np.asarray(box, dtype=float) / input_size

        label = np.append(box, row['Target'])

        X.append(img)
        Y.append(label)

    X = np.array(X)
    data_X_len = len(X)
    X = np.expand_dims(X, axis=3)
    X = tf.convert_to_tensor(X, dtype=tf.float32)
    Y = tf.convert_to_tensor(Y, dtype=tf.float32)

    result = tf.data.Dataset.from_tensor_slices((X, Y))

    return result,data_X_len
raw_train_ds,train_len = data_load(train_labels[:5200],ds_type="train")
print(train_len)
raw_valid_ds,valid_len = data_load(train_labels[5200:5900],ds_type="not train")
raw_test_ds, test_len = data_load(train_labels[5900:6501],ds_type="not train")
```

```
Loading data: 100%|██████████| 5200/5200 [02:41<00:00, 32.30it/s]
5200
Loading data: 100%|██████████| 700/700 [00:23<00:00, 30.31it/s]
Loading data: 100%|██████████| 601/601 [00:19<00:00, 30.72it/s]
```

Defines a function to ready the images, label pair to be used in TensorFlow.

In [9]:

```
CLASSES = 2

def format_instance(image, label):
    return image, (tf.one_hot(int(label[4]), CLASSES), [label[0], label[1], label[2
```

Defines a function to optimize the training dataset for the tensor flow model.

In [11]:

```
BATCH_SIZE = 32

def tune_training_ds(dataset):
    dataset = dataset.map(format_instance, num_parallel_calls=tf.data.AUTOTUNE)
    dataset = dataset.shuffle(1024, reshuffle_each_iteration=True)
    dataset = dataset.repeat() # The dataset be repeated indefinitely.
    dataset = dataset.batch(BATCH_SIZE)
    dataset = dataset.prefetch(tf.data.AUTOTUNE)
    return dataset
```

Defines a function to optimize the validation dataset for the tensor flow model.

In [12]:

```
def tune_validation_ds(dataset):
    dataset = dataset.map(format_instance, num_parallel_calls=tf.data.AUTOTUNE)
    dataset = dataset.batch(len(dataset) // 4)
    dataset = dataset.repeat()
    return dataset
```

Using the above functions, preparing the training and validation datasets.

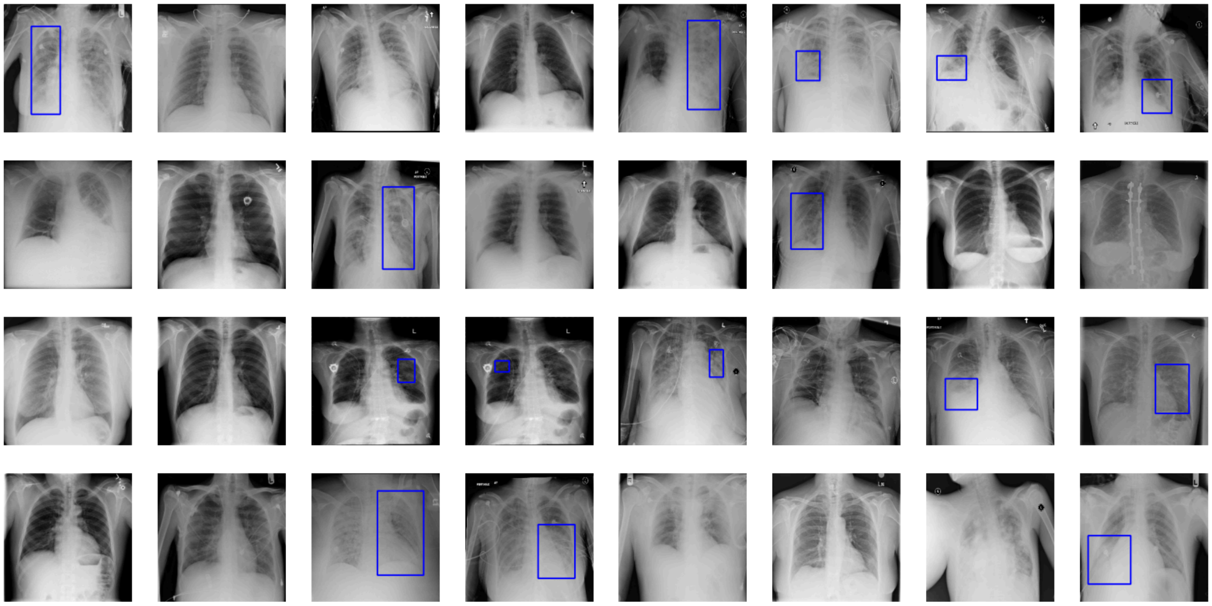
In [13]:

```
train_ds = tune_training_ds(raw_train_ds)
validation_ds = tune_validation_ds(raw_valid_ds)
```

Visualizing part of the training data set.

In [14]:

```
plt.figure(figsize=(20, 10))
for images, labels in train_ds.take(1):
    for i in range(BATCH_SIZE):
        # print(labels.shape)
        ax = plt.subplot(4, BATCH_SIZE//4, i + 1)
        label = labels[0][i]
        box = (labels[1][i] * input_size)
        box = tf.cast(box, tf.int32)

        image = images[i].numpy().astype("float") * 255.0
        image = image.astype(np.uint8)
        image_color = cv.cvtColor(image, cv.COLOR_GRAY2RGB)

        color = (0, 0, 255)
        if label[0] > 0:
            color = (0, 255, 0)

        cv.rectangle(image_color, box.numpy(), color, 2)
        plt.imshow(image_color)
        plt.axis("off")
```



Building model architecture to perform classification (detecting pneumonia) and bounding box regression (localizing pneumonia in X-ray images).

In [15]:
```
DROPOUT_FACTOR = 0.5

def build_feature_extractor(inputs):

    x = tf.keras.layers.Conv2D(16, kernel_size=3, activation='relu', input_shape=(i
    x = tf.keras.layers.AveragePooling2D(2,2)(x)

    x = tf.keras.layers.Conv2D(32, kernel_size=3, activation = 'relu')(x)
    x = tf.keras.layers.AveragePooling2D(2,2)(x)

    x = tf.keras.layers.Conv2D(64, kernel_size=3, activation = 'relu')(x)
    x = tf.keras.layers.Dropout(DROPOUT_FACTOR)(x)
    x = tf.keras.layers.AveragePooling2D(2,2)(x)

    return x

def build_model_adaptor(inputs):
    x = tf.keras.layers.Flatten()(inputs)
    x = tf.keras.layers.Dense(64, activation='relu')(x)
    return x

def build_classifier_head(inputs):
    return tf.keras.layers.Dense(CLASSES, activation='softmax', name = 'classifier_

def build_regressor_head(inputs):
    return tf.keras.layers.Dense(units = 4, name = 'regressor_head')(inputs)

def build_model(inputs):

    feature_extractor = build_feature_extractor(inputs)
    model_adaptor = build_model_adaptor(feature_extractor)

    classification_head = build_classifier_head(model_adaptor)

    regressor_head = build_regressor_head(model_adaptor)

    model = tf.keras.Model(inputs = inputs, outputs = [classification_head, regress

    model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss = {'classifier_head' : 'categorical_crossentropy', 'regressor_he
              metrics = {'classifier_head' : 'accuracy', 'regressor_head' : 'mse' }

    return model
```

Initializing the model with the standardized image dimensions and displaying the model structure.

In [16]:
```
model = build_model(tf.keras.layers.Input(shape=(input_size, input_size, 1,)))

model.summary()
```

```
C:\Users\Sanan\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfr
a8p0\LocalCache\local-packages\Python311\site-packages\keras\src\layers\convolutiona
l\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument t
o a layer. When using Sequential models, prefer using an `Input(shape)` object as th
e first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "functional"

```
┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)        ┃ Output Shape      ┃    Param # ┃ Connected to      ┃
┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━┩
│ input_layer         │ (None, 244, 244,  │          0 │ -                 │
│ (InputLayer)        │ 1)                │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv2d (Conv2D)     │ (None, 242, 242,  │        160 │ input_layer[0][0] │
│                     │ 16)               │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ average_pooling2d   │ (None, 121, 121,  │          0 │ conv2d[0][0]      │
│ (AveragePooling2D)  │ 16)               │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv2d_1 (Conv2D)   │ (None, 119, 119,  │      4,640 │ average_pooling2… │
│                     │ 32)               │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ average_pooling2d_1 │ (None, 59, 59,    │          0 │ conv2d_1[0][0]    │
│ (AveragePooling2D)  │ 32)               │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv2d_2 (Conv2D)   │ (None, 57, 57,    │     18,496 │ average_pooling2… │
│                     │ 64)               │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dropout (Dropout)   │ (None, 57, 57,    │          0 │ conv2d_2[0][0]    │
│                     │ 64)               │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ average_pooling2d_2 │ (None, 28, 28,    │          0 │ dropout[0][0]     │
│ (AveragePooling2D)  │ 64)               │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ flatten (Flatten)   │ (None, 50176)     │          0 │ average_pooling2… │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dense (Dense)       │ (None, 64)        │  3,211,328 │ flatten[0][0]     │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ classifier_head     │ (None, 2)         │        130 │ dense[0][0]       │
│ (Dense)             │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ regressor_head      │ (None, 4)         │        260 │ dense[0][0]       │
│ (Dense)             │                   │            │                   │
└─────────────────────┴───────────────────┴────────────┴───────────────────┘
 Total params: 3,235,014 (12.34 MB)
 Trainable params: 3,235,014 (12.34 MB)
 Non-trainable params: 0 (0.00 B)
```

Visualizing model strucutre in another way.

In [41]:

```
from tensorflow.keras.utils import plot_model

plot_model(model, show_shapes=True, show_layer_names=True)
```

```
You must install pydot (`pip install pydot`) for `plot_model` to work.
```

Training the model.

In [17]:

```
history = model.fit(train_ds,
                    steps_per_epoch=(6000 // BATCH_SIZE),
                    validation_data=validation_ds, validation_steps=1,
                    epochs=100)
```

```
Epoch 1/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 46s 216ms/step - classifier_head_accuracy: 0.6468 - cla
ssifier_head_loss: 0.6232 - loss: 0.7324 - regressor_head_loss: 0.1093 - regressor_h
ead_mse: 0.1093 - val_classifier_head_accuracy: 0.7257 - val_classifier_head_loss:
0.6091 - val_loss: 0.6427 - val_regressor_head_loss: 0.0336 - val_regressor_head_ms
e: 0.0336
Epoch 2/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 210ms/step - classifier_head_accuracy: 0.7603 - cla
ssifier_head_loss: 0.5045 - loss: 0.5396 - regressor_head_loss: 0.0351 - regressor_h
ead_mse: 0.0351 - val_classifier_head_accuracy: 0.7486 - val_classifier_head_loss:
0.5078 - val_loss: 0.5263 - val_regressor_head_loss: 0.0185 - val_regressor_head_ms
e: 0.0185
Epoch 3/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 211ms/step - classifier_head_accuracy: 0.7682 - cla
ssifier_head_loss: 0.4903 - loss: 0.5217 - regressor_head_loss: 0.0314 - regressor_h
ead_mse: 0.0314 - val_classifier_head_accuracy: 0.6629 - val_classifier_head_loss:
0.6634 - val_loss: 0.6998 - val_regressor_head_loss: 0.0364 - val_regressor_head_ms
e: 0.0364
Epoch 4/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 208ms/step - classifier_head_accuracy: 0.7798 - cla
ssifier_head_loss: 0.4686 - loss: 0.4985 - regressor_head_loss: 0.0300 - regressor_h
ead_mse: 0.0300 - val_classifier_head_accuracy: 0.6514 - val_classifier_head_loss:
0.6827 - val_loss: 0.7124 - val_regressor_head_loss: 0.0297 - val_regressor_head_ms
e: 0.0297
Epoch 5/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 210ms/step - classifier_head_accuracy: 0.7776 - cla
ssifier_head_loss: 0.4695 - loss: 0.4998 - regressor_head_loss: 0.0303 - regressor_h
ead_mse: 0.0303 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
0.5079 - val_loss: 0.5259 - val_regressor_head_loss: 0.0180 - val_regressor_head_ms
e: 0.0180
Epoch 6/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 209ms/step - classifier_head_accuracy: 0.7875 - cla
ssifier_head_loss: 0.4539 - loss: 0.4823 - regressor_head_loss: 0.0284 - regressor_h
ead_mse: 0.0284 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_loss:
0.4271 - val_loss: 0.4441 - val_regressor_head_loss: 0.0171 - val_regressor_head_ms
e: 0.0171
Epoch 7/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 211ms/step - classifier_head_accuracy: 0.7997 - cla
ssifier_head_loss: 0.4311 - loss: 0.4581 - regressor_head_loss: 0.0270 - regressor_h
ead_mse: 0.0270 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_loss:
0.4704 - val_loss: 0.4950 - val_regressor_head_loss: 0.0246 - val_regressor_head_ms
e: 0.0246
Epoch 8/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 215ms/step - classifier_head_accuracy: 0.8196 - cla
ssifier_head_loss: 0.4092 - loss: 0.4364 - regressor_head_loss: 0.0272 - regressor_h
ead_mse: 0.0272 - val_classifier_head_accuracy: 0.7600 - val_classifier_head_loss:
0.4895 - val_loss: 0.5075 - val_regressor_head_loss: 0.0180 - val_regressor_head_ms
e: 0.0180
Epoch 9/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.8323 - cla
ssifier_head_loss: 0.3694 - loss: 0.3955 - regressor_head_loss: 0.0261 - regressor_h
ead_mse: 0.0261 - val_classifier_head_accuracy: 0.7600 - val_classifier_head_loss:
0.5952 - val_loss: 0.6190 - val_regressor_head_loss: 0.0238 - val_regressor_head_ms
e: 0.0238
Epoch 10/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 211ms/step - classifier_head_accuracy: 0.8463 - cla
```

```
ssifier_head_loss: 0.3521 - loss: 0.3771 - regressor_head_loss: 0.0250 - regressor_h
ead_mse: 0.0250 - val_classifier_head_accuracy: 0.7371 - val_classifier_head_loss:
0.6611 - val_loss: 0.6849 - val_regressor_head_loss: 0.0238 - val_regressor_head_ms
e: 0.0238
Epoch 11/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 224ms/step - classifier_head_accuracy: 0.8576 - cla
ssifier_head_loss: 0.3304 - loss: 0.3536 - regressor_head_loss: 0.0232 - regressor_h
ead_mse: 0.0232 - val_classifier_head_accuracy: 0.7200 - val_classifier_head_loss:
0.6706 - val_loss: 0.6952 - val_regressor_head_loss: 0.0246 - val_regressor_head_ms
e: 0.0246
Epoch 12/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 43s 230ms/step - classifier_head_accuracy: 0.8727 - cla
ssifier_head_loss: 0.2956 - loss: 0.3175 - regressor_head_loss: 0.0219 - regressor_h
ead_mse: 0.0219 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
0.5039 - val_loss: 0.5236 - val_regressor_head_loss: 0.0196 - val_regressor_head_ms
e: 0.0196
Epoch 13/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 210ms/step - classifier_head_accuracy: 0.8896 - cla
ssifier_head_loss: 0.2717 - loss: 0.2916 - regressor_head_loss: 0.0199 - regressor_h
ead_mse: 0.0199 - val_classifier_head_accuracy: 0.8286 - val_classifier_head_loss:
0.4894 - val_loss: 0.5090 - val_regressor_head_loss: 0.0196 - val_regressor_head_ms
e: 0.0196
Epoch 14/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 210ms/step - classifier_head_accuracy: 0.8915 - cla
ssifier_head_loss: 0.2516 - loss: 0.2726 - regressor_head_loss: 0.0211 - regressor_h
ead_mse: 0.0211 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
0.4354 - val_loss: 0.4537 - val_regressor_head_loss: 0.0183 - val_regressor_head_ms
e: 0.0183
Epoch 15/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9046 - cla
ssifier_head_loss: 0.2220 - loss: 0.2432 - regressor_head_loss: 0.0212 - regressor_h
ead_mse: 0.0212 - val_classifier_head_accuracy: 0.7486 - val_classifier_head_loss:
0.6907 - val_loss: 0.7143 - val_regressor_head_loss: 0.0236 - val_regressor_head_ms
e: 0.0236
Epoch 16/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 210ms/step - classifier_head_accuracy: 0.9155 - cla
ssifier_head_loss: 0.2036 - loss: 0.2242 - regressor_head_loss: 0.0206 - regressor_h
ead_mse: 0.0206 - val_classifier_head_accuracy: 0.7543 - val_classifier_head_loss:
0.6855 - val_loss: 0.7110 - val_regressor_head_loss: 0.0255 - val_regressor_head_ms
e: 0.0255
Epoch 17/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 211ms/step - classifier_head_accuracy: 0.9185 - cla
ssifier_head_loss: 0.1870 - loss: 0.2075 - regressor_head_loss: 0.0205 - regressor_h
ead_mse: 0.0205 - val_classifier_head_accuracy: 0.6514 - val_classifier_head_loss:
1.0186 - val_loss: 1.0488 - val_regressor_head_loss: 0.0303 - val_regressor_head_ms
e: 0.0303
Epoch 18/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 211ms/step - classifier_head_accuracy: 0.9286 - cla
ssifier_head_loss: 0.1714 - loss: 0.1916 - regressor_head_loss: 0.0202 - regressor_h
ead_mse: 0.0202 - val_classifier_head_accuracy: 0.6971 - val_classifier_head_loss:
0.8996 - val_loss: 0.9282 - val_regressor_head_loss: 0.0286 - val_regressor_head_ms
e: 0.0286
Epoch 19/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 210ms/step - classifier_head_accuracy: 0.9359 - cla
ssifier_head_loss: 0.1538 - loss: 0.1736 - regressor_head_loss: 0.0198 - regressor_h
ead_mse: 0.0198 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_loss:
```

```
0.6495 - val_loss: 0.6691 - val_regressor_head_loss: 0.0196 - val_regressor_head_ms
e: 0.0196
Epoch 20/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 211ms/step - classifier_head_accuracy: 0.9410 - cla
ssifier_head_loss: 0.1401 - loss: 0.1590 - regressor_head_loss: 0.0189 - regressor_h
ead_mse: 0.0189 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
0.7286 - val_loss: 0.7482 - val_regressor_head_loss: 0.0196 - val_regressor_head_ms
e: 0.0196
Epoch 21/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9459 - cla
ssifier_head_loss: 0.1261 - loss: 0.1456 - regressor_head_loss: 0.0195 - regressor_h
ead_mse: 0.0195 - val_classifier_head_accuracy: 0.7600 - val_classifier_head_loss:
0.9365 - val_loss: 0.9600 - val_regressor_head_loss: 0.0235 - val_regressor_head_ms
e: 0.0235
Epoch 22/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9503 - cla
ssifier_head_loss: 0.1123 - loss: 0.1315 - regressor_head_loss: 0.0192 - regressor_h
ead_mse: 0.0192 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
1.0586 - val_loss: 1.0834 - val_regressor_head_loss: 0.0248 - val_regressor_head_ms
e: 0.0248
Epoch 23/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 213ms/step - classifier_head_accuracy: 0.9627 - cla
ssifier_head_loss: 0.0936 - loss: 0.1125 - regressor_head_loss: 0.0189 - regressor_h
ead_mse: 0.0189 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
1.1600 - val_loss: 1.1850 - val_regressor_head_loss: 0.0250 - val_regressor_head_ms
e: 0.0250
Epoch 24/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 214ms/step - classifier_head_accuracy: 0.9596 - cla
ssifier_head_loss: 0.0967 - loss: 0.1159 - regressor_head_loss: 0.0192 - regressor_h
ead_mse: 0.0192 - val_classifier_head_accuracy: 0.7200 - val_classifier_head_loss:
1.0390 - val_loss: 1.0637 - val_regressor_head_loss: 0.0246 - val_regressor_head_ms
e: 0.0246
Epoch 25/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 211ms/step - classifier_head_accuracy: 0.9560 - cla
ssifier_head_loss: 0.0985 - loss: 0.1170 - regressor_head_loss: 0.0185 - regressor_h
ead_mse: 0.0185 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
0.9505 - val_loss: 0.9728 - val_regressor_head_loss: 0.0223 - val_regressor_head_ms
e: 0.0223
Epoch 26/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 211ms/step - classifier_head_accuracy: 0.9680 - cla
ssifier_head_loss: 0.0797 - loss: 0.0974 - regressor_head_loss: 0.0177 - regressor_h
ead_mse: 0.0177 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_loss:
0.6768 - val_loss: 0.6943 - val_regressor_head_loss: 0.0175 - val_regressor_head_ms
e: 0.0175
Epoch 27/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 211ms/step - classifier_head_accuracy: 0.9669 - cla
ssifier_head_loss: 0.0795 - loss: 0.0978 - regressor_head_loss: 0.0183 - regressor_h
ead_mse: 0.0183 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.0299 - val_loss: 1.0523 - val_regressor_head_loss: 0.0224 - val_regressor_head_ms
e: 0.0224
Epoch 28/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 211ms/step - classifier_head_accuracy: 0.9759 - cla
ssifier_head_loss: 0.0623 - loss: 0.0804 - regressor_head_loss: 0.0181 - regressor_h
ead_mse: 0.0181 - val_classifier_head_accuracy: 0.7143 - val_classifier_head_loss:
1.5558 - val_loss: 1.5805 - val_regressor_head_loss: 0.0247 - val_regressor_head_ms
e: 0.0247
```

```
Epoch 29/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 211ms/step - classifier_head_accuracy: 0.9766 - cla
ssifier_head_loss: 0.0607 - loss: 0.0799 - regressor_head_loss: 0.0192 - regressor_h
ead_mse: 0.0192 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.3288 - val_loss: 1.3517 - val_regressor_head_loss: 0.0229 - val_regressor_head_ms
e: 0.0229
Epoch 30/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 213ms/step - classifier_head_accuracy: 0.9741 - cla
ssifier_head_loss: 0.0652 - loss: 0.0839 - regressor_head_loss: 0.0187 - regressor_h
ead_mse: 0.0187 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_loss:
1.0911 - val_loss: 1.1119 - val_regressor_head_loss: 0.0208 - val_regressor_head_ms
e: 0.0208
Epoch 31/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9631 - cla
ssifier_head_loss: 0.0898 - loss: 0.1093 - regressor_head_loss: 0.0195 - regressor_h
ead_mse: 0.0195 - val_classifier_head_accuracy: 0.7086 - val_classifier_head_loss:
1.7324 - val_loss: 1.7611 - val_regressor_head_loss: 0.0286 - val_regressor_head_ms
e: 0.0286
Epoch 32/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9743 - cla
ssifier_head_loss: 0.0616 - loss: 0.0796 - regressor_head_loss: 0.0179 - regressor_h
ead_mse: 0.0179 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.1491 - val_loss: 1.1705 - val_regressor_head_loss: 0.0215 - val_regressor_head_ms
e: 0.0215
Epoch 33/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9807 - cla
ssifier_head_loss: 0.0486 - loss: 0.0661 - regressor_head_loss: 0.0175 - regressor_h
ead_mse: 0.0175 - val_classifier_head_accuracy: 0.7886 - val_classifier_head_loss:
1.1015 - val_loss: 1.1209 - val_regressor_head_loss: 0.0194 - val_regressor_head_ms
e: 0.0194
Epoch 34/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9819 - cla
ssifier_head_loss: 0.0477 - loss: 0.0654 - regressor_head_loss: 0.0178 - regressor_h
ead_mse: 0.0178 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_loss:
1.3595 - val_loss: 1.3809 - val_regressor_head_loss: 0.0214 - val_regressor_head_ms
e: 0.0214
Epoch 35/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9810 - cla
ssifier_head_loss: 0.0543 - loss: 0.0727 - regressor_head_loss: 0.0183 - regressor_h
ead_mse: 0.0183 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
1.5935 - val_loss: 1.6174 - val_regressor_head_loss: 0.0239 - val_regressor_head_ms
e: 0.0239
Epoch 36/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9782 - cla
ssifier_head_loss: 0.0566 - loss: 0.0748 - regressor_head_loss: 0.0182 - regressor_h
ead_mse: 0.0182 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
1.4251 - val_loss: 1.4471 - val_regressor_head_loss: 0.0220 - val_regressor_head_ms
e: 0.0220
Epoch 37/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 39s 211ms/step - classifier_head_accuracy: 0.9774 - cla
ssifier_head_loss: 0.0500 - loss: 0.0680 - regressor_head_loss: 0.0180 - regressor_h
ead_mse: 0.0180 - val_classifier_head_accuracy: 0.7543 - val_classifier_head_loss:
1.5222 - val_loss: 1.5457 - val_regressor_head_loss: 0.0235 - val_regressor_head_ms
e: 0.0235
Epoch 38/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 214ms/step - classifier_head_accuracy: 0.9872 - cla
```

```
ssifier_head_loss: 0.0369 - loss: 0.0548 - regressor_head_loss: 0.0179 - regressor_h
ead_mse: 0.0179 - val_classifier_head_accuracy: 0.7543 - val_classifier_head_loss:
1.4985 - val_loss: 1.5231 - val_regressor_head_loss: 0.0246 - val_regressor_head_ms
e: 0.0246
Epoch 39/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9837 - cla
ssifier_head_loss: 0.0395 - loss: 0.0563 - regressor_head_loss: 0.0168 - regressor_h
ead_mse: 0.0168 - val_classifier_head_accuracy: 0.7086 - val_classifier_head_loss:
1.6551 - val_loss: 1.6777 - val_regressor_head_loss: 0.0227 - val_regressor_head_ms
e: 0.0227
Epoch 40/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 214ms/step - classifier_head_accuracy: 0.9881 - cla
ssifier_head_loss: 0.0318 - loss: 0.0485 - regressor_head_loss: 0.0167 - regressor_h
ead_mse: 0.0167 - val_classifier_head_accuracy: 0.6914 - val_classifier_head_loss:
1.9072 - val_loss: 1.9326 - val_regressor_head_loss: 0.0255 - val_regressor_head_ms
e: 0.0255
Epoch 41/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 215ms/step - classifier_head_accuracy: 0.9887 - cla
ssifier_head_loss: 0.0313 - loss: 0.0483 - regressor_head_loss: 0.0170 - regressor_h
ead_mse: 0.0170 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.5979 - val_loss: 1.6225 - val_regressor_head_loss: 0.0246 - val_regressor_head_ms
e: 0.0246
Epoch 42/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 217ms/step - classifier_head_accuracy: 0.9848 - cla
ssifier_head_loss: 0.0409 - loss: 0.0588 - regressor_head_loss: 0.0179 - regressor_h
ead_mse: 0.0179 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
1.7820 - val_loss: 1.8067 - val_regressor_head_loss: 0.0247 - val_regressor_head_ms
e: 0.0247
Epoch 43/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 215ms/step - classifier_head_accuracy: 0.9843 - cla
ssifier_head_loss: 0.0400 - loss: 0.0577 - regressor_head_loss: 0.0177 - regressor_h
ead_mse: 0.0177 - val_classifier_head_accuracy: 0.7771 - val_classifier_head_loss:
1.5309 - val_loss: 1.5536 - val_regressor_head_loss: 0.0227 - val_regressor_head_ms
e: 0.0227
Epoch 44/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 213ms/step - classifier_head_accuracy: 0.9843 - cla
ssifier_head_loss: 0.0430 - loss: 0.0609 - regressor_head_loss: 0.0179 - regressor_h
ead_mse: 0.0179 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
1.7636 - val_loss: 1.7875 - val_regressor_head_loss: 0.0239 - val_regressor_head_ms
e: 0.0239
Epoch 45/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 213ms/step - classifier_head_accuracy: 0.9856 - cla
ssifier_head_loss: 0.0342 - loss: 0.0512 - regressor_head_loss: 0.0170 - regressor_h
ead_mse: 0.0170 - val_classifier_head_accuracy: 0.7371 - val_classifier_head_loss:
1.5928 - val_loss: 1.6188 - val_regressor_head_loss: 0.0260 - val_regressor_head_ms
e: 0.0260
Epoch 46/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 213ms/step - classifier_head_accuracy: 0.9832 - cla
ssifier_head_loss: 0.0459 - loss: 0.0623 - regressor_head_loss: 0.0164 - regressor_h
ead_mse: 0.0164 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.2363 - val_loss: 1.2575 - val_regressor_head_loss: 0.0212 - val_regressor_head_ms
e: 0.0212
Epoch 47/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9807 - cla
ssifier_head_loss: 0.0500 - loss: 0.0670 - regressor_head_loss: 0.0171 - regressor_h
ead_mse: 0.0171 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
```

```
1.5888 - val_loss: 1.6130 - val_regressor_head_loss: 0.0243 - val_regressor_head_ms
e: 0.0243
Epoch 48/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 214ms/step - classifier_head_accuracy: 0.9861 - cla
ssifier_head_loss: 0.0384 - loss: 0.0559 - regressor_head_loss: 0.0175 - regressor_h
ead_mse: 0.0175 - val_classifier_head_accuracy: 0.6914 - val_classifier_head_loss:
2.4200 - val_loss: 2.4500 - val_regressor_head_loss: 0.0300 - val_regressor_head_ms
e: 0.0300
Epoch 49/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 217ms/step - classifier_head_accuracy: 0.9878 - cla
ssifier_head_loss: 0.0309 - loss: 0.0482 - regressor_head_loss: 0.0173 - regressor_h
ead_mse: 0.0173 - val_classifier_head_accuracy: 0.7600 - val_classifier_head_loss:
1.4386 - val_loss: 1.4632 - val_regressor_head_loss: 0.0246 - val_regressor_head_ms
e: 0.0246
Epoch 50/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 212ms/step - classifier_head_accuracy: 0.9832 - cla
ssifier_head_loss: 0.0470 - loss: 0.0647 - regressor_head_loss: 0.0177 - regressor_h
ead_mse: 0.0177 - val_classifier_head_accuracy: 0.7314 - val_classifier_head_loss:
1.7721 - val_loss: 1.7955 - val_regressor_head_loss: 0.0233 - val_regressor_head_ms
e: 0.0233
Epoch 51/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 213ms/step - classifier_head_accuracy: 0.9903 - cla
ssifier_head_loss: 0.0230 - loss: 0.0392 - regressor_head_loss: 0.0162 - regressor_h
ead_mse: 0.0162 - val_classifier_head_accuracy: 0.6629 - val_classifier_head_loss:
2.1473 - val_loss: 2.1723 - val_regressor_head_loss: 0.0250 - val_regressor_head_ms
e: 0.0250
Epoch 52/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 213ms/step - classifier_head_accuracy: 0.9843 - cla
ssifier_head_loss: 0.0364 - loss: 0.0528 - regressor_head_loss: 0.0164 - regressor_h
ead_mse: 0.0164 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.6558 - val_loss: 1.6782 - val_regressor_head_loss: 0.0225 - val_regressor_head_ms
e: 0.0225
Epoch 53/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 214ms/step - classifier_head_accuracy: 0.9881 - cla
ssifier_head_loss: 0.0363 - loss: 0.0524 - regressor_head_loss: 0.0161 - regressor_h
ead_mse: 0.0161 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.4727 - val_loss: 1.4945 - val_regressor_head_loss: 0.0218 - val_regressor_head_ms
e: 0.0218
Epoch 54/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 215ms/step - classifier_head_accuracy: 0.9892 - cla
ssifier_head_loss: 0.0312 - loss: 0.0480 - regressor_head_loss: 0.0168 - regressor_h
ead_mse: 0.0168 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
1.9103 - val_loss: 1.9365 - val_regressor_head_loss: 0.0262 - val_regressor_head_ms
e: 0.0262
Epoch 55/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 40s 213ms/step - classifier_head_accuracy: 0.9899 - cla
ssifier_head_loss: 0.0235 - loss: 0.0403 - regressor_head_loss: 0.0168 - regressor_h
ead_mse: 0.0168 - val_classifier_head_accuracy: 0.7086 - val_classifier_head_loss:
2.0845 - val_loss: 2.1087 - val_regressor_head_loss: 0.0242 - val_regressor_head_ms
e: 0.0242
Epoch 56/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 217ms/step - classifier_head_accuracy: 0.9876 - cla
ssifier_head_loss: 0.0321 - loss: 0.0490 - regressor_head_loss: 0.0169 - regressor_h
ead_mse: 0.0169 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
1.2885 - val_loss: 1.3101 - val_regressor_head_loss: 0.0216 - val_regressor_head_ms
e: 0.0216
```

```
Epoch 57/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 219ms/step - classifier_head_accuracy: 0.9874 - cla
ssifier_head_loss: 0.0369 - loss: 0.0536 - regressor_head_loss: 0.0168 - regressor_h
ead_mse: 0.0168 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_loss:
1.4535 - val_loss: 1.4765 - val_regressor_head_loss: 0.0230 - val_regressor_head_ms
e: 0.0230
Epoch 58/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 219ms/step - classifier_head_accuracy: 0.9857 - cla
ssifier_head_loss: 0.0352 - loss: 0.0522 - regressor_head_loss: 0.0169 - regressor_h
ead_mse: 0.0169 - val_classifier_head_accuracy: 0.7543 - val_classifier_head_loss:
1.8257 - val_loss: 1.8501 - val_regressor_head_loss: 0.0243 - val_regressor_head_ms
e: 0.0243
Epoch 59/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 218ms/step - classifier_head_accuracy: 0.9882 - cla
ssifier_head_loss: 0.0289 - loss: 0.0442 - regressor_head_loss: 0.0153 - regressor_h
ead_mse: 0.0153 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_loss:
1.5359 - val_loss: 1.5569 - val_regressor_head_loss: 0.0210 - val_regressor_head_ms
e: 0.0210
Epoch 60/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 219ms/step - classifier_head_accuracy: 0.9870 - cla
ssifier_head_loss: 0.0336 - loss: 0.0500 - regressor_head_loss: 0.0164 - regressor_h
ead_mse: 0.0164 - val_classifier_head_accuracy: 0.7371 - val_classifier_head_loss:
2.0457 - val_loss: 2.0709 - val_regressor_head_loss: 0.0251 - val_regressor_head_ms
e: 0.0251
Epoch 61/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 220ms/step - classifier_head_accuracy: 0.9878 - cla
ssifier_head_loss: 0.0307 - loss: 0.0475 - regressor_head_loss: 0.0168 - regressor_h
ead_mse: 0.0168 - val_classifier_head_accuracy: 0.7886 - val_classifier_head_loss:
1.6897 - val_loss: 1.7127 - val_regressor_head_loss: 0.0230 - val_regressor_head_ms
e: 0.0230
Epoch 62/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 220ms/step - classifier_head_accuracy: 0.9925 - cla
ssifier_head_loss: 0.0196 - loss: 0.0364 - regressor_head_loss: 0.0168 - regressor_h
ead_mse: 0.0168 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
1.8872 - val_loss: 1.9099 - val_regressor_head_loss: 0.0227 - val_regressor_head_ms
e: 0.0227
Epoch 63/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 220ms/step - classifier_head_accuracy: 0.9900 - cla
ssifier_head_loss: 0.0268 - loss: 0.0432 - regressor_head_loss: 0.0164 - regressor_h
ead_mse: 0.0164 - val_classifier_head_accuracy: 0.7257 - val_classifier_head_loss:
2.0900 - val_loss: 2.1133 - val_regressor_head_loss: 0.0233 - val_regressor_head_ms
e: 0.0233
Epoch 64/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 221ms/step - classifier_head_accuracy: 0.9883 - cla
ssifier_head_loss: 0.0281 - loss: 0.0447 - regressor_head_loss: 0.0166 - regressor_h
ead_mse: 0.0166 - val_classifier_head_accuracy: 0.7886 - val_classifier_head_loss:
1.5127 - val_loss: 1.5365 - val_regressor_head_loss: 0.0238 - val_regressor_head_ms
e: 0.0238
Epoch 65/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 222ms/step - classifier_head_accuracy: 0.9866 - cla
ssifier_head_loss: 0.0361 - loss: 0.0530 - regressor_head_loss: 0.0168 - regressor_h
ead_mse: 0.0168 - val_classifier_head_accuracy: 0.7600 - val_classifier_head_loss:
1.6834 - val_loss: 1.7091 - val_regressor_head_loss: 0.0257 - val_regressor_head_ms
e: 0.0257
Epoch 66/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 223ms/step - classifier_head_accuracy: 0.9841 - cla
```

```
ssifier_head_loss: 0.0440 - loss: 0.0602 - regressor_head_loss: 0.0162 - regressor_h
ead_mse: 0.0162 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.6038 - val_loss: 1.6273 - val_regressor_head_loss: 0.0235 - val_regressor_head_ms
e: 0.0235
Epoch 67/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 221ms/step - classifier_head_accuracy: 0.9910 - cla
ssifier_head_loss: 0.0218 - loss: 0.0375 - regressor_head_loss: 0.0157 - regressor_h
ead_mse: 0.0157 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_loss:
1.7083 - val_loss: 1.7297 - val_regressor_head_loss: 0.0213 - val_regressor_head_ms
e: 0.0213
Epoch 68/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 223ms/step - classifier_head_accuracy: 0.9914 - cla
ssifier_head_loss: 0.0216 - loss: 0.0374 - regressor_head_loss: 0.0158 - regressor_h
ead_mse: 0.0158 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_loss:
1.1383 - val_loss: 1.1571 - val_regressor_head_loss: 0.0188 - val_regressor_head_ms
e: 0.0188
Epoch 69/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 221ms/step - classifier_head_accuracy: 0.9857 - cla
ssifier_head_loss: 0.0404 - loss: 0.0572 - regressor_head_loss: 0.0168 - regressor_h
ead_mse: 0.0168 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
1.9426 - val_loss: 1.9648 - val_regressor_head_loss: 0.0222 - val_regressor_head_ms
e: 0.0222
Epoch 70/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 219ms/step - classifier_head_accuracy: 0.9905 - cla
ssifier_head_loss: 0.0259 - loss: 0.0424 - regressor_head_loss: 0.0165 - regressor_h
ead_mse: 0.0165 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.7696 - val_loss: 1.7928 - val_regressor_head_loss: 0.0233 - val_regressor_head_ms
e: 0.0233
Epoch 71/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 222ms/step - classifier_head_accuracy: 0.9931 - cla
ssifier_head_loss: 0.0162 - loss: 0.0320 - regressor_head_loss: 0.0158 - regressor_h
ead_mse: 0.0158 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.7214 - val_loss: 1.7413 - val_regressor_head_loss: 0.0199 - val_regressor_head_ms
e: 0.0199
Epoch 72/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 221ms/step - classifier_head_accuracy: 0.9896 - cla
ssifier_head_loss: 0.0256 - loss: 0.0410 - regressor_head_loss: 0.0153 - regressor_h
ead_mse: 0.0153 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_loss:
1.5728 - val_loss: 1.5931 - val_regressor_head_loss: 0.0204 - val_regressor_head_ms
e: 0.0204
Epoch 73/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 221ms/step - classifier_head_accuracy: 0.9892 - cla
ssifier_head_loss: 0.0294 - loss: 0.0450 - regressor_head_loss: 0.0156 - regressor_h
ead_mse: 0.0156 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.4767 - val_loss: 1.4969 - val_regressor_head_loss: 0.0201 - val_regressor_head_ms
e: 0.0201
Epoch 74/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 222ms/step - classifier_head_accuracy: 0.9909 - cla
ssifier_head_loss: 0.0263 - loss: 0.0425 - regressor_head_loss: 0.0162 - regressor_h
ead_mse: 0.0162 - val_classifier_head_accuracy: 0.7771 - val_classifier_head_loss:
1.7613 - val_loss: 1.7829 - val_regressor_head_loss: 0.0216 - val_regressor_head_ms
e: 0.0216
Epoch 75/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 222ms/step - classifier_head_accuracy: 0.9907 - cla
ssifier_head_loss: 0.0237 - loss: 0.0400 - regressor_head_loss: 0.0163 - regressor_h
ead_mse: 0.0163 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_loss:
```

```
1.5534 - val_loss: 1.5750 - val_regressor_head_loss: 0.0217 - val_regressor_head_ms
e: 0.0217
Epoch 76/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 223ms/step - classifier_head_accuracy: 0.9913 - cla
ssifier_head_loss: 0.0210 - loss: 0.0371 - regressor_head_loss: 0.0161 - regressor_h
ead_mse: 0.0161 - val_classifier_head_accuracy: 0.7771 - val_classifier_head_loss:
1.8577 - val_loss: 1.8808 - val_regressor_head_loss: 0.0231 - val_regressor_head_ms
e: 0.0231
Epoch 77/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 224ms/step - classifier_head_accuracy: 0.9902 - cla
ssifier_head_loss: 0.0259 - loss: 0.0423 - regressor_head_loss: 0.0165 - regressor_h
ead_mse: 0.0165 - val_classifier_head_accuracy: 0.6857 - val_classifier_head_loss:
2.7404 - val_loss: 2.7652 - val_regressor_head_loss: 0.0248 - val_regressor_head_ms
e: 0.0248
Epoch 78/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 225ms/step - classifier_head_accuracy: 0.9881 - cla
ssifier_head_loss: 0.0227 - loss: 0.0380 - regressor_head_loss: 0.0153 - regressor_h
ead_mse: 0.0153 - val_classifier_head_accuracy: 0.7257 - val_classifier_head_loss:
2.2765 - val_loss: 2.3002 - val_regressor_head_loss: 0.0237 - val_regressor_head_ms
e: 0.0237
Epoch 79/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 223ms/step - classifier_head_accuracy: 0.9898 - cla
ssifier_head_loss: 0.0231 - loss: 0.0378 - regressor_head_loss: 0.0146 - regressor_h
ead_mse: 0.0146 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.4309 - val_loss: 1.4506 - val_regressor_head_loss: 0.0198 - val_regressor_head_ms
e: 0.0198
Epoch 80/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 224ms/step - classifier_head_accuracy: 0.9901 - cla
ssifier_head_loss: 0.0279 - loss: 0.0434 - regressor_head_loss: 0.0155 - regressor_h
ead_mse: 0.0155 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.4946 - val_loss: 1.5150 - val_regressor_head_loss: 0.0204 - val_regressor_head_ms
e: 0.0204
Epoch 81/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 225ms/step - classifier_head_accuracy: 0.9901 - cla
ssifier_head_loss: 0.0233 - loss: 0.0387 - regressor_head_loss: 0.0154 - regressor_h
ead_mse: 0.0154 - val_classifier_head_accuracy: 0.7086 - val_classifier_head_loss:
2.0832 - val_loss: 2.1091 - val_regressor_head_loss: 0.0259 - val_regressor_head_ms
e: 0.0259
Epoch 82/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 225ms/step - classifier_head_accuracy: 0.9916 - cla
ssifier_head_loss: 0.0202 - loss: 0.0359 - regressor_head_loss: 0.0157 - regressor_h
ead_mse: 0.0157 - val_classifier_head_accuracy: 0.7771 - val_classifier_head_loss:
1.6226 - val_loss: 1.6419 - val_regressor_head_loss: 0.0192 - val_regressor_head_ms
e: 0.0192
Epoch 83/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 225ms/step - classifier_head_accuracy: 0.9929 - cla
ssifier_head_loss: 0.0178 - loss: 0.0331 - regressor_head_loss: 0.0154 - regressor_h
ead_mse: 0.0154 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.6744 - val_loss: 1.6945 - val_regressor_head_loss: 0.0201 - val_regressor_head_ms
e: 0.0201
Epoch 84/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 43s 230ms/step - classifier_head_accuracy: 0.9912 - cla
ssifier_head_loss: 0.0171 - loss: 0.0321 - regressor_head_loss: 0.0150 - regressor_h
ead_mse: 0.0150 - val_classifier_head_accuracy: 0.7371 - val_classifier_head_loss:
2.3194 - val_loss: 2.3408 - val_regressor_head_loss: 0.0214 - val_regressor_head_ms
e: 0.0214
```

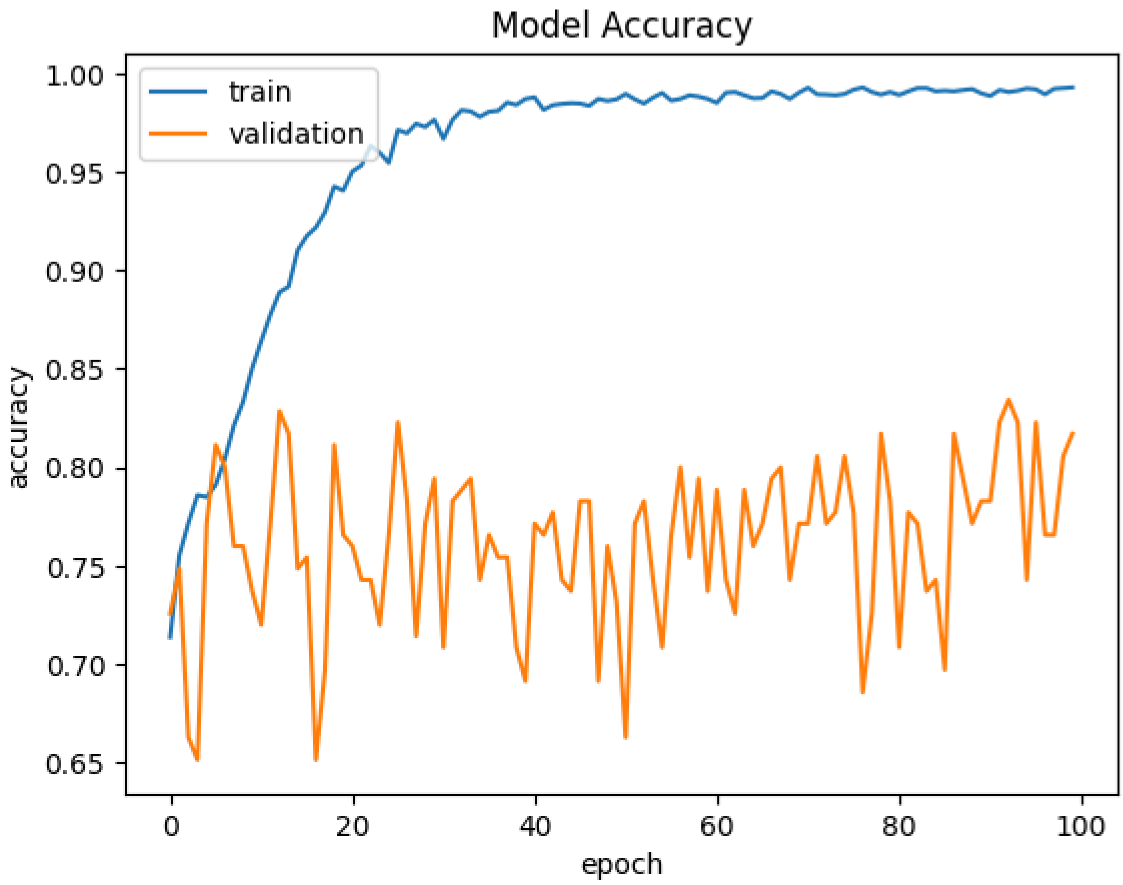
```
Epoch 85/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 225ms/step - classifier_head_accuracy: 0.9897 - cla
ssifier_head_loss: 0.0215 - loss: 0.0366 - regressor_head_loss: 0.0151 - regressor_h
ead_mse: 0.0151 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
1.8041 - val_loss: 1.8260 - val_regressor_head_loss: 0.0218 - val_regressor_head_ms
e: 0.0218
Epoch 86/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 41s 221ms/step - classifier_head_accuracy: 0.9903 - cla
ssifier_head_loss: 0.0201 - loss: 0.0344 - regressor_head_loss: 0.0143 - regressor_h
ead_mse: 0.0143 - val_classifier_head_accuracy: 0.6971 - val_classifier_head_loss:
2.3628 - val_loss: 2.3851 - val_regressor_head_loss: 0.0223 - val_regressor_head_ms
e: 0.0223
Epoch 87/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 222ms/step - classifier_head_accuracy: 0.9883 - cla
ssifier_head_loss: 0.0404 - loss: 0.0559 - regressor_head_loss: 0.0155 - regressor_h
ead_mse: 0.0155 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.5490 - val_loss: 1.5697 - val_regressor_head_loss: 0.0207 - val_regressor_head_ms
e: 0.0207
Epoch 88/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 223ms/step - classifier_head_accuracy: 0.9914 - cla
ssifier_head_loss: 0.0225 - loss: 0.0381 - regressor_head_loss: 0.0157 - regressor_h
ead_mse: 0.0157 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_loss:
1.6218 - val_loss: 1.6419 - val_regressor_head_loss: 0.0201 - val_regressor_head_ms
e: 0.0201
Epoch 89/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 223ms/step - classifier_head_accuracy: 0.9943 - cla
ssifier_head_loss: 0.0140 - loss: 0.0293 - regressor_head_loss: 0.0154 - regressor_h
ead_mse: 0.0154 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.8369 - val_loss: 1.8575 - val_regressor_head_loss: 0.0206 - val_regressor_head_ms
e: 0.0206
Epoch 90/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 223ms/step - classifier_head_accuracy: 0.9872 - cla
ssifier_head_loss: 0.0284 - loss: 0.0444 - regressor_head_loss: 0.0160 - regressor_h
ead_mse: 0.0160 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.9429 - val_loss: 1.9614 - val_regressor_head_loss: 0.0185 - val_regressor_head_ms
e: 0.0185
Epoch 91/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 223ms/step - classifier_head_accuracy: 0.9893 - cla
ssifier_head_loss: 0.0236 - loss: 0.0393 - regressor_head_loss: 0.0156 - regressor_h
ead_mse: 0.0156 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.7864 - val_loss: 1.8084 - val_regressor_head_loss: 0.0220 - val_regressor_head_ms
e: 0.0220
Epoch 92/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 224ms/step - classifier_head_accuracy: 0.9908 - cla
ssifier_head_loss: 0.0204 - loss: 0.0354 - regressor_head_loss: 0.0150 - regressor_h
ead_mse: 0.0150 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_loss:
1.4334 - val_loss: 1.4527 - val_regressor_head_loss: 0.0193 - val_regressor_head_ms
e: 0.0193
Epoch 93/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 224ms/step - classifier_head_accuracy: 0.9905 - cla
ssifier_head_loss: 0.0222 - loss: 0.0366 - regressor_head_loss: 0.0144 - regressor_h
ead_mse: 0.0144 - val_classifier_head_accuracy: 0.8343 - val_classifier_head_loss:
1.1606 - val_loss: 1.1783 - val_regressor_head_loss: 0.0177 - val_regressor_head_ms
e: 0.0177
Epoch 94/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 224ms/step - classifier_head_accuracy: 0.9914 - cla
```

```
ssifier_head_loss: 0.0207 - loss: 0.0358 - regressor_head_loss: 0.0151 - regressor_h
ead_mse: 0.0151 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_loss:
1.6305 - val_loss: 1.6514 - val_regressor_head_loss: 0.0209 - val_regressor_head_ms
e: 0.0209
Epoch 95/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 224ms/step - classifier_head_accuracy: 0.9948 - cla
ssifier_head_loss: 0.0164 - loss: 0.0317 - regressor_head_loss: 0.0152 - regressor_h
ead_mse: 0.0152 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
2.1975 - val_loss: 2.2191 - val_regressor_head_loss: 0.0215 - val_regressor_head_ms
e: 0.0215
Epoch 96/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 226ms/step - classifier_head_accuracy: 0.9921 - cla
ssifier_head_loss: 0.0184 - loss: 0.0339 - regressor_head_loss: 0.0155 - regressor_h
ead_mse: 0.0155 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_loss:
1.6465 - val_loss: 1.6665 - val_regressor_head_loss: 0.0200 - val_regressor_head_ms
e: 0.0200
Epoch 97/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 224ms/step - classifier_head_accuracy: 0.9860 - cla
ssifier_head_loss: 0.0343 - loss: 0.0502 - regressor_head_loss: 0.0159 - regressor_h
ead_mse: 0.0159 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
1.7489 - val_loss: 1.7703 - val_regressor_head_loss: 0.0214 - val_regressor_head_ms
e: 0.0214
Epoch 98/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 42s 226ms/step - classifier_head_accuracy: 0.9928 - cla
ssifier_head_loss: 0.0166 - loss: 0.0311 - regressor_head_loss: 0.0145 - regressor_h
ead_mse: 0.0145 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
1.7484 - val_loss: 1.7706 - val_regressor_head_loss: 0.0222 - val_regressor_head_ms
e: 0.0222
Epoch 99/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 43s 228ms/step - classifier_head_accuracy: 0.9908 - cla
ssifier_head_loss: 0.0210 - loss: 0.0353 - regressor_head_loss: 0.0142 - regressor_h
ead_mse: 0.0142 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_loss:
1.7498 - val_loss: 1.7687 - val_regressor_head_loss: 0.0189 - val_regressor_head_ms
e: 0.0189
Epoch 100/100
187/187 ━━━━━━━━━━━━━━━━━━━━ 46s 247ms/step - classifier_head_accuracy: 0.9934 - cla
ssifier_head_loss: 0.0141 - loss: 0.0279 - regressor_head_loss: 0.0138 - regressor_h
ead_mse: 0.0138 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.4322 - val_loss: 1.4530 - val_regressor_head_loss: 0.0208 - val_regressor_head_ms
e: 0.0208
```

Visualizing the training and validation accuracy of your classification model across epochs.

In [18]:

```
plt.plot(history.history['classifier_head_accuracy'])
plt.plot(history.history['val_classifier_head_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



Defining function to calculate IoU.

In [21]:

```
def intersection_over_union(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[0] + boxA[2], boxB[0] + boxB[2])
    yB = min(boxA[1] + boxA[3], boxB[1] + boxB[3])

    interWidth = max(0, xB - xA)
    interHeight = max(0, yB - yA)
    interArea = interWidth * interHeight

    boxAArea = boxA[2] * boxA[3]
    boxBArea = boxB[2] * boxB[3]


    if boxAArea == 0 or boxBArea == 0:
        return 0.0


    iou = interArea / float(boxAArea + boxBArea - interArea)
    return iou
```

Defining function to prepare the test data set for testing the model.

In [19]:

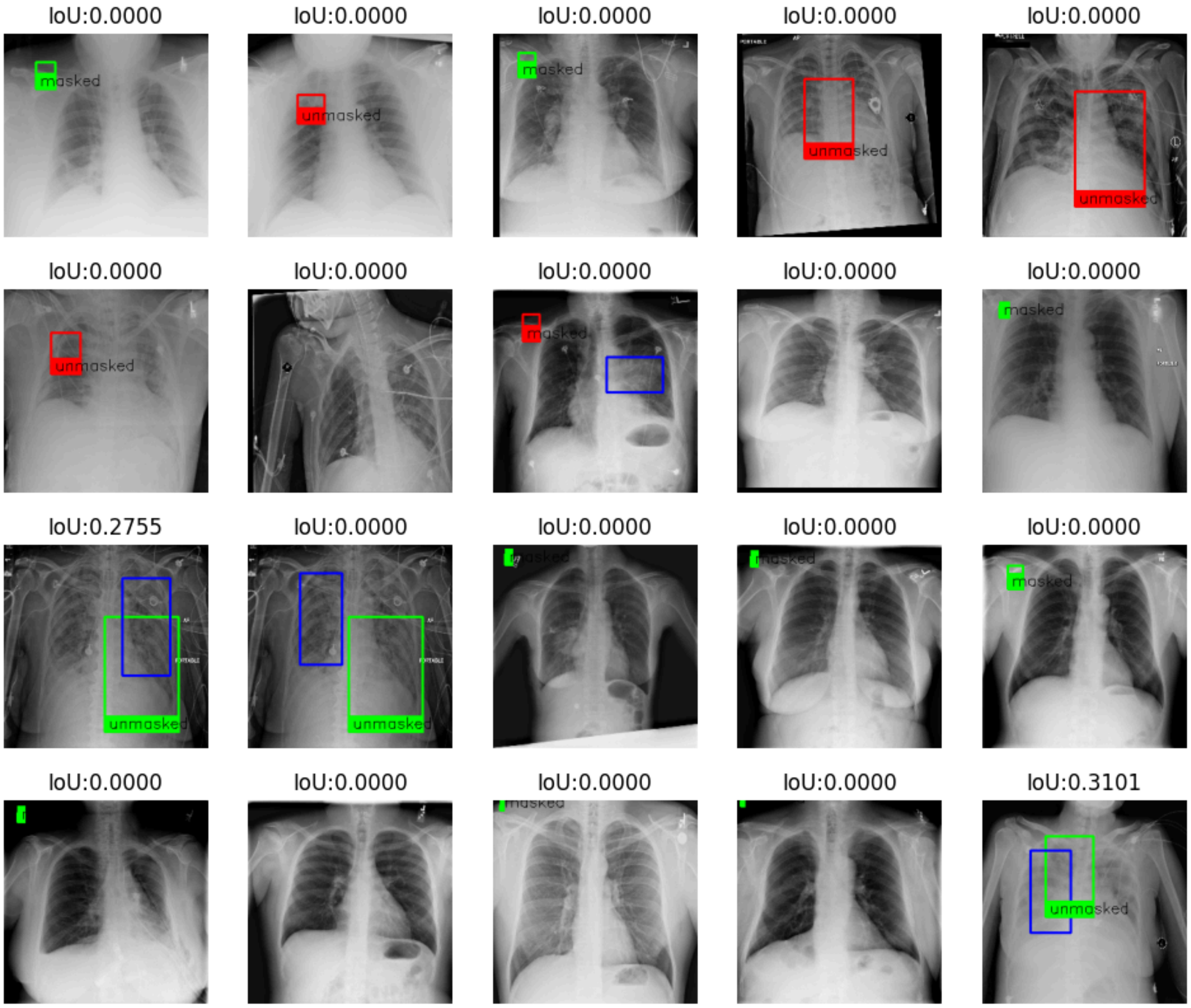
```
def tune_test_ds(dataset):
    dataset = dataset.map(format_instance, num_parallel_calls=tf.data.AUTOTUNE)
    dataset = dataset.batch(1)
    dataset = dataset.repeat()
    return dataset

test_ds = tune_test_ds(raw_test_ds)
```

Visualizing test predicitons for the model along with the calculated IoU.

In [22]:

```
plt.figure(figsize=(12, 10))

test_list = list(test_ds.take(20).as_numpy_iterator())

print(len(test_list))

image, labels = test_list[0]

for i in range(len(test_list)):

    ax = plt.subplot(4, 5, i + 1)
    image, labels = test_list[i]

    predictions = model(image)

    predicted_box = predictions[1][0] * input_size
    predicted_box = tf.cast(predicted_box, tf.int32)

    predicted_label = predictions[0][0]

    image = image[0]

    actual_label = labels[0][0]
    actual_box = labels[1][0] * input_size
    actual_box = tf.cast(actual_box, tf.int32)

    image = image.astype("float") * 255.0
    image = image.astype(np.uint8)
    image_color = cv.cvtColor(image, cv.COLOR_GRAY2RGB)

    color = (255, 0, 0)
    # print box red if predicted and actual label do not match
    if (predicted_label[0] > 0.5 and actual_label[0] > 0) or (predicted_label[0] <
        color = (0, 255, 0)

    img_label = "unmasked"
    if predicted_label[0] > 0.5:
        img_label = "masked"

    predicted_box_n = predicted_box.numpy()
    cv.rectangle(image_color, predicted_box_n, color, 2)
    cv.rectangle(image_color, actual_box.numpy(), (0, 0, 255), 2)
    cv.rectangle(image_color, (predicted_box_n[0], predicted_box_n[1] + predicted_b
    cv.putText(image_color, img_label, (predicted_box_n[0] + 5, predicted_box_n[1]

    IoU = intersection_over_union(predicted_box.numpy(), actual_box.numpy())
    plt.title("IoU:" + format(IoU, '.4f'))
    plt.imshow(image_color)
    plt.axis("off")
```

```
20
```



A detailed evaluation pipeline to assess your trained model's predictions on 200 test samples, visualize bounding boxes, and calculate accuracy & Intersection over Union (IoU).

In [23]:

```
output_dir = "output_predictions"
os.makedirs(output_dir, exist_ok=True)

plt.figure(figsize=(12, 10))

test_list = list(test_ds.take(200).as_numpy_iterator())
print(f"Test Data Size: {len(test_list)}")

correct_count = 0
total_count = 0
iou_list = []

for i in range(len(test_list)):

    image, labels = test_list[i]
    predictions = model(image)
    predicted_box = predictions[1][0] * input_size
    predicted_box = tf.cast(predicted_box, tf.int32)
    predicted_label = predictions[0][0]

    image = image[0]
    actual_label = labels[0][0]
    actual_box = labels[1][0] * input_size
    actual_box = tf.cast(actual_box, tf.int32)

    image = image.astype("float") * 255.0
    image = image.astype(np.uint8)
    image_color = cv.cvtColor(image, cv.COLOR_GRAY2RGB)

    color = (255, 0, 0)
    if (predicted_label[0] > 0.5 and actual_label[0] > 0) or (predicted_label[0] <
        color = (0, 255, 0)
        correct_count += 1

    total_count += 1

    img_label = "unmasked"
    if predicted_label[0] > 0.5:
        img_label = "masked"

    predicted_box_n = predicted_box.numpy()
    cv.rectangle(image_color, predicted_box_n, color, 2)
    cv.rectangle(image_color, actual_box.numpy(), (0, 0, 255), 2)
    cv.rectangle(image_color, (predicted_box_n[0], predicted_box_n[1] + predicted_b
                 (predicted_box_n[0] + predicted_box_n[2], predicted_box_n[1] + pre
    cv.putText(image_color, img_label, (predicted_box_n[0] + 5, predicted_box_n[1]
               cv.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0))

    #IoU
    IoU = intersection_over_union(predicted_box.numpy(), actual_box.numpy())
    iou_list.append(IoU)

    output_path = os.path.join(output_dir, f"prediction_{i + 1}.png")
    cv.imwrite(output_path, cv.cvtColor(image_color, cv.COLOR_RGB2BGR))

accuracy = correct_count / total_count
average_iou = np.mean(iou_list)

print(f"Accuracy: {accuracy:.4f}")
print(f"Mean IoU: {average_iou:.4f}")

plt.savefig(os.path.join(output_dir, "all_predictions.png"))
plt.show()
```

```
Test Data Size: 200
Accuracy: 0.7550
Mean IoU: 0.0198
<Figure size 1200x1000 with 0 Axes>
```