Importing relevant libaries and packages.

```
In [85]: import numpy as np
         import pandas as pd
         import os
         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 graphviz
         import tensorflow as tf
         from tensorflow.keras.utils import plot_model
         from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

Load labels and display first 5 rows. Notice how the first 4 patients did not have pneumonia and do not have a box attahed, while the last patient has pneumonia and has a box.

```
In [2]: train_labels = pd.read_csv('rsna-pneumonia-detection-challenge\stage_2_train_labels
        train_labels.head()
```

```
Out[2]:
                                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
```

Creating function designed to resize an image to fit within a fixed input_size of 244 pixels while adjusting aspect ratio and bounding box (if there is one) accordingly.

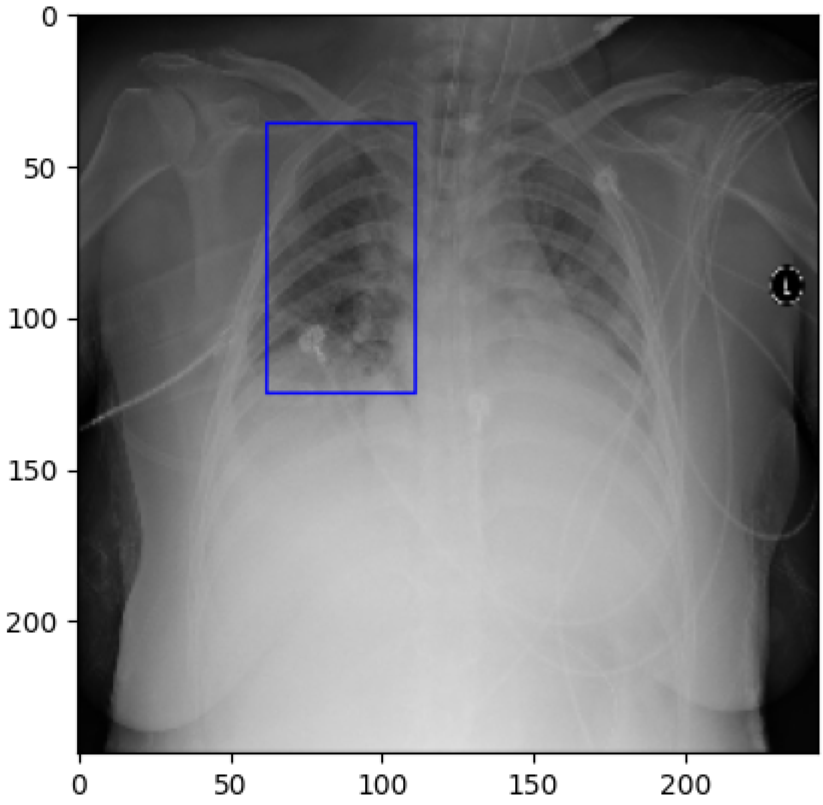
```
In [3]: input_size = 244

        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
```

Processes a sample image (00436515-...) from the training data, to visualize how the data looks. This image is one with pneumonia.

In [6]:
```
datapath = 'rsna-pneumonia-detection-challenge\stage_2_train_images/00436515-870c-4
temp_img = pydicom.dcmread(datapath).pixel_array
temp_box = train_labels[train_labels['patientId'] == '00436515-870c-4b36-a041-de910

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, 0, 255), 1)

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

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



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

In [7]:
```
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 [01:00<00:00, 86.67it/s]
5200
Loading data: 100%|██████████| 700/700 [00:08<00:00, 84.23it/s]
Loading data: 100%|██████████| 601/601 [00:07<00:00, 85.53it/s]
```

Defines a function to ready the (images, label) pair to be used in TensorFlow. The two classes are "has pneomonia" and "doesn't have pneumonia," followed by the box dimensions.

In [8]:

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

Defines three function to optimize the training/validation/testing dataset respecively for the tensor flow model.

In [10]:

```
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

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

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
```

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

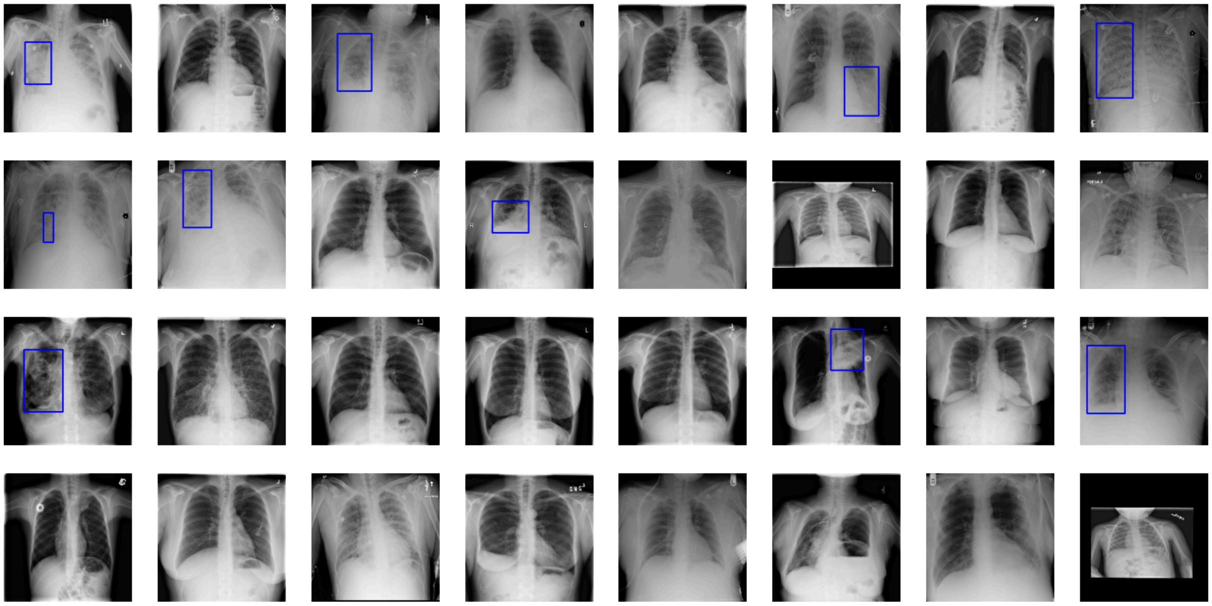
In [11]:

```
train_ds = tune_training_ds(raw_train_ds)
validation_ds = tune_validation_ds(raw_valid_ds)
test_ds = tune_test_ds(raw_test_ds)
```

Visualizing one batch of the training data set. Again, see that some patients don't have pneumonia.

In [12]:

```
plt.figure(figsize=(20, 10))
for images, labels in train_ds.take(1):
    for i in range(BATCH_SIZE):
        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)(secondary) and bounding box regression (localizing pneumonia in X-ray images).

In [13]:

```
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(2, activation='softmax', name = 'classifier_head')

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 [14]:
```
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 [16]: plot_model(model, show_shapes=True, show_layer_names=True)
```

```
You must install graphviz (see instructions at https://graphviz.gitlab.io/download/)
for `plot_model` to work.
```

Training the model with 100 epochs.

```
In [17]: history = model.fit(train_ds,
                             steps_per_epoch=(len(raw_train_ds) // BATCH_SIZE),
                             validation_data=validation_ds, validation_steps=1,
                             epochs=100)
```

```
Epoch 1/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 40s 217ms/step - classifier_head_accuracy: 0.6452 - cla
ssifier_head_loss: 0.6567 - loss: 0.7293 - regressor_head_loss: 0.0725 - regressor_h
ead_mse: 0.0725 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
0.3978 - val_loss: 0.4143 - val_regressor_head_loss: 0.0164 - val_regressor_head_ms
e: 0.0164
Epoch 2/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 214ms/step - classifier_head_accuracy: 0.7615 - cla
ssifier_head_loss: 0.5063 - loss: 0.5366 - regressor_head_loss: 0.0303 - regressor_h
ead_mse: 0.0303 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
0.4577 - val_loss: 0.4764 - val_regressor_head_loss: 0.0186 - val_regressor_head_ms
e: 0.0186
Epoch 3/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 208ms/step - classifier_head_accuracy: 0.7741 - cla
ssifier_head_loss: 0.4907 - loss: 0.5208 - regressor_head_loss: 0.0300 - regressor_h
ead_mse: 0.0300 - val_classifier_head_accuracy: 0.7600 - val_classifier_head_loss:
0.4983 - val_loss: 0.5171 - val_regressor_head_loss: 0.0188 - val_regressor_head_ms
e: 0.0188
Epoch 4/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 207ms/step - classifier_head_accuracy: 0.7893 - cla
ssifier_head_loss: 0.4619 - loss: 0.4888 - regressor_head_loss: 0.0269 - regressor_h
ead_mse: 0.0269 - val_classifier_head_accuracy: 0.8286 - val_classifier_head_loss:
0.4157 - val_loss: 0.4302 - val_regressor_head_loss: 0.0145 - val_regressor_head_ms
e: 0.0145
Epoch 5/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 207ms/step - classifier_head_accuracy: 0.7894 - cla
ssifier_head_loss: 0.4507 - loss: 0.4774 - regressor_head_loss: 0.0267 - regressor_h
ead_mse: 0.0267 - val_classifier_head_accuracy: 0.7771 - val_classifier_head_loss:
0.4703 - val_loss: 0.4874 - val_regressor_head_loss: 0.0171 - val_regressor_head_ms
e: 0.0171
Epoch 6/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 208ms/step - classifier_head_accuracy: 0.8094 - cla
ssifier_head_loss: 0.4247 - loss: 0.4506 - regressor_head_loss: 0.0259 - regressor_h
ead_mse: 0.0259 - val_classifier_head_accuracy: 0.7886 - val_classifier_head_loss:
0.4398 - val_loss: 0.4601 - val_regressor_head_loss: 0.0203 - val_regressor_head_ms
e: 0.0203
Epoch 7/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 210ms/step - classifier_head_accuracy: 0.8139 - cla
ssifier_head_loss: 0.4126 - loss: 0.4397 - regressor_head_loss: 0.0272 - regressor_h
ead_mse: 0.0272 - val_classifier_head_accuracy: 0.7371 - val_classifier_head_loss:
0.5485 - val_loss: 0.5710 - val_regressor_head_loss: 0.0225 - val_regressor_head_ms
e: 0.0225
Epoch 8/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 209ms/step - classifier_head_accuracy: 0.8354 - cla
ssifier_head_loss: 0.3653 - loss: 0.3910 - regressor_head_loss: 0.0256 - regressor_h
ead_mse: 0.0256 - val_classifier_head_accuracy: 0.7600 - val_classifier_head_loss:
0.5039 - val_loss: 0.5304 - val_regressor_head_loss: 0.0266 - val_regressor_head_ms
e: 0.0266
Epoch 9/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 213ms/step - classifier_head_accuracy: 0.8473 - cla
ssifier_head_loss: 0.3372 - loss: 0.3624 - regressor_head_loss: 0.0252 - regressor_h
ead_mse: 0.0252 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
0.5124 - val_loss: 0.5351 - val_regressor_head_loss: 0.0226 - val_regressor_head_ms
e: 0.0226
Epoch 10/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 37s 226ms/step - classifier_head_accuracy: 0.8663 - cla
```

```
ssifier_head_loss: 0.3098 - loss: 0.3344 - regressor_head_loss: 0.0246 - regressor_h
ead_mse: 0.0246 - val_classifier_head_accuracy: 0.7600 - val_classifier_head_loss:
0.5920 - val_loss: 0.6182 - val_regressor_head_loss: 0.0262 - val_regressor_head_ms
e: 0.0262
Epoch 11/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 37s 227ms/step - classifier_head_accuracy: 0.8817 - cla
ssifier_head_loss: 0.2754 - loss: 0.3009 - regressor_head_loss: 0.0255 - regressor_h
ead_mse: 0.0255 - val_classifier_head_accuracy: 0.7371 - val_classifier_head_loss:
0.6680 - val_loss: 0.6967 - val_regressor_head_loss: 0.0287 - val_regressor_head_ms
e: 0.0287
Epoch 12/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 209ms/step - classifier_head_accuracy: 0.9083 - cla
ssifier_head_loss: 0.2265 - loss: 0.2495 - regressor_head_loss: 0.0230 - regressor_h
ead_mse: 0.0230 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
0.5077 - val_loss: 0.5315 - val_regressor_head_loss: 0.0238 - val_regressor_head_ms
e: 0.0238
Epoch 13/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 213ms/step - classifier_head_accuracy: 0.9247 - cla
ssifier_head_loss: 0.1963 - loss: 0.2175 - regressor_head_loss: 0.0212 - regressor_h
ead_mse: 0.0212 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
0.5909 - val_loss: 0.6187 - val_regressor_head_loss: 0.0278 - val_regressor_head_ms
e: 0.0278
Epoch 14/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 215ms/step - classifier_head_accuracy: 0.9446 - cla
ssifier_head_loss: 0.1424 - loss: 0.1637 - regressor_head_loss: 0.0213 - regressor_h
ead_mse: 0.0213 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_loss:
0.7058 - val_loss: 0.7315 - val_regressor_head_loss: 0.0256 - val_regressor_head_ms
e: 0.0256
Epoch 15/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 213ms/step - classifier_head_accuracy: 0.9529 - cla
ssifier_head_loss: 0.1195 - loss: 0.1391 - regressor_head_loss: 0.0196 - regressor_h
ead_mse: 0.0196 - val_classifier_head_accuracy: 0.7371 - val_classifier_head_loss:
0.8945 - val_loss: 0.9232 - val_regressor_head_loss: 0.0287 - val_regressor_head_ms
e: 0.0287
Epoch 16/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 37s 225ms/step - classifier_head_accuracy: 0.9671 - cla
ssifier_head_loss: 0.0898 - loss: 0.1091 - regressor_head_loss: 0.0192 - regressor_h
ead_mse: 0.0192 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
0.6903 - val_loss: 0.7120 - val_regressor_head_loss: 0.0217 - val_regressor_head_ms
e: 0.0217
Epoch 17/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 220ms/step - classifier_head_accuracy: 0.9672 - cla
ssifier_head_loss: 0.0903 - loss: 0.1082 - regressor_head_loss: 0.0179 - regressor_h
ead_mse: 0.0179 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_loss:
0.7706 - val_loss: 0.7938 - val_regressor_head_loss: 0.0232 - val_regressor_head_ms
e: 0.0232
Epoch 18/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 216ms/step - classifier_head_accuracy: 0.9687 - cla
ssifier_head_loss: 0.0839 - loss: 0.1005 - regressor_head_loss: 0.0166 - regressor_h
ead_mse: 0.0166 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_loss:
0.7477 - val_loss: 0.7710 - val_regressor_head_loss: 0.0233 - val_regressor_head_ms
e: 0.0233
Epoch 19/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 209ms/step - classifier_head_accuracy: 0.9849 - cla
ssifier_head_loss: 0.0482 - loss: 0.0633 - regressor_head_loss: 0.0151 - regressor_h
ead_mse: 0.0151 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_loss:
```

```
1.0263 - val_loss: 1.0482 - val_regressor_head_loss: 0.0220 - val_regressor_head_ms
e: 0.0220
Epoch 20/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 214ms/step - classifier_head_accuracy: 0.9860 - cla
ssifier_head_loss: 0.0406 - loss: 0.0551 - regressor_head_loss: 0.0145 - regressor_h
ead_mse: 0.0145 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.1319 - val_loss: 1.1519 - val_regressor_head_loss: 0.0200 - val_regressor_head_ms
e: 0.0200
Epoch 21/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9931 - cla
ssifier_head_loss: 0.0231 - loss: 0.0371 - regressor_head_loss: 0.0139 - regressor_h
ead_mse: 0.0139 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.1090 - val_loss: 1.1299 - val_regressor_head_loss: 0.0208 - val_regressor_head_ms
e: 0.0208
Epoch 22/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 210ms/step - classifier_head_accuracy: 0.9887 - cla
ssifier_head_loss: 0.0372 - loss: 0.0526 - regressor_head_loss: 0.0153 - regressor_h
ead_mse: 0.0153 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_loss:
1.1427 - val_loss: 1.1616 - val_regressor_head_loss: 0.0190 - val_regressor_head_ms
e: 0.0190
Epoch 23/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 0.9950 - cla
ssifier_head_loss: 0.0220 - loss: 0.0353 - regressor_head_loss: 0.0133 - regressor_h
ead_mse: 0.0133 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.3353 - val_loss: 1.3548 - val_regressor_head_loss: 0.0195 - val_regressor_head_ms
e: 0.0195
Epoch 24/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 214ms/step - classifier_head_accuracy: 0.9943 - cla
ssifier_head_loss: 0.0232 - loss: 0.0368 - regressor_head_loss: 0.0136 - regressor_h
ead_mse: 0.0136 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_loss:
1.0506 - val_loss: 1.0711 - val_regressor_head_loss: 0.0205 - val_regressor_head_ms
e: 0.0205
Epoch 25/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 213ms/step - classifier_head_accuracy: 0.9940 - cla
ssifier_head_loss: 0.0219 - loss: 0.0354 - regressor_head_loss: 0.0135 - regressor_h
ead_mse: 0.0135 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.1778 - val_loss: 1.1978 - val_regressor_head_loss: 0.0200 - val_regressor_head_ms
e: 0.0200
Epoch 26/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 209ms/step - classifier_head_accuracy: 0.9959 - cla
ssifier_head_loss: 0.0136 - loss: 0.0256 - regressor_head_loss: 0.0120 - regressor_h
ead_mse: 0.0120 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_loss:
1.0799 - val_loss: 1.0989 - val_regressor_head_loss: 0.0190 - val_regressor_head_ms
e: 0.0190
Epoch 27/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9977 - cla
ssifier_head_loss: 0.0133 - loss: 0.0256 - regressor_head_loss: 0.0124 - regressor_h
ead_mse: 0.0124 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_loss:
1.2098 - val_loss: 1.2291 - val_regressor_head_loss: 0.0193 - val_regressor_head_ms
e: 0.0193
Epoch 28/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9992 - cla
ssifier_head_loss: 0.0052 - loss: 0.0161 - regressor_head_loss: 0.0109 - regressor_h
ead_mse: 0.0109 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_loss:
1.3813 - val_loss: 1.4012 - val_regressor_head_loss: 0.0199 - val_regressor_head_ms
e: 0.0199
```

```
Epoch 29/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 210ms/step - classifier_head_accuracy: 0.9969 - cla
ssifier_head_loss: 0.0122 - loss: 0.0243 - regressor_head_loss: 0.0121 - regressor_h
ead_mse: 0.0121 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_loss:
1.0791 - val_loss: 1.0977 - val_regressor_head_loss: 0.0186 - val_regressor_head_ms
e: 0.0186
Epoch 30/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 0.9937 - cla
ssifier_head_loss: 0.0213 - loss: 0.0335 - regressor_head_loss: 0.0122 - regressor_h
ead_mse: 0.0122 - val_classifier_head_accuracy: 0.7771 - val_classifier_head_loss:
1.2198 - val_loss: 1.2407 - val_regressor_head_loss: 0.0209 - val_regressor_head_ms
e: 0.0209
Epoch 31/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9974 - cla
ssifier_head_loss: 0.0109 - loss: 0.0232 - regressor_head_loss: 0.0123 - regressor_h
ead_mse: 0.0123 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.5600 - val_loss: 1.5830 - val_regressor_head_loss: 0.0230 - val_regressor_head_ms
e: 0.0230
Epoch 32/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9966 - cla
ssifier_head_loss: 0.0105 - loss: 0.0222 - regressor_head_loss: 0.0117 - regressor_h
ead_mse: 0.0117 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
1.6427 - val_loss: 1.6646 - val_regressor_head_loss: 0.0219 - val_regressor_head_ms
e: 0.0219
Epoch 33/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9978 - cla
ssifier_head_loss: 0.0074 - loss: 0.0188 - regressor_head_loss: 0.0114 - regressor_h
ead_mse: 0.0114 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_loss:
1.2235 - val_loss: 1.2445 - val_regressor_head_loss: 0.0210 - val_regressor_head_ms
e: 0.0210
Epoch 34/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9961 - cla
ssifier_head_loss: 0.0145 - loss: 0.0254 - regressor_head_loss: 0.0109 - regressor_h
ead_mse: 0.0109 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_loss:
1.5171 - val_loss: 1.5365 - val_regressor_head_loss: 0.0194 - val_regressor_head_ms
e: 0.0194
Epoch 35/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9998 - cla
ssifier_head_loss: 0.0021 - loss: 0.0126 - regressor_head_loss: 0.0105 - regressor_h
ead_mse: 0.0105 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_loss:
1.6815 - val_loss: 1.7016 - val_regressor_head_loss: 0.0202 - val_regressor_head_ms
e: 0.0202
Epoch 36/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9987 - cla
ssifier_head_loss: 0.0029 - loss: 0.0127 - regressor_head_loss: 0.0098 - regressor_h
ead_mse: 0.0098 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.4336 - val_loss: 1.4541 - val_regressor_head_loss: 0.0205 - val_regressor_head_ms
e: 0.0205
Epoch 37/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 215ms/step - classifier_head_accuracy: 0.9960 - cla
ssifier_head_loss: 0.0141 - loss: 0.0256 - regressor_head_loss: 0.0115 - regressor_h
ead_mse: 0.0115 - val_classifier_head_accuracy: 0.7714 - val_classifier_head_loss:
1.2205 - val_loss: 1.2437 - val_regressor_head_loss: 0.0232 - val_regressor_head_ms
e: 0.0232
Epoch 38/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9947 - cla
```

```
ssifier_head_loss: 0.0131 - loss: 0.0255 - regressor_head_loss: 0.0124 - regressor_h
ead_mse: 0.0124 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.0944 - val_loss: 1.1109 - val_regressor_head_loss: 0.0165 - val_regressor_head_ms
e: 0.0165
Epoch 39/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9966 - cla
ssifier_head_loss: 0.0149 - loss: 0.0259 - regressor_head_loss: 0.0110 - regressor_h
ead_mse: 0.0110 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_loss:
1.6370 - val_loss: 1.6571 - val_regressor_head_loss: 0.0201 - val_regressor_head_ms
e: 0.0201
Epoch 40/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 213ms/step - classifier_head_accuracy: 0.9988 - cla
ssifier_head_loss: 0.0065 - loss: 0.0167 - regressor_head_loss: 0.0102 - regressor_h
ead_mse: 0.0102 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.4408 - val_loss: 1.4603 - val_regressor_head_loss: 0.0195 - val_regressor_head_ms
e: 0.0195
Epoch 41/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 210ms/step - classifier_head_accuracy: 0.9980 - cla
ssifier_head_loss: 0.0052 - loss: 0.0150 - regressor_head_loss: 0.0098 - regressor_h
ead_mse: 0.0098 - val_classifier_head_accuracy: 0.8343 - val_classifier_head_loss:
1.6916 - val_loss: 1.7093 - val_regressor_head_loss: 0.0177 - val_regressor_head_ms
e: 0.0177
Epoch 42/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 0.9980 - cla
ssifier_head_loss: 0.0061 - loss: 0.0165 - regressor_head_loss: 0.0104 - regressor_h
ead_mse: 0.0104 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.4380 - val_loss: 1.4579 - val_regressor_head_loss: 0.0199 - val_regressor_head_ms
e: 0.0199
Epoch 43/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9992 - cla
ssifier_head_loss: 0.0031 - loss: 0.0131 - regressor_head_loss: 0.0100 - regressor_h
ead_mse: 0.0100 - val_classifier_head_accuracy: 0.8343 - val_classifier_head_loss:
1.4816 - val_loss: 1.5005 - val_regressor_head_loss: 0.0189 - val_regressor_head_ms
e: 0.0189
Epoch 44/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 0.9990 - cla
ssifier_head_loss: 0.0028 - loss: 0.0123 - regressor_head_loss: 0.0096 - regressor_h
ead_mse: 0.0096 - val_classifier_head_accuracy: 0.8286 - val_classifier_head_loss:
1.4306 - val_loss: 1.4506 - val_regressor_head_loss: 0.0200 - val_regressor_head_ms
e: 0.0200
Epoch 45/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9983 - cla
ssifier_head_loss: 0.0063 - loss: 0.0164 - regressor_head_loss: 0.0101 - regressor_h
ead_mse: 0.0101 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.6602 - val_loss: 1.6809 - val_regressor_head_loss: 0.0207 - val_regressor_head_ms
e: 0.0207
Epoch 46/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9994 - cla
ssifier_head_loss: 0.0036 - loss: 0.0137 - regressor_head_loss: 0.0101 - regressor_h
ead_mse: 0.0101 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_loss:
1.8120 - val_loss: 1.8312 - val_regressor_head_loss: 0.0192 - val_regressor_head_ms
e: 0.0192
Epoch 47/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9993 - cla
ssifier_head_loss: 0.0023 - loss: 0.0116 - regressor_head_loss: 0.0093 - regressor_h
ead_mse: 0.0093 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_loss:
```

```
1.7646 - val_loss: 1.7830 - val_regressor_head_loss: 0.0183 - val_regressor_head_ms
e: 0.0183
Epoch 48/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9996 - cla
ssifier_head_loss: 0.0015 - loss: 0.0106 - regressor_head_loss: 0.0091 - regressor_h
ead_mse: 0.0091 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.5256 - val_loss: 1.5426 - val_regressor_head_loss: 0.0171 - val_regressor_head_ms
e: 0.0171
Epoch 49/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 214ms/step - classifier_head_accuracy: 0.9986 - cla
ssifier_head_loss: 0.0038 - loss: 0.0135 - regressor_head_loss: 0.0097 - regressor_h
ead_mse: 0.0097 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.5835 - val_loss: 1.6023 - val_regressor_head_loss: 0.0188 - val_regressor_head_ms
e: 0.0188
Epoch 50/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 210ms/step - classifier_head_accuracy: 0.9973 - cla
ssifier_head_loss: 0.0110 - loss: 0.0208 - regressor_head_loss: 0.0098 - regressor_h
ead_mse: 0.0098 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.7233 - val_loss: 1.7447 - val_regressor_head_loss: 0.0213 - val_regressor_head_ms
e: 0.0213
Epoch 51/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 0.9990 - cla
ssifier_head_loss: 0.0037 - loss: 0.0135 - regressor_head_loss: 0.0098 - regressor_h
ead_mse: 0.0098 - val_classifier_head_accuracy: 0.8286 - val_classifier_head_loss:
1.3194 - val_loss: 1.3351 - val_regressor_head_loss: 0.0157 - val_regressor_head_ms
e: 0.0157
Epoch 52/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9986 - cla
ssifier_head_loss: 0.0048 - loss: 0.0143 - regressor_head_loss: 0.0095 - regressor_h
ead_mse: 0.0095 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_loss:
1.4620 - val_loss: 1.4825 - val_regressor_head_loss: 0.0205 - val_regressor_head_ms
e: 0.0205
Epoch 53/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9980 - cla
ssifier_head_loss: 0.0048 - loss: 0.0143 - regressor_head_loss: 0.0095 - regressor_h
ead_mse: 0.0095 - val_classifier_head_accuracy: 0.8286 - val_classifier_head_loss:
1.5368 - val_loss: 1.5559 - val_regressor_head_loss: 0.0191 - val_regressor_head_ms
e: 0.0191
Epoch 54/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9998 - cla
ssifier_head_loss: 8.2562e-04 - loss: 0.0092 - regressor_head_loss: 0.0084 - regress
or_head_mse: 0.0084 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_los
s: 1.5467 - val_loss: 1.5653 - val_regressor_head_loss: 0.0186 - val_regressor_head_
mse: 0.0186
Epoch 55/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 213ms/step - classifier_head_accuracy: 0.9977 - cla
ssifier_head_loss: 0.0071 - loss: 0.0164 - regressor_head_loss: 0.0093 - regressor_h
ead_mse: 0.0093 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_loss:
1.4298 - val_loss: 1.4514 - val_regressor_head_loss: 0.0216 - val_regressor_head_ms
e: 0.0216
Epoch 56/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 0.9969 - cla
ssifier_head_loss: 0.0131 - loss: 0.0239 - regressor_head_loss: 0.0107 - regressor_h
ead_mse: 0.0107 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.2084 - val_loss: 1.2268 - val_regressor_head_loss: 0.0184 - val_regressor_head_ms
e: 0.0184
```

```
Epoch 57/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 216ms/step - classifier_head_accuracy: 0.9983 - cla
ssifier_head_loss: 0.0069 - loss: 0.0169 - regressor_head_loss: 0.0100 - regressor_h
ead_mse: 0.0100 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_loss:
1.3518 - val_loss: 1.3701 - val_regressor_head_loss: 0.0183 - val_regressor_head_ms
e: 0.0183
Epoch 58/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 0.9992 - cla
ssifier_head_loss: 0.0020 - loss: 0.0113 - regressor_head_loss: 0.0093 - regressor_h
ead_mse: 0.0093 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_loss:
1.5631 - val_loss: 1.5817 - val_regressor_head_loss: 0.0185 - val_regressor_head_ms
e: 0.0185
Epoch 59/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 4.1217e-04 - loss: 0.0092 - regressor_head_loss: 0.0088 - regress
or_head_mse: 0.0088 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_los
s: 1.4780 - val_loss: 1.4956 - val_regressor_head_loss: 0.0176 - val_regressor_head_
mse: 0.0176
Epoch 60/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 0.9999 - cla
ssifier_head_loss: 5.5547e-04 - loss: 0.0085 - regressor_head_loss: 0.0080 - regress
or_head_mse: 0.0080 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_los
s: 1.4814 - val_loss: 1.4993 - val_regressor_head_loss: 0.0179 - val_regressor_head_
mse: 0.0179
Epoch 61/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 0.9989 - cla
ssifier_head_loss: 0.0013 - loss: 0.0093 - regressor_head_loss: 0.0080 - regressor_h
ead_mse: 0.0080 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_loss:
1.7696 - val_loss: 1.7881 - val_regressor_head_loss: 0.0185 - val_regressor_head_ms
e: 0.0185
Epoch 62/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 211ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 3.4875e-04 - loss: 0.0081 - regressor_head_loss: 0.0077 - regress
or_head_mse: 0.0077 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_los
s: 1.6263 - val_loss: 1.6437 - val_regressor_head_loss: 0.0174 - val_regressor_head_
mse: 0.0174
Epoch 63/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 34s 212ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 2.4859e-04 - loss: 0.0080 - regressor_head_loss: 0.0077 - regress
or_head_mse: 0.0077 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_los
s: 1.7244 - val_loss: 1.7428 - val_regressor_head_loss: 0.0184 - val_regressor_head_
mse: 0.0184
Epoch 64/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 214ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 2.6890e-04 - loss: 0.0077 - regressor_head_loss: 0.0074 - regress
or_head_mse: 0.0074 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_los
s: 1.5679 - val_loss: 1.5855 - val_regressor_head_loss: 0.0177 - val_regressor_head_
mse: 0.0177
Epoch 65/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 216ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 1.9765e-04 - loss: 0.0079 - regressor_head_loss: 0.0077 - regress
or_head_mse: 0.0077 - val_classifier_head_accuracy: 0.7886 - val_classifier_head_los
s: 1.8826 - val_loss: 1.9005 - val_regressor_head_loss: 0.0179 - val_regressor_head_
mse: 0.0179
Epoch 66/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 223ms/step - classifier_head_accuracy: 1.0000 - cla
```

```
ssifier_head_loss: 4.2061e-04 - loss: 0.0077 - regressor_head_loss: 0.0073 - regress
or_head_mse: 0.0073 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_los
s: 1.6940 - val_loss: 1.7121 - val_regressor_head_loss: 0.0181 - val_regressor_head_
mse: 0.0181
Epoch 67/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 217ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 1.1513e-04 - loss: 0.0072 - regressor_head_loss: 0.0071 - regress
or_head_mse: 0.0071 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_los
s: 1.6397 - val_loss: 1.6567 - val_regressor_head_loss: 0.0170 - val_regressor_head_
mse: 0.0170
Epoch 68/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 222ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 3.3160e-04 - loss: 0.0075 - regressor_head_loss: 0.0072 - regress
or_head_mse: 0.0072 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_los
s: 1.6849 - val_loss: 1.7025 - val_regressor_head_loss: 0.0176 - val_regressor_head_
mse: 0.0176
Epoch 69/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 217ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 1.2766e-04 - loss: 0.0070 - regressor_head_loss: 0.0068 - regress
or_head_mse: 0.0068 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_los
s: 1.6986 - val_loss: 1.7166 - val_regressor_head_loss: 0.0180 - val_regressor_head_
mse: 0.0180
Epoch 70/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 216ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 2.8524e-04 - loss: 0.0074 - regressor_head_loss: 0.0071 - regress
or_head_mse: 0.0071 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_los
s: 1.6423 - val_loss: 1.6600 - val_regressor_head_loss: 0.0177 - val_regressor_head_
mse: 0.0177
Epoch 71/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 217ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 8.8166e-05 - loss: 0.0068 - regressor_head_loss: 0.0067 - regress
or_head_mse: 0.0067 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_los
s: 1.6678 - val_loss: 1.6851 - val_regressor_head_loss: 0.0172 - val_regressor_head_
mse: 0.0172
Epoch 72/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 217ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 5.8608e-04 - loss: 0.0078 - regressor_head_loss: 0.0073 - regress
or_head_mse: 0.0073 - val_classifier_head_accuracy: 0.8229 - val_classifier_head_los
s: 1.4933 - val_loss: 1.5087 - val_regressor_head_loss: 0.0154 - val_regressor_head_
mse: 0.0154
Epoch 73/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 217ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 2.2080e-04 - loss: 0.0071 - regressor_head_loss: 0.0068 - regress
or_head_mse: 0.0068 - val_classifier_head_accuracy: 0.7886 - val_classifier_head_los
s: 1.7092 - val_loss: 1.7277 - val_regressor_head_loss: 0.0185 - val_regressor_head_
mse: 0.0185
Epoch 74/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 216ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 8.9695e-05 - loss: 0.0065 - regressor_head_loss: 0.0065 - regress
or_head_mse: 0.0065 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_los
s: 1.7266 - val_loss: 1.7445 - val_regressor_head_loss: 0.0179 - val_regressor_head_
mse: 0.0179
Epoch 75/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 218ms/step - classifier_head_accuracy: 0.9996 - cla
ssifier_head_loss: 4.0032e-04 - loss: 0.0071 - regressor_head_loss: 0.0067 - regress
or_head_mse: 0.0067 - val_classifier_head_accuracy: 0.7486 - val_classifier_head_los
```

```
s: 1.7179 - val_loss: 1.7371 - val_regressor_head_loss: 0.0192 - val_regressor_head_
mse: 0.0192
Epoch 76/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 221ms/step - classifier_head_accuracy: 0.9897 - cla
ssifier_head_loss: 0.0293 - loss: 0.0402 - regressor_head_loss: 0.0108 - regressor_h
ead_mse: 0.0108 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_loss:
1.4712 - val_loss: 1.4933 - val_regressor_head_loss: 0.0222 - val_regressor_head_ms
e: 0.0222
Epoch 77/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 219ms/step - classifier_head_accuracy: 0.9925 - cla
ssifier_head_loss: 0.0226 - loss: 0.0344 - regressor_head_loss: 0.0118 - regressor_h
ead_mse: 0.0118 - val_classifier_head_accuracy: 0.8171 - val_classifier_head_loss:
1.1496 - val_loss: 1.1692 - val_regressor_head_loss: 0.0196 - val_regressor_head_ms
e: 0.0196
Epoch 78/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 219ms/step - classifier_head_accuracy: 0.9968 - cla
ssifier_head_loss: 0.0111 - loss: 0.0214 - regressor_head_loss: 0.0103 - regressor_h
ead_mse: 0.0103 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_loss:
1.8183 - val_loss: 1.8365 - val_regressor_head_loss: 0.0181 - val_regressor_head_ms
e: 0.0181
Epoch 79/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 222ms/step - classifier_head_accuracy: 0.9994 - cla
ssifier_head_loss: 0.0027 - loss: 0.0121 - regressor_head_loss: 0.0093 - regressor_h
ead_mse: 0.0093 - val_classifier_head_accuracy: 0.7886 - val_classifier_head_loss:
1.6601 - val_loss: 1.6779 - val_regressor_head_loss: 0.0178 - val_regressor_head_ms
e: 0.0178
Epoch 80/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 217ms/step - classifier_head_accuracy: 0.9999 - cla
ssifier_head_loss: 7.0734e-04 - loss: 0.0086 - regressor_head_loss: 0.0079 - regress
or_head_mse: 0.0079 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_los
s: 1.7285 - val_loss: 1.7471 - val_regressor_head_loss: 0.0186 - val_regressor_head_
mse: 0.0186
Epoch 81/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 220ms/step - classifier_head_accuracy: 0.9986 - cla
ssifier_head_loss: 0.0037 - loss: 0.0122 - regressor_head_loss: 0.0085 - regressor_h
ead_mse: 0.0085 - val_classifier_head_accuracy: 0.8343 - val_classifier_head_loss:
1.6328 - val_loss: 1.6500 - val_regressor_head_loss: 0.0172 - val_regressor_head_ms
e: 0.0172
Epoch 82/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 223ms/step - classifier_head_accuracy: 0.9986 - cla
ssifier_head_loss: 0.0049 - loss: 0.0142 - regressor_head_loss: 0.0093 - regressor_h
ead_mse: 0.0093 - val_classifier_head_accuracy: 0.7886 - val_classifier_head_loss:
1.7291 - val_loss: 1.7478 - val_regressor_head_loss: 0.0187 - val_regressor_head_ms
e: 0.0187
Epoch 83/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 222ms/step - classifier_head_accuracy: 0.9989 - cla
ssifier_head_loss: 0.0020 - loss: 0.0106 - regressor_head_loss: 0.0086 - regressor_h
ead_mse: 0.0086 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_loss:
1.7481 - val_loss: 1.7658 - val_regressor_head_loss: 0.0177 - val_regressor_head_ms
e: 0.0177
Epoch 84/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 220ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 2.5190e-04 - loss: 0.0082 - regressor_head_loss: 0.0079 - regress
or_head_mse: 0.0079 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_los
s: 1.9020 - val_loss: 1.9218 - val_regressor_head_loss: 0.0198 - val_regressor_head_
mse: 0.0198
```

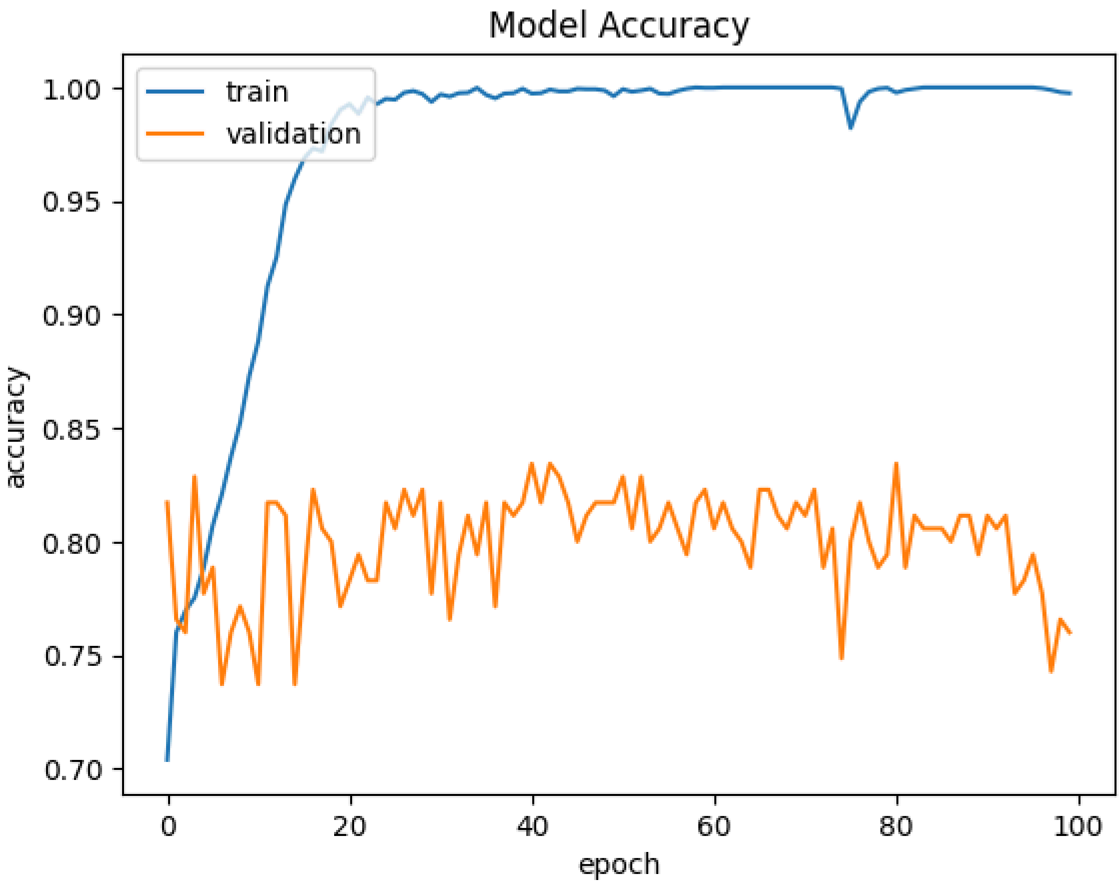
```
Epoch 85/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 221ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 9.9876e-05 - loss: 0.0076 - regressor_head_loss: 0.0075 - regress
or_head_mse: 0.0075 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_los
s: 1.8471 - val_loss: 1.8648 - val_regressor_head_loss: 0.0176 - val_regressor_head_
mse: 0.0176
Epoch 86/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 35s 219ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 5.9785e-04 - loss: 0.0080 - regressor_head_loss: 0.0075 - regress
or_head_mse: 0.0075 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_los
s: 1.8178 - val_loss: 1.8357 - val_regressor_head_loss: 0.0179 - val_regressor_head_
mse: 0.0179
Epoch 87/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 219ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 3.1416e-04 - loss: 0.0074 - regressor_head_loss: 0.0071 - regress
or_head_mse: 0.0071 - val_classifier_head_accuracy: 0.8000 - val_classifier_head_los
s: 1.8685 - val_loss: 1.8862 - val_regressor_head_loss: 0.0178 - val_regressor_head_
mse: 0.0178
Epoch 88/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 220ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 2.3685e-04 - loss: 0.0072 - regressor_head_loss: 0.0070 - regress
or_head_mse: 0.0070 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_los
s: 1.9044 - val_loss: 1.9230 - val_regressor_head_loss: 0.0185 - val_regressor_head_
mse: 0.0185
Epoch 89/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 219ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 6.0100e-05 - loss: 0.0069 - regressor_head_loss: 0.0068 - regress
or_head_mse: 0.0068 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_los
s: 1.9018 - val_loss: 1.9203 - val_regressor_head_loss: 0.0185 - val_regressor_head_
mse: 0.0185
Epoch 90/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 221ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 4.1210e-05 - loss: 0.0069 - regressor_head_loss: 0.0068 - regress
or_head_mse: 0.0068 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_los
s: 2.0295 - val_loss: 2.0485 - val_regressor_head_loss: 0.0190 - val_regressor_head_
mse: 0.0190
Epoch 91/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 221ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 1.4826e-04 - loss: 0.0073 - regressor_head_loss: 0.0072 - regress
or_head_mse: 0.0072 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_los
s: 1.9025 - val_loss: 1.9209 - val_regressor_head_loss: 0.0184 - val_regressor_head_
mse: 0.0184
Epoch 92/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 221ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 5.2011e-04 - loss: 0.0073 - regressor_head_loss: 0.0068 - regress
or_head_mse: 0.0068 - val_classifier_head_accuracy: 0.8057 - val_classifier_head_los
s: 1.8884 - val_loss: 1.9058 - val_regressor_head_loss: 0.0175 - val_regressor_head_
mse: 0.0175
Epoch 93/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 220ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 6.1657e-05 - loss: 0.0068 - regressor_head_loss: 0.0067 - regress
or_head_mse: 0.0067 - val_classifier_head_accuracy: 0.8114 - val_classifier_head_los
s: 1.7995 - val_loss: 1.8171 - val_regressor_head_loss: 0.0177 - val_regressor_head_
mse: 0.0177
Epoch 94/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 220ms/step - classifier_head_accuracy: 1.0000 - cla
```

```
ssifier_head_loss: 5.8021e-05 - loss: 0.0067 - regressor_head_loss: 0.0067 - regress
or_head_mse: 0.0067 - val_classifier_head_accuracy: 0.7771 - val_classifier_head_los
s: 1.8456 - val_loss: 1.8631 - val_regressor_head_loss: 0.0174 - val_regressor_head_
mse: 0.0174
Epoch 95/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 222ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 5.5537e-04 - loss: 0.0074 - regressor_head_loss: 0.0069 - regress
or_head_mse: 0.0069 - val_classifier_head_accuracy: 0.7829 - val_classifier_head_los
s: 1.8145 - val_loss: 1.8328 - val_regressor_head_loss: 0.0183 - val_regressor_head_
mse: 0.0183
Epoch 96/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 221ms/step - classifier_head_accuracy: 1.0000 - cla
ssifier_head_loss: 5.9073e-05 - loss: 0.0066 - regressor_head_loss: 0.0065 - regress
or_head_mse: 0.0065 - val_classifier_head_accuracy: 0.7943 - val_classifier_head_los
s: 1.6198 - val_loss: 1.6365 - val_regressor_head_loss: 0.0167 - val_regressor_head_
mse: 0.0167
Epoch 97/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 221ms/step - classifier_head_accuracy: 0.9998 - cla
ssifier_head_loss: 3.2754e-04 - loss: 0.0071 - regressor_head_loss: 0.0067 - regress
or_head_mse: 0.0067 - val_classifier_head_accuracy: 0.7771 - val_classifier_head_los
s: 1.6916 - val_loss: 1.7090 - val_regressor_head_loss: 0.0174 - val_regressor_head_
mse: 0.0174
Epoch 98/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 221ms/step - classifier_head_accuracy: 0.9997 - cla
ssifier_head_loss: 0.0012 - loss: 0.0088 - regressor_head_loss: 0.0076 - regressor_h
ead_mse: 0.0076 - val_classifier_head_accuracy: 0.7429 - val_classifier_head_loss:
2.2307 - val_loss: 2.2541 - val_regressor_head_loss: 0.0234 - val_regressor_head_ms
e: 0.0234
Epoch 99/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 221ms/step - classifier_head_accuracy: 0.9980 - cla
ssifier_head_loss: 0.0071 - loss: 0.0171 - regressor_head_loss: 0.0100 - regressor_h
ead_mse: 0.0100 - val_classifier_head_accuracy: 0.7657 - val_classifier_head_loss:
1.9907 - val_loss: 2.0117 - val_regressor_head_loss: 0.0210 - val_regressor_head_ms
e: 0.0210
Epoch 100/100
162/162 ━━━━━━━━━━━━━━━━━━━━ 36s 224ms/step - classifier_head_accuracy: 0.9965 - cla
ssifier_head_loss: 0.0073 - loss: 0.0170 - regressor_head_loss: 0.0098 - regressor_h
ead_mse: 0.0098 - val_classifier_head_accuracy: 0.7600 - val_classifier_head_loss:
2.1485 - val_loss: 2.1669 - val_regressor_head_loss: 0.0184 - val_regressor_head_ms
e: 0.0184
```

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

In [18]:

```python
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, using the formula in the kaggle challenge overview.

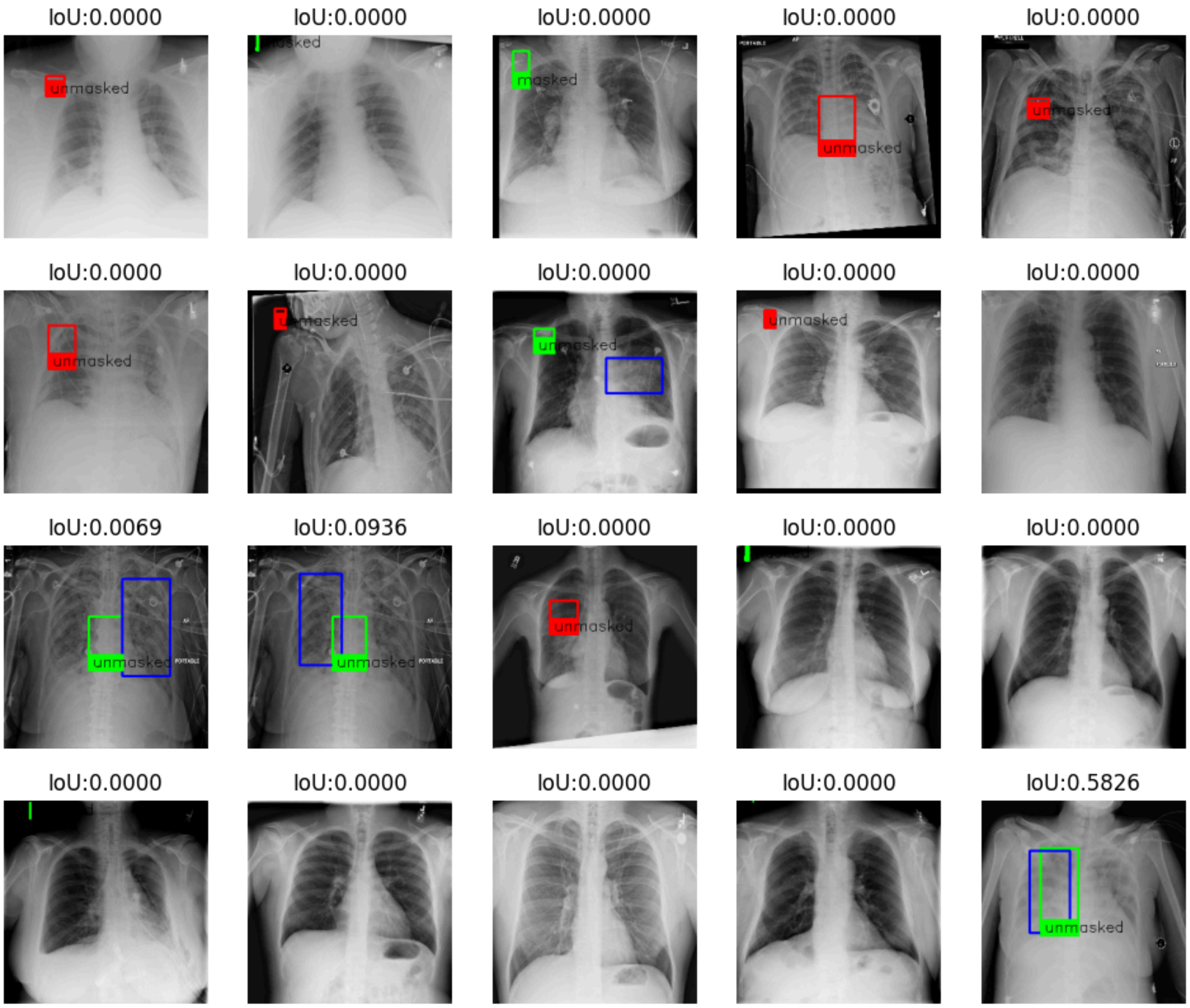
In [19]:

```
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
```

Visualizing 20 test predicitons for the model along with the calculated IoU. Red box printed if a case was wrongly classified. Otherwise, printed a green box if correctly classified. The box label is unmasked if the model classified it as not having pneumonia and masked if it was classified as having pneumonia.

In [ ]:

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

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

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")
```

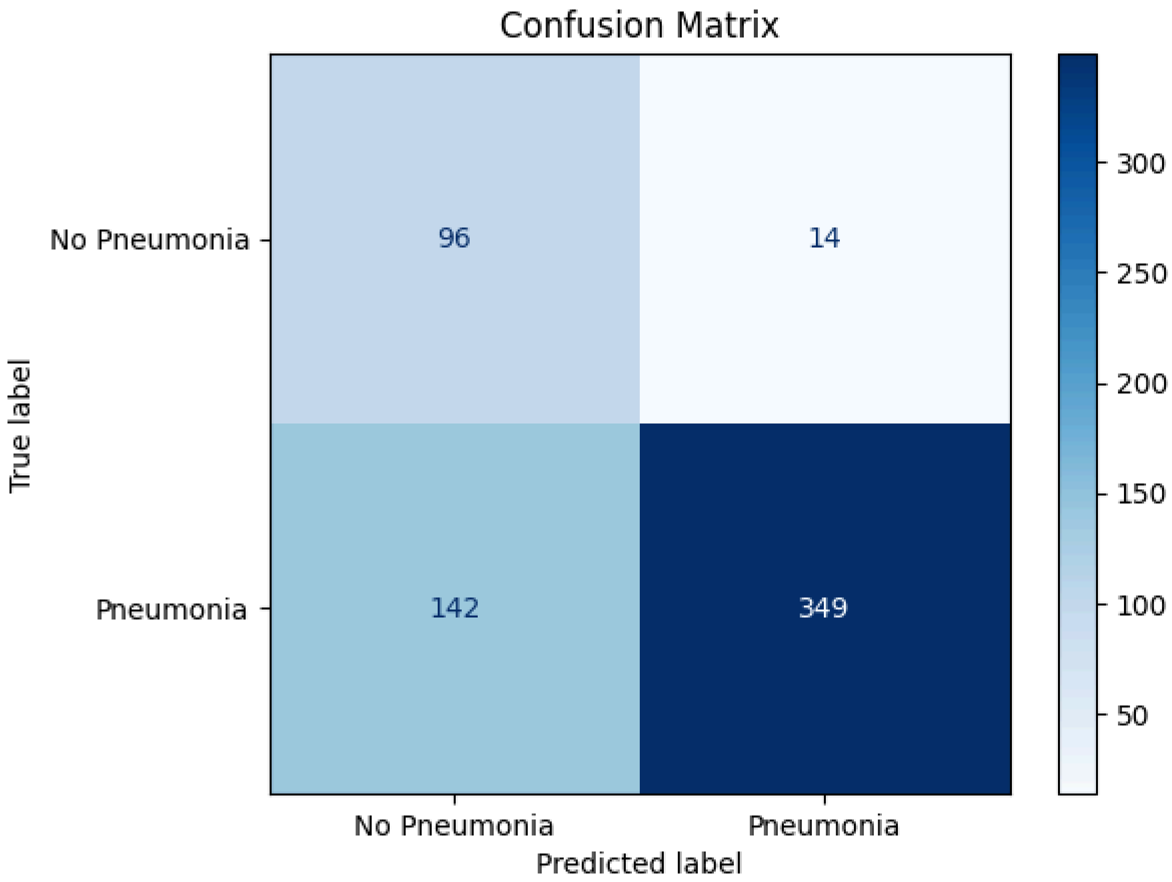


Evaluation of the trained model's predictions on test set: calculating accuracy & Intersection over Union (IoU). Also created confusion matrix for the categorization.

In [111…

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

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

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

correct_count = 0
total_count = 0
iou_list = []
label_predicted = []
label_actual = []

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]

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

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

    total_count += 1

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


    #IoU
    IoU = intersection_over_union(predicted_box.numpy(), actual_box.numpy())
    iou_list.append(IoU)
    label_predicted.append(int(predicted_label[0]>0.5))
    label_actual.append(int(actual_label[0]))

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



confusionmatrix = confusion_matrix(label_actual, label_predicted)
disp = ConfusionMatrixDisplay(confusionmatrix, display_labels=['No Pneumonia', 'Pne
disp.plot
plt.title("Confusion Matrix")
plt.show()

recall = confusionmatrix[1][1]/(confusionmatrix[1][1] + confusionmatrix[1][0])
precision = confusionmatrix[1][1]/(confusionmatrix[1][1] + confusionmatrix[0][1])
F1 = 2 * (precision * recall) / (precision + recall)
print(f"Recall: {recall:.4f}, Precision: {precision:.4f}, F1 Score: {F1:.4f}")
print(f"Accuracy: {accuracy:.4f}")
print(f"Mean IoU: {average_iou:.4f}")
```

```
Test Data Size: 601
<Figure size 1200x1000 with 0 Axes>
```



```
Recall: 0.7108, Precision: 0.9614, F1 Score: 0.8173
Accuracy: 0.7404
Mean IoU: 0.0148
```