Detection and Sorting of Screws Using Deep CNNs

February 3, 2022

1 Detection and Sorting of Screws Using Deep CNNs

1.1 Group Memebers:

1. 2018BTEEN00065 Pradyumna Santosh Akolkar
2. 2019BTEEN00206 Aishwarya Jagannath Kumbhar

1.2 Problem Statement:

We have seven images of two types of screws and our aim is to detect, classify and create bounding box around them.

1.3 Objectives:

- Understanding the variables included and preprocessing the data.
- Graphically representing the data for insights.
- Classification of various types of screws.
- Creating bounding box around detected screw.

1.4 Introduction:

Deep Learning is proved to be excellent while classifying and detecting objects from an image. Here, we need to perform screw detection which is classification problem as well as bounding box creation which is regression problem.

1.5 Importing Necessary Libraries:

[ ]:
```
import PIL
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import pandas as pd
import os

from ast import literal_eval
from random import randrange
from PIL import Image
```

```
from tensorflow import keras
from tensorflow.keras import layers
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
import zipfile
zf = zipfile.ZipFile('/content/drive/MyDrive/Screw Detection/FinalDatasetNew.
↪zip','r')
zf.extractall()
```

[ ]:
```
drive.flush_and_unmount()
```

[ ]:
```
physical_devices = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(physical_devices[0], True)
```

[ ]:
```
os.chdir('FinalDatasetNew')
```

1.6 Preprocessing of the Data:

We have 7 original images which are given to us as training examples. Deep neural networks need much larger training data for good performance. Thus, we use data augmentation techniques to create more mumber of images from original images. To do that we use different methods such as rotation, flipping, taking random crops etc. For our problem statement, we require positive as well as negative images i.e. images which do not contain any screws. To make sure the dataset is not skewed, we need to make number of various types of images remains nearly equal. Hence, overall we created 244 images (86 NonScrews + 72 TypeA + 86 TypeB) and then convert them into grayscale images. Original size of images is 1024x768 which require more computing power. To reduce this we resized them to 512x384. The following code is used to achieve all of this -

[ ]:
```
# Code to create non-screw images

def create_non_screw_images(number_of_images):
    global count
    for each in range(0,256,int(255/number_of_images)):
        count+=1
        img = Image.fromarray(np.array([each])).resize((512,384),Image.
↪ANTIALIAS).convert("L")
        img.save("UnlabeledDataset/NonScrews/"+str(count)+".png")
```

[ ]:
```
# Code to create rotating images

def create_rotated_images(img,i):
    global count
    for each in range(0,360,30):
```

```
        count += 1
        img1 = img.rotate(each,expand=0,fillcolor='#929292').resize((512,384),␣
↪Image.ANTIALIAS).convert('L')
        if(i<4):
            img1.save("UnlabeledDataset/Screws/TypeA/"+str(count)+".png")
        else:
            img1.save("UnlabeledDataset/Screws/TypeB/"+str(count)+".png")
```

[ ]:
```
# Code to horizontally and vertically flipped images

def create_flipped_images(img,i):
    global count
    #Flipping image left right
    img2 = img.transpose(Image.FLIP_LEFT_RIGHT).resize((512,384), Image.
↪ANTIALIAS).convert('L')
    #Flipping image top bottom
    img3 = img.transpose(Image.FLIP_TOP_BOTTOM).resize((512,384), Image.
↪ANTIALIAS).convert('L')
    if(i<4):
        count += 1
        img2.save("UnlabeledDataset/Screws/TypeA/"+str(count)+".png")
        count+=1
        img3.save("UnlabeledDataset/Screws/TypeA/"+str(count)+".png")
    else:
        count+=1
        img2.save("UnlabeledDataset/Screws/TypeB/"+str(count)+".png")
        count+=1
        img3.save("UnlabeledDataset/Screws/TypeB/"+str(count)+".png")
```

[ ]:
```
# Code to take random crops from images

def create_random_crops(img,i,crop_size):
    global count
    x, y = img.size
    sample = 10

    for j in range(sample):
        count+=1
        x1 = randrange(0, x - crop_size)
        y1 = randrange(0, y - crop_size)
        img1 = img.crop((x1, y1, x1 + crop_size, y1 + crop_size)).
↪resize((512,384),Image.ANTIALIAS).convert("L")
        if(i<4):
            img1.save("UnlabeledDataset/Screws/TypeA/"+str(count)+".png")
        else:
            img1.save("UnlabeledDataset/Screws/TypeB/"+str(count)+".png")
```

```
[ ]: # Code to perform data augmentation (code to create images)

count = 0 # Global count to keep track of number of images
for i in range(1,8):
    # Opening original images
    if(i<4):
        img = Image.open('OriginalImages/'+str(i)+'.jpeg').convert('L')
    else:
        img = Image.open('OriginalImages/'+str(i)+'.jpg').convert('L')
    create_rotated_images(img,i)
    create_flipped_images(img,i)
    if(i<4):
        crop_size = 655
    else:
        crop_size = 625
    if(i!=7):
        create_random_crops(img,i,crop_size)
screw_image_count = count
print("Number of Screw images: ",screw_image_count)
num_of_non_screw_images = 80
create_non_screw_images(num_of_non_screw_images)
print("Number of Non-Screw images: ",count-screw_image_count)
print("Number of Total images: ",count)
```

```
Number of Screw images:  158
Number of Non-Screw images:  86
Number of Total images:  244
```

After creating the images, we need labels for them denoting following things - - pc = If image contains screws then 1, otherwise 0 - c1 = If present screw is of typeA then 1, otherwise 0 - c2 = If present screw is of typeB then 1, otherwise 0 - bx = x coordinate of center of detected screw - by = y coordinate of center of detected screw - bh = Height of bounding box around the detected screw - bw = Width of bounding box around the detected screw

To create labels for images we implement following algorithm - 1. Using Inkscape software, we created red colored bounding box around screws in images, we colored the upper right corner with green color and lower left corner with blue color. 2. We saved the colored images in such a way that the folder containing them gives us labels pc, c1 and c2 3. In find_bb_labels function, we take these colored images and split each image in Red, Green, Blue channels. 4. We used index of maximum value in each of these channels which gives us upper left corner, upper right corner and lower left corner of the bounding box. 5. We used these indices to find bx, by, bh and bw. 6. Finally, we created and saved a dataframe as “LabeledDataset.csv” which contains two columns- Image path and labels for each image.

```
[ ]: def find_bb_labels(img):
    red, green, blue = img.split()
    red = np.array(red).T
    red = np.array(np.unravel_index(np.argmax(red),red.shape))
```

```
    green = np.array(green).T
    green = np.array(np.unravel_index(np.argmax(green),green.shape))
    blue = np.array(blue).T
    blue = np.array(np.unravel_index(np.argmax(blue),blue.shape))
    bh = (blue - red)[1]
    bw = (green - red)[0]
    bx = red[0] + round(bw/2)
    by = red[1] + round(bh/2)

    return bx, by, bh, bw
```

[ ]:
```
df = pd.DataFrame(columns=['Image Path','Label (pc, c1, c2, bx, by, bh, bw)'])
```

[ ]:
```
#Labels for NonScrews images
for i in os.listdir('ColoredDataset/NonScrews'):
    pc = 0
    c1 = 0
    c2 = 0
    bx = 0
    by = 0
    bh = 0
    bw = 0
    df = df.append({'Image Path':'LabeledDataset/NonScrews/'+i,'Label (pc, c1,␣
 ↪c2, bx, by, bh, bw)':(pc,c1,c2,bx,by,bh,bw)},ignore_index=True)
```

[ ]:
```
#Labels for Screws images
for i in os.listdir('ColoredDataset/Screws/TypeA'):
    img = Image.open('ColoredDataset/Screws/TypeA/'+i).convert("RGB")
    pc = 1
    c1 = 1 #These are type A screws
    c2 = 0
    bx, by, bh, bw = find_bb_labels(img)
    df = df.append({'Image Path':'LabeledDataset/Screws/TypeA/'+i,'Label (pc,␣
 ↪c1, c2, bx, by, bh, bw)':(pc,c1,c2,bx,by,bh,bw)},ignore_index=True)

for i in os.listdir('ColoredDataset/Screws/TypeB'):
    img = Image.open('ColoredDataset/Screws/TypeB/'+i).convert("RGB")
    pc = 1
    c1 = 0
    c2 = 1 #These are type B screws
    bx, by, bh, bw = find_bb_labels(img)
    df = df.append({'Image Path':'LabeledDataset/Screws/TypeB/'+i,'Label (pc,␣
 ↪c1, c2, bx, by, bh, bw)':(pc,c1,c2,bx,by,bh,bw)},ignore_index=True)
```

[ ]:
```
df
```

[ ]:
```
                              Image Path Label (pc, c1, c2, bx, by, bh, bw)
0      LabeledDataset/NonScrews/159.png              (0, 0, 0, 0, 0, 0, 0)
1      LabeledDataset/NonScrews/160.png              (0, 0, 0, 0, 0, 0, 0)
2      LabeledDataset/NonScrews/161.png              (0, 0, 0, 0, 0, 0, 0)
3      LabeledDataset/NonScrews/162.png              (0, 0, 0, 0, 0, 0, 0)
4      LabeledDataset/NonScrews/163.png              (0, 0, 0, 0, 0, 0, 0)
..                                  …                                  …
239  LabeledDataset/Screws/TypeB/95.png      (1, 0, 1, 207, 254, 129, 384)
240  LabeledDataset/Screws/TypeB/96.png      (1, 0, 1, 315, 283, 130, 384)
241  LabeledDataset/Screws/TypeB/97.png       (1, 0, 1, 305, 129, 233, 95)
242  LabeledDataset/Screws/TypeB/98.png      (1, 0, 1, 260, 111, 222, 170)
243  LabeledDataset/Screws/TypeB/99.png      (1, 0, 1, 221, 110, 172, 230)

[244 rows x 2 columns]
```

[ ]:
```
df.to_csv("LabeledDataset.csv")
```

1.6.1 Code to check the correctness of labels:

[ ]:
```
df = pd.read_csv('LabeledDataset.csv')
```

[ ]:
```
# Converting string data to tuple
df['Label (pc, c1, c2, bx, by, bh, bw)'] = [literal_eval(str(x)) for x in␣
↪df['Label (pc, c1, c2, bx, by, bh, bw)']]
```

[ ]:
```
# Code to create bounding box with lables

def create_bounding_box(image_path, labels):
    sample = Image.open(image_path).convert("RGB")
    pc, c1, c2, bx, by, bh, bw = labels
    fig, ax = plt.subplots(figsize=(10,10))
    ax.imshow(sample)
    rect = patches.Rectangle((bx-bw/2,by-bh/2), bw, bh, linewidth=1,␣
↪edgecolor='r', facecolor='none')
    ax.add_patch(rect)
    plt.show()
    print("pc: ",pc)
    print("c1: ",c1)
    print("c2: ",c2)
    print("bx: ",bx)
    print("by: ",by)
    print("bh: ",bh)
    print("bw: ",bw)
```

Original Image -

[ ]:
```python
img_path = 'LabeledDataset/Screws/TypeA/25.png'
img = Image.open(img_path)
img
```

[ ]:

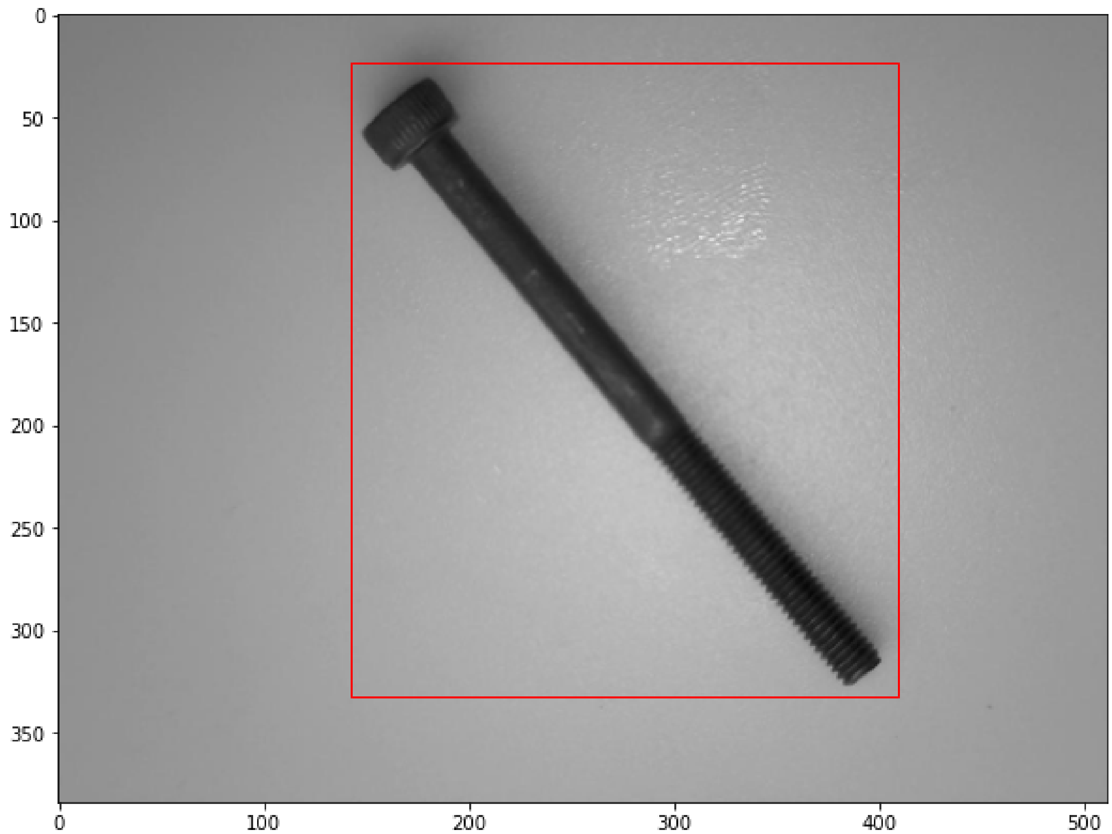
Colored Image using Inkscape -

[ ]:
```python
img_path = 'ColoredDataset/Screws/TypeA/25.png'
img = Image.open(img_path)
img
```

[ ]:

Bounding box recreated on original image from labels -

[ ]:
```
img_path = 'LabeledDataset/Screws/TypeA/25.png'
labels = df.loc[df['Image Path']==img_path, 'Label (pc, c1, c2, bx, by, bh,␣
↪bw)'].iloc[0]
create_bounding_box(img_path, labels)
```




```
pc:  1
c1:  1
c2:  0
bx:  276
by:  178
bh:  309
bw:  267
```

1.7 Model Insights:

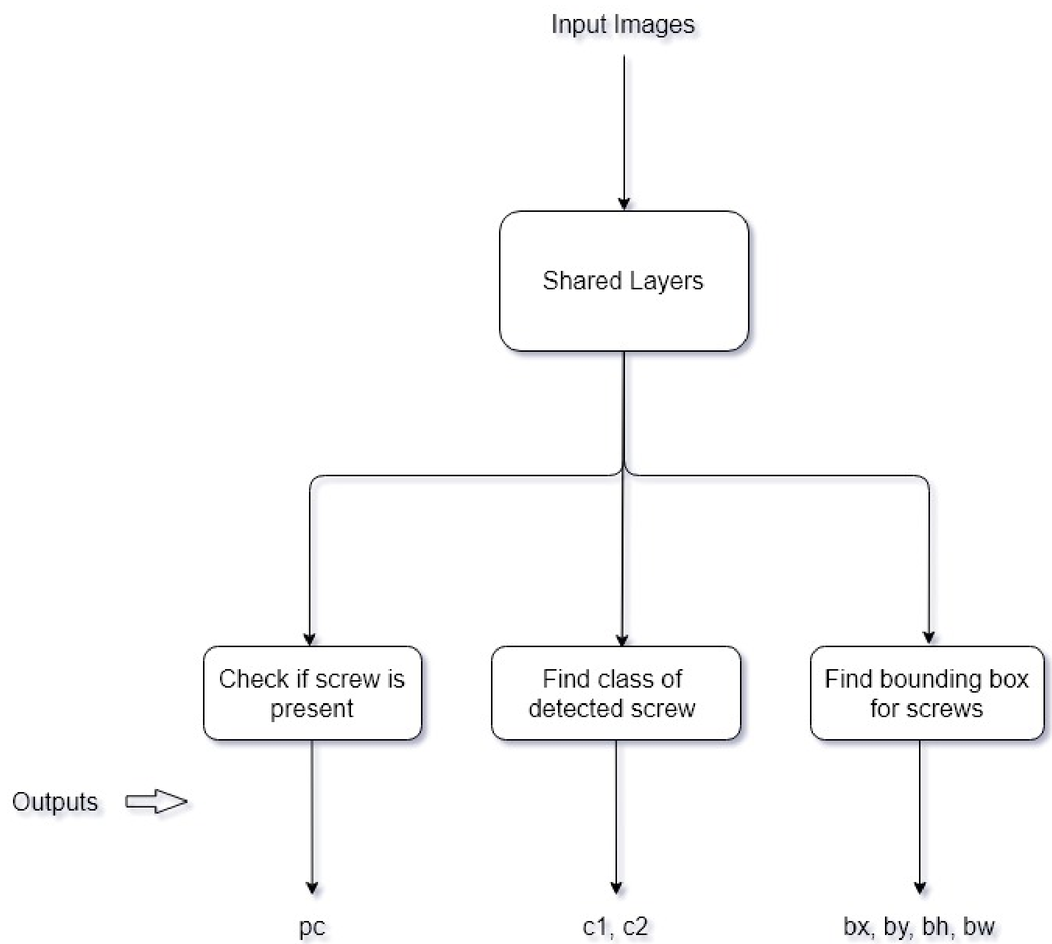


Fig.1 - Block Diagram of Model

In our project, we need to perform three tasks which are, to check if the screw is present or not (pc), if the screw is present then find the class of the screw (c1,c2), and find bounding box for the screw (bx,by,bh,bw). Usually, one Neural Network is required to perform one task. Here, we use Multi-Task learning which is more efficient way of performing these tasks where we use single Neural Network for implementation. Here, we use Convolution Neural Network. CNN is basically a stack of layers which defines number of filters on input. Those filters are nothing but kernels which neural network learns. Kernel size refers to width and height of filter. Max pooling returns the maximum value from number of pixels. Stride denotes how many pixels we are moving our kernel in each step in convolution. In our model, We use ReLu and sigmoid as activation functions. ReLu is non-linear activation function used in multi-layer neural network.The main advantage of ReLu is that it does not activate all neurons at the same time. This means that neurons will be deactivated if the output of linear transformation is less than 0. Sigmoid function takes a value as input and outputs another value between 0 and 1. A flatten layer is used to collapses the spatial dimensions of the input into one single dimension i.e. it flattens all feature inputs into a single column. Dense layer is interconnected neural network layer. Loss is nothing but a prediction error of neural network which is used to calculate gradients and those are used to update weights. Here we use Mean Square Error (MSE) which is the sum of squared distances between target values and predicted values; and categorical_crossentropy loss is used in multi-class classification. Stochastic Gradient Descent optimizer is used to change attributes of the neural network such as weights to reduce the losses. Metrics are used for model evaluation. Here, we use accuracy to measure model performance for ‘pc’ and ‘c12’ as these are classification problems; and we use mean squared error for ‘bb’ as this is regression problem.

1.8 Define helper functions for ML Model:

[ ]:
```
# Code to create train, test datasets

def ttdataset(df,train_percentage):

    df = df.sample(frac=1).reset_index(drop = True)
    names = df['Image Path']
    labels = [literal_eval(str(x)) for x in df['Label (pc, c1, c2, bx, by, bh,␣
↪bw)']]
    labels_pc = np.array([np.array(i[0]) for i in labels])
    labels_c12 = np.array([(i[1],i[2]) for i in labels])
    labels_bb = np.array([(i[3]/512,i[4]/384,i[5]/384,i[6]/512) for i in␣
↪labels],dtype='float')

    total = df.shape[0]
    train_index = int(total*train_percentage/100)

    img_list=[]
    for each in names:
        img = tf.keras.preprocessing.image.load_img(each,color_mode='grayscale')
        img = tf.keras.preprocessing.image.img_to_array(img)
        img = img/255
        img_list.append(img)

    img_list = np.array(img_list)
    train_x = img_list[:train_index]
    train_y_pc = labels_pc[:train_index]
    train_y_c12 = labels_c12[:train_index]
    train_y_bb = labels_bb[:train_index]
    test_x = img_list[train_index:]
    test_y_pc = labels_pc[train_index:]
    test_y_c12 = labels_c12[train_index:]
    test_y_bb = labels_bb[train_index:]

    return train_x, train_y_pc, train_y_c12, train_y_bb, test_x, test_y_pc,␣
↪test_y_c12, test_y_bb
```

```
[ ]: # code to plot the performance of the model

def plot_performance(history):
    plt.figure(figsize=(20,10))
    plt.plot(history.history['pc_accuracy'])
    plt.plot(history.history['c12_accuracy'])
    plt.plot(history.history['bb_mse'])
    plt.plot(history.history['val_pc_accuracy'])
    plt.plot(history.history['val_c12_accuracy'])
    plt.plot(history.history['val_bb_mse'])
    plt.title('Model Performance')
    plt.ylabel('performance measure')
    plt.xlabel('epoch')
    plt.legend(['pc_accuracy', 'c12_accuracy', 'bb_mse','val_pc_accuracy',␣
 ↪'val_c12_accuracy', 'val_bb_mse'], loc='center right')
    plt.show()
```

```
[ ]: # Code to build model

def build_model(seed):
    np.random.seed(seed)
    inputs = keras.Input((384,512,1))
    x = layers.Conv2D(filters=32, kernel_size=(30,30), strides=(1,1),␣
 ↪activation='relu')(inputs)
    x = layers.MaxPooling2D(pool_size=(3,3))(x)
    x = layers.Conv2D(filters=64, kernel_size=(25,25), strides=(1,1),␣
 ↪activation='relu')(x)
    x = layers.MaxPooling2D(pool_size=(3,3))(x)
    x = layers.Conv2D(filters=64, kernel_size=(15,15), strides=(1,1),␣
 ↪activation='relu')(x)
    x = layers.MaxPooling2D(pool_size=(2,2))(x)
    x = layers.Conv2D(filters=64, kernel_size=(5,5), strides=(1,1),␣
 ↪activation='relu')(x)
    x = layers.MaxPooling2D(pool_size=(2,2))(x)
    x = layers.Flatten()(x)
    #x = layers.Dense(units=300, activation='relu')(x)
    x = layers.Dense(units=200, activation='relu')(x)
    x = layers.Dense(units=100, activation='relu')(x)
    x = layers.Dense(units=75, activation='sigmoid')(x)
    #x = layers.Dense(units=20, activation='relu')(x)
    output1 = layers.Dense(units=1, activation='sigmoid', name='pc')(x)
    x = layers.Dense(units=50, activation='relu')(x)
    x = layers.Dense(units=20, activation='relu')(x)
    output2 = layers.Dense(units=2, activation='sigmoid',name='c12')(x)
    output3 = layers.Dense(units=4, activation='sigmoid', name='bb')(x)
    model = keras.Model(inputs=inputs,outputs=[output1,output2,output3],␣
 ↪name='ScrewDetector')
```

```
    model.compile(optimizer='sgd', loss={
              'pc':'mse',
              'c12':'mse',
              'bb':'mse'},
              metrics=['accuracy','mse','mae'])


    return model
```

[ ]:
```
# Code to run the model

def run_model(model, train_x, train_y_pc, train_y_c12, train_y_bb, epochs,␣
 ↪batch_size=64, verbose=0):
    history=model.fit(x=train_x, y={'pc':train_y_pc,'c12':train_y_c12,'bb':
 ↪train_y_bb}, batch_size=batch_size,epochs=epochs, verbose=verbose,␣
 ↪validation_split=0.1)

    return history
```

1.9 Implement the Machine Learning model:

[ ]:
```
# Building the model

model = build_model(48)
```

[ ]:
```
# Printing model summary

model.summary()
```

```
Model: "ScrewDetector"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_12 (InputLayer)           [(None, 384, 512, 1) 0

________________________________________________________________________________
__________________
conv2d_44 (Conv2D)              (None, 355, 483, 32) 28832       input_12[0][0]

________________________________________________________________________________
__________________
max_pooling2d_44 (MaxPooling2D) (None, 118, 161, 32) 0           conv2d_44[0][0]

________________________________________________________________________________
__________________
conv2d_45 (Conv2D)              (None, 94, 137, 64)  1280064
max_pooling2d_44[0][0]
```

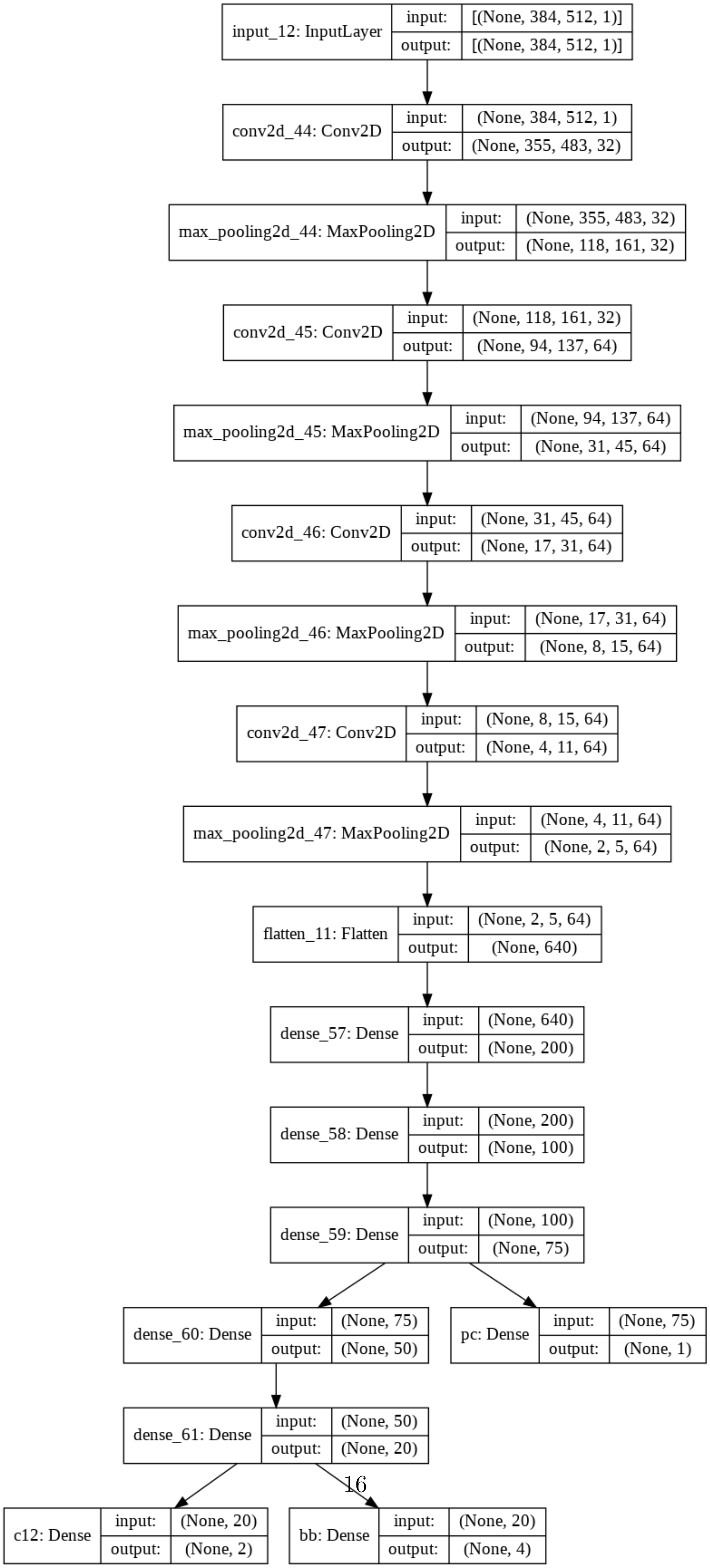
```
__________________________________________________________________________________
__________________
max_pooling2d_45 (MaxPooling2D) (None, 31, 45, 64)   0           conv2d_45[0][0]
__________________________________________________________________________________
__________________
conv2d_46 (Conv2D)              (None, 17, 31, 64)   921664
max_pooling2d_45[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_46 (MaxPooling2D) (None, 8, 15, 64)    0           conv2d_46[0][0]
__________________________________________________________________________________
__________________
conv2d_47 (Conv2D)              (None, 4, 11, 64)    102464
max_pooling2d_46[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_47 (MaxPooling2D) (None, 2, 5, 64)     0           conv2d_47[0][0]
__________________________________________________________________________________
__________________
flatten_11 (Flatten)            (None, 640)          0
max_pooling2d_47[0][0]
__________________________________________________________________________________
__________________
dense_57 (Dense)                (None, 200)          128200
flatten_11[0][0]
__________________________________________________________________________________
__________________
dense_58 (Dense)                (None, 100)          20100       dense_57[0][0]
__________________________________________________________________________________
__________________
dense_59 (Dense)                (None, 75)           7575        dense_58[0][0]
__________________________________________________________________________________
__________________
dense_60 (Dense)                (None, 50)           3800        dense_59[0][0]
__________________________________________________________________________________
__________________
dense_61 (Dense)                (None, 20)           1020        dense_60[0][0]
__________________________________________________________________________________
__________________
pc (Dense)                      (None, 1)            76          dense_59[0][0]
__________________________________________________________________________________
__________________
c12 (Dense)                     (None, 2)            42          dense_61[0][0]
__________________________________________________________________________________
__________________
bb (Dense)                      (None, 4)            84          dense_61[0][0]
==================================================================================
==================
```

```
Total params: 2,493,921
Trainable params: 2,493,921
Non-trainable params: 0
________________________________________________________________________________
_________________
```

[ ]:
```
# Plotting model architecture for better visualization

keras.utils.plot_model(model, 'model_architecture.png', show_shapes=True)
```

[ ]:


input_12: InputLayer
input: [(None, 384, 512, 1)]
output: [(None, 384, 512, 1)]
conv2d_44: Conv2D
input: (None, 384, 512, 1)
output: (None, 355, 483, 32)
max_pooling2d_44: MaxPooling2D
input: (None, 355, 483, 32)
output: (None, 118, 161, 32)
conv2d_45: Conv2D
input: (None, 118, 161, 32)
output: (None, 94, 137, 64)
max_pooling2d_45: MaxPooling2D
input: (None, 94, 137, 64)
output: (None, 31, 45, 64)
conv2d_46: Conv2D
input: (None, 31, 45, 64)
output: (None, 17, 31, 64)
max_pooling2d_46: MaxPooling2D
input: (None, 17, 31, 64)
output: (None, 8, 15, 64)
conv2d_47: Conv2D
input: (None, 8, 15, 64)
output: (None, 4, 11, 64)
max_pooling2d_47: MaxPooling2D
input: (None, 4, 11, 64)
output: (None, 2, 5, 64)
flatten_11: Flatten
input: (None, 2, 5, 64)
output: (None, 640)
dense_57: Dense
input: (None, 640)
output: (None, 200)
dense_58: Dense
input: (None, 200)
output: (None, 100)
dense_59: Dense
input: (None, 100)
output: (None, 75)
dense_60: Dense
input: (None, 75)
output: (None, 50)
pc: Dense
input: (None, 75)
output: (None, 1)
dense_61: Dense
input: (None, 50)
output: (None, 20)
c12: Dense
input: (None, 20)
output: (None, 2)
bb: Dense
input: (None, 20)
output: (None, 4)

[ ]:
```
# Creating train and test datasets

train_set_percentage = 90
train_x, train_y_pc, train_y_c12, train_y_bb, test_x, test_y_pc, test_y_c12,␣
 ↪test_y_bb = ttdataset(df,train_set_percentage)
print("Number of Train Set Examples: ",train_x.shape)
print("Number of Train pc labels: ",train_y_pc.shape)
print("Number of Train c12 labels: ",train_y_c12.shape)
print("Number of Train bb labels: ",train_y_bb.shape)
print("Number of Test Set Examples: ",test_x.shape)
print("Number of Test pc labels: ",test_y_pc.shape)
print("Number of Test c12 labels: ",test_y_c12.shape)
print("Number of Test bb labels: ",test_y_bb.shape)
```

[ ]:
```
# Running the model

history = run_model(model,train_x,train_y_pc, train_y_c12,␣
 ↪train_y_bb,batch_size=64,epochs=100,verbose=1)
```

```
Epoch 1/100
4/4 [==============================] - 3s 719ms/step - loss: 0.3662 - pc_loss:
0.1382 - c12_loss: 0.1625 - bb_loss: 0.0656 - pc_accuracy: 0.8477 - pc_mse:
0.1382 - pc_mae: 0.3222 - c12_accuracy: 0.8325 - c12_mse: 0.1625 - c12_mae:
0.3708 - bb_accuracy: 0.2792 - bb_mse: 0.0656 - bb_mae: 0.2180 - val_loss:
0.3352 - val_pc_loss: 0.1111 - val_c12_loss: 0.1639 - val_bb_loss: 0.0602 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.1111 - val_pc_mae: 0.2888 -
val_c12_accuracy: 0.7273 - val_c12_mse: 0.1639 - val_c12_mae: 0.3647 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0602 - val_bb_mae: 0.2138
Epoch 2/100
4/4 [==============================] - 3s 710ms/step - loss: 0.2353 - pc_loss:
0.0636 - c12_loss: 0.1197 - bb_loss: 0.0520 - pc_accuracy: 0.9848 - pc_mse:
0.0636 - pc_mae: 0.2213 - c12_accuracy: 0.8985 - c12_mse: 0.1197 - c12_mae:
0.3077 - bb_accuracy: 0.3096 - bb_mse: 0.0520 - bb_mae: 0.1957 - val_loss:
0.3545 - val_pc_loss: 0.1138 - val_c12_loss: 0.1685 - val_bb_loss: 0.0722 -
val_pc_accuracy: 0.8636 - val_pc_mse: 0.1138 - val_pc_mae: 0.2645 -
val_c12_accuracy: 0.3182 - val_c12_mse: 0.1685 - val_c12_mae: 0.3054 -
val_bb_accuracy: 0.0909 - val_bb_mse: 0.0722 - val_bb_mae: 0.2242
Epoch 3/100
4/4 [==============================] - 3s 716ms/step - loss: 0.3315 - pc_loss:
0.1077 - c12_loss: 0.1654 - bb_loss: 0.0584 - pc_accuracy: 0.8020 - pc_mse:
0.1077 - pc_mae: 0.2603 - c12_accuracy: 0.6091 - c12_mse: 0.1654 - c12_mae:
0.3326 - bb_accuracy: 0.2640 - bb_mse: 0.0584 - bb_mae: 0.2059 - val_loss:
0.3644 - val_pc_loss: 0.1215 - val_c12_loss: 0.1746 - val_bb_loss: 0.0683 -
val_pc_accuracy: 0.8636 - val_pc_mse: 0.1215 - val_pc_mae: 0.2928 -
val_c12_accuracy: 0.7273 - val_c12_mse: 0.1746 - val_c12_mae: 0.3221 -
```

```
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0683 - val_bb_mae: 0.2262
Epoch 4/100
4/4 [==============================] - 3s 716ms/step - loss: 0.2775 - pc_loss:
0.0944 - c12_loss: 0.1262 - bb_loss: 0.0568 - pc_accuracy: 0.8883 - pc_mse:
0.0944 - pc_mae: 0.2465 - c12_accuracy: 0.9137 - c12_mse: 0.1262 - c12_mae:
0.2905 - bb_accuracy: 0.3249 - bb_mse: 0.0568 - bb_mae: 0.2046 - val_loss:
0.2827 - val_pc_loss: 0.0927 - val_c12_loss: 0.1310 - val_bb_loss: 0.0590 -
val_pc_accuracy: 0.8182 - val_pc_mse: 0.0927 - val_pc_mae: 0.2383 -
val_c12_accuracy: 0.8182 - val_c12_mse: 0.1310 - val_c12_mae: 0.3268 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0590 - val_bb_mae: 0.2115
Epoch 5/100
4/4 [==============================] - 3s 721ms/step - loss: 0.1777 - pc_loss:
0.0355 - c12_loss: 0.1006 - bb_loss: 0.0416 - pc_accuracy: 0.9442 - pc_mse:
0.0355 - pc_mae: 0.1415 - c12_accuracy: 0.8883 - c12_mse: 0.1006 - c12_mae:
0.2679 - bb_accuracy: 0.3452 - bb_mse: 0.0416 - bb_mae: 0.1744 - val_loss:
0.2151 - val_pc_loss: 0.0330 - val_c12_loss: 0.1350 - val_bb_loss: 0.0472 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0330 - val_pc_mae: 0.1230 -
val_c12_accuracy: 0.7727 - val_c12_mse: 0.1350 - val_c12_mae: 0.2779 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0472 - val_bb_mae: 0.1901
Epoch 6/100
4/4 [==============================] - 3s 774ms/step - loss: 0.1264 - pc_loss:
0.0123 - c12_loss: 0.0763 - bb_loss: 0.0378 - pc_accuracy: 1.0000 - pc_mse:
0.0123 - pc_mae: 0.0980 - c12_accuracy: 0.9340 - c12_mse: 0.0763 - c12_mae:
0.2307 - bb_accuracy: 0.3350 - bb_mse: 0.0378 - bb_mae: 0.1702 - val_loss:
0.1818 - val_pc_loss: 0.0264 - val_c12_loss: 0.1122 - val_bb_loss: 0.0432 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0264 - val_pc_mae: 0.1096 -
val_c12_accuracy: 0.7727 - val_c12_mse: 0.1122 - val_c12_mae: 0.2592 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0432 - val_bb_mae: 0.1800
Epoch 7/100
4/4 [==============================] - 3s 724ms/step - loss: 0.1114 - pc_loss:
0.0083 - c12_loss: 0.0673 - bb_loss: 0.0358 - pc_accuracy: 1.0000 - pc_mse:
0.0083 - pc_mae: 0.0839 - c12_accuracy: 0.9442 - c12_mse: 0.0673 - c12_mae:
0.2177 - bb_accuracy: 0.3452 - bb_mse: 0.0358 - bb_mae: 0.1650 - val_loss:
0.1829 - val_pc_loss: 0.0251 - val_c12_loss: 0.1143 - val_bb_loss: 0.0436 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0251 - val_pc_mae: 0.1026 -
val_c12_accuracy: 0.7727 - val_c12_mse: 0.1143 - val_c12_mae: 0.2544 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0436 - val_bb_mae: 0.1798
Epoch 8/100
4/4 [==============================] - 3s 730ms/step - loss: 0.1034 - pc_loss:
0.0073 - c12_loss: 0.0602 - bb_loss: 0.0359 - pc_accuracy: 1.0000 - pc_mse:
0.0073 - pc_mae: 0.0781 - c12_accuracy: 0.9594 - c12_mse: 0.0602 - c12_mae:
0.2042 - bb_accuracy: 0.3350 - bb_mse: 0.0359 - bb_mae: 0.1652 - val_loss:
0.1672 - val_pc_loss: 0.0211 - val_c12_loss: 0.1081 - val_bb_loss: 0.0380 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0211 - val_pc_mae: 0.0964 -
val_c12_accuracy: 0.8182 - val_c12_mse: 0.1081 - val_c12_mae: 0.2506 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0380 - val_bb_mae: 0.1646
Epoch 9/100
4/4 [==============================] - 3s 727ms/step - loss: 0.1026 - pc_loss:
```

```
0.0064 - c12_loss: 0.0620 - bb_loss: 0.0342 - pc_accuracy: 1.0000 - pc_mse:
0.0064 - pc_mae: 0.0742 - c12_accuracy: 0.9442 - c12_mse: 0.0620 - c12_mae:
0.2051 - bb_accuracy: 0.3452 - bb_mse: 0.0342 - bb_mae: 0.1606 - val_loss:
0.1840 - val_pc_loss: 0.0170 - val_c12_loss: 0.1306 - val_bb_loss: 0.0364 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0170 - val_pc_mae: 0.0922 -
val_c12_accuracy: 0.8182 - val_c12_mse: 0.1306 - val_c12_mae: 0.2667 -
val_bb_accuracy: 0.2727 - val_bb_mse: 0.0364 - val_bb_mae: 0.1606
Epoch 10/100
4/4 [==============================] - 3s 734ms/step - loss: 0.1016 - pc_loss:
0.0061 - c12_loss: 0.0609 - bb_loss: 0.0346 - pc_accuracy: 1.0000 - pc_mse:
0.0061 - pc_mae: 0.0727 - c12_accuracy: 0.9442 - c12_mse: 0.0609 - c12_mae:
0.1994 - bb_accuracy: 0.3401 - bb_mse: 0.0346 - bb_mae: 0.1616 - val_loss:
0.1734 - val_pc_loss: 0.0319 - val_c12_loss: 0.1000 - val_bb_loss: 0.0414 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0319 - val_pc_mae: 0.0951 -
val_c12_accuracy: 0.8636 - val_c12_mse: 0.1000 - val_c12_mae: 0.2342 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0414 - val_bb_mae: 0.1729
Epoch 11/100
4/4 [==============================] - 3s 736ms/step - loss: 0.0900 - pc_loss:
0.0056 - c12_loss: 0.0493 - bb_loss: 0.0351 - pc_accuracy: 1.0000 - pc_mse:
0.0056 - pc_mae: 0.0680 - c12_accuracy: 0.9543 - c12_mse: 0.0493 - c12_mae:
0.1832 - bb_accuracy: 0.3503 - bb_mse: 0.0351 - bb_mae: 0.1629 - val_loss:
0.2134 - val_pc_loss: 0.0402 - val_c12_loss: 0.1256 - val_bb_loss: 0.0477 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0402 - val_pc_mae: 0.1057 -
val_c12_accuracy: 0.7727 - val_c12_mse: 0.1256 - val_c12_mae: 0.2478 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0477 - val_bb_mae: 0.1875
Epoch 12/100
4/4 [==============================] - 3s 730ms/step - loss: 0.1238 - pc_loss:
0.0132 - c12_loss: 0.0723 - bb_loss: 0.0382 - pc_accuracy: 0.9949 - pc_mse:
0.0132 - pc_mae: 0.0908 - c12_accuracy: 0.9137 - c12_mse: 0.0723 - c12_mae:
0.2148 - bb_accuracy: 0.3401 - bb_mse: 0.0382 - bb_mae: 0.1709 - val_loss:
0.1625 - val_pc_loss: 0.0370 - val_c12_loss: 0.0848 - val_bb_loss: 0.0407 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0370 - val_pc_mae: 0.0950 -
val_c12_accuracy: 0.8636 - val_c12_mse: 0.0848 - val_c12_mae: 0.2176 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0407 - val_bb_mae: 0.1705
Epoch 13/100
4/4 [==============================] - 3s 730ms/step - loss: 0.0891 - pc_loss:
0.0059 - c12_loss: 0.0484 - bb_loss: 0.0348 - pc_accuracy: 1.0000 - pc_mse:
0.0059 - pc_mae: 0.0671 - c12_accuracy: 0.9543 - c12_mse: 0.0484 - c12_mae:
0.1805 - bb_accuracy: 0.3452 - bb_mse: 0.0348 - bb_mae: 0.1619 - val_loss:
0.1387 - val_pc_loss: 0.0217 - val_c12_loss: 0.0805 - val_bb_loss: 0.0365 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0217 - val_pc_mae: 0.0846 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0805 - val_c12_mae: 0.2157 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0365 - val_bb_mae: 0.1610
Epoch 14/100
4/4 [==============================] - 3s 730ms/step - loss: 0.0841 - pc_loss:
0.0051 - c12_loss: 0.0449 - bb_loss: 0.0341 - pc_accuracy: 1.0000 - pc_mse:
0.0051 - pc_mae: 0.0652 - c12_accuracy: 0.9695 - c12_mse: 0.0449 - c12_mae:
0.1739 - bb_accuracy: 0.3350 - bb_mse: 0.0341 - bb_mae: 0.1606 - val_loss:
```

```
0.1462 - val_pc_loss: 0.0289 - val_c12_loss: 0.0785 - val_bb_loss: 0.0388 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0289 - val_pc_mae: 0.0878 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0785 - val_c12_mae: 0.2094 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0388 - val_bb_mae: 0.1658
Epoch 15/100
4/4 [==============================] - 3s 728ms/step - loss: 0.0848 - pc_loss:
0.0047 - c12_loss: 0.0462 - bb_loss: 0.0339 - pc_accuracy: 1.0000 - pc_mse:
0.0047 - pc_mae: 0.0633 - c12_accuracy: 0.9594 - c12_mse: 0.0462 - c12_mae:
0.1750 - bb_accuracy: 0.3452 - bb_mse: 0.0339 - bb_mae: 0.1598 - val_loss:
0.1338 - val_pc_loss: 0.0200 - val_c12_loss: 0.0762 - val_bb_loss: 0.0376 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0200 - val_pc_mae: 0.0813 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0762 - val_c12_mae: 0.2069 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0376 - val_bb_mae: 0.1632
Epoch 16/100
4/4 [==============================] - 3s 721ms/step - loss: 0.0792 - pc_loss:
0.0046 - c12_loss: 0.0405 - bb_loss: 0.0341 - pc_accuracy: 1.0000 - pc_mse:
0.0046 - pc_mae: 0.0628 - c12_accuracy: 0.9695 - c12_mse: 0.0405 - c12_mae:
0.1664 - bb_accuracy: 0.3401 - bb_mse: 0.0341 - bb_mae: 0.1603 - val_loss:
0.1229 - val_pc_loss: 0.0155 - val_c12_loss: 0.0712 - val_bb_loss: 0.0363 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0155 - val_pc_mae: 0.0767 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0712 - val_c12_mae: 0.2019 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0363 - val_bb_mae: 0.1608
Epoch 17/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0776 - pc_loss:
0.0046 - c12_loss: 0.0392 - bb_loss: 0.0338 - pc_accuracy: 1.0000 - pc_mse:
0.0046 - pc_mae: 0.0626 - c12_accuracy: 0.9695 - c12_mse: 0.0392 - c12_mae:
0.1637 - bb_accuracy: 0.3350 - bb_mse: 0.0338 - bb_mae: 0.1599 - val_loss:
0.1876 - val_pc_loss: 0.0277 - val_c12_loss: 0.1173 - val_bb_loss: 0.0425 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0277 - val_pc_mae: 0.0875 -
val_c12_accuracy: 0.7727 - val_c12_mse: 0.1173 - val_c12_mae: 0.2362 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0425 - val_bb_mae: 0.1765
Epoch 18/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0867 - pc_loss:
0.0047 - c12_loss: 0.0482 - bb_loss: 0.0339 - pc_accuracy: 1.0000 - pc_mse:
0.0047 - pc_mae: 0.0628 - c12_accuracy: 0.9543 - c12_mse: 0.0482 - c12_mae:
0.1773 - bb_accuracy: 0.3503 - bb_mse: 0.0339 - bb_mae: 0.1600 - val_loss:
0.1596 - val_pc_loss: 0.0393 - val_c12_loss: 0.0824 - val_bb_loss: 0.0379 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0393 - val_pc_mae: 0.0973 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0824 - val_c12_mae: 0.2126 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0379 - val_bb_mae: 0.1620
Epoch 19/100
4/4 [==============================] - 3s 714ms/step - loss: 0.0855 - pc_loss:
0.0068 - c12_loss: 0.0447 - bb_loss: 0.0340 - pc_accuracy: 1.0000 - pc_mse:
0.0068 - pc_mae: 0.0683 - c12_accuracy: 0.9695 - c12_mse: 0.0447 - c12_mae:
0.1722 - bb_accuracy: 0.3249 - bb_mse: 0.0340 - bb_mae: 0.1602 - val_loss:
0.1353 - val_pc_loss: 0.0114 - val_c12_loss: 0.0866 - val_bb_loss: 0.0372 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0114 - val_pc_mae: 0.0734 -
val_c12_accuracy: 0.8636 - val_c12_mse: 0.0866 - val_c12_mae: 0.2131 -
```

```
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0372 - val_bb_mae: 0.1657
Epoch 20/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0801 - pc_loss:
0.0044 - c12_loss: 0.0423 - bb_loss: 0.0334 - pc_accuracy: 1.0000 - pc_mse:
0.0044 - pc_mae: 0.0617 - c12_accuracy: 0.9594 - c12_mse: 0.0423 - c12_mae:
0.1675 - bb_accuracy: 0.3452 - bb_mse: 0.0334 - bb_mae: 0.1593 - val_loss:
0.1188 - val_pc_loss: 0.0189 - val_c12_loss: 0.0639 - val_bb_loss: 0.0360 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0189 - val_pc_mae: 0.0804 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0639 - val_c12_mae: 0.1886 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0360 - val_bb_mae: 0.1598
Epoch 21/100
4/4 [==============================] - 3s 714ms/step - loss: 0.0730 - pc_loss:
0.0046 - c12_loss: 0.0351 - bb_loss: 0.0333 - pc_accuracy: 1.0000 - pc_mse:
0.0046 - pc_mae: 0.0622 - c12_accuracy: 0.9695 - c12_mse: 0.0351 - c12_mae:
0.1563 - bb_accuracy: 0.3350 - bb_mse: 0.0333 - bb_mae: 0.1584 - val_loss:
0.1052 - val_pc_loss: 0.0079 - val_c12_loss: 0.0630 - val_bb_loss: 0.0344 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0079 - val_pc_mae: 0.0707 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0630 - val_c12_mae: 0.1905 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0344 - val_bb_mae: 0.1579
Epoch 22/100
4/4 [==============================] - 3s 714ms/step - loss: 0.0720 - pc_loss:
0.0044 - c12_loss: 0.0344 - bb_loss: 0.0332 - pc_accuracy: 1.0000 - pc_mse:
0.0044 - pc_mae: 0.0619 - c12_accuracy: 0.9695 - c12_mse: 0.0344 - c12_mae:
0.1553 - bb_accuracy: 0.3350 - bb_mse: 0.0332 - bb_mae: 0.1583 - val_loss:
0.1134 - val_pc_loss: 0.0129 - val_c12_loss: 0.0670 - val_bb_loss: 0.0335 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0129 - val_pc_mae: 0.0758 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0670 - val_c12_mae: 0.1925 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0335 - val_bb_mae: 0.1546
Epoch 23/100
4/4 [==============================] - 3s 710ms/step - loss: 0.0723 - pc_loss:
0.0045 - c12_loss: 0.0348 - bb_loss: 0.0329 - pc_accuracy: 1.0000 - pc_mse:
0.0045 - pc_mae: 0.0619 - c12_accuracy: 0.9695 - c12_mse: 0.0348 - c12_mae:
0.1554 - bb_accuracy: 0.3350 - bb_mse: 0.0329 - bb_mae: 0.1578 - val_loss:
0.2105 - val_pc_loss: 0.0250 - val_c12_loss: 0.1427 - val_bb_loss: 0.0428 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0250 - val_pc_mae: 0.0945 -
val_c12_accuracy: 0.7727 - val_c12_mse: 0.1427 - val_c12_mae: 0.2517 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0428 - val_bb_mae: 0.1790
Epoch 24/100
4/4 [==============================] - 3s 708ms/step - loss: 0.1085 - pc_loss:
0.0056 - c12_loss: 0.0689 - bb_loss: 0.0340 - pc_accuracy: 1.0000 - pc_mse:
0.0056 - pc_mae: 0.0692 - c12_accuracy: 0.9137 - c12_mse: 0.0689 - c12_mae:
0.2006 - bb_accuracy: 0.3299 - bb_mse: 0.0340 - bb_mae: 0.1598 - val_loss:
0.1440 - val_pc_loss: 0.0267 - val_c12_loss: 0.0801 - val_bb_loss: 0.0372 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0267 - val_pc_mae: 0.0859 -
val_c12_accuracy: 0.8636 - val_c12_mse: 0.0801 - val_c12_mae: 0.2035 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0372 - val_bb_mae: 0.1620
Epoch 25/100
4/4 [==============================] - 3s 711ms/step - loss: 0.0717 - pc_loss:
```

```
0.0046 - c12_loss: 0.0341 - bb_loss: 0.0329 - pc_accuracy: 1.0000 - pc_mse:
0.0046 - pc_mae: 0.0617 - c12_accuracy: 0.9695 - c12_mse: 0.0341 - c12_mae:
0.1536 - bb_accuracy: 0.3299 - bb_mse: 0.0329 - bb_mae: 0.1572 - val_loss:
0.1396 - val_pc_loss: 0.0297 - val_c12_loss: 0.0734 - val_bb_loss: 0.0365 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0297 - val_pc_mae: 0.0867 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0734 - val_c12_mae: 0.1955 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0365 - val_bb_mae: 0.1589
Epoch 26/100
4/4 [==============================] - 3s 715ms/step - loss: 0.0697 - pc_loss:
0.0044 - c12_loss: 0.0326 - bb_loss: 0.0327 - pc_accuracy: 1.0000 - pc_mse:
0.0044 - pc_mae: 0.0605 - c12_accuracy: 0.9695 - c12_mse: 0.0326 - c12_mae:
0.1502 - bb_accuracy: 0.3198 - bb_mse: 0.0327 - bb_mae: 0.1567 - val_loss:
0.1097 - val_pc_loss: 0.0127 - val_c12_loss: 0.0630 - val_bb_loss: 0.0339 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0127 - val_pc_mae: 0.0731 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0630 - val_c12_mae: 0.1859 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0339 - val_bb_mae: 0.1558
Epoch 27/100
4/4 [==============================] - 3s 710ms/step - loss: 0.0684 - pc_loss:
0.0042 - c12_loss: 0.0319 - bb_loss: 0.0324 - pc_accuracy: 1.0000 - pc_mse:
0.0042 - pc_mae: 0.0597 - c12_accuracy: 0.9695 - c12_mse: 0.0319 - c12_mae:
0.1490 - bb_accuracy: 0.3249 - bb_mse: 0.0324 - bb_mae: 0.1559 - val_loss:
0.1330 - val_pc_loss: 0.0225 - val_c12_loss: 0.0767 - val_bb_loss: 0.0338 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0225 - val_pc_mae: 0.0824 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0767 - val_c12_mae: 0.1976 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0338 - val_bb_mae: 0.1548
Epoch 28/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0713 - pc_loss:
0.0044 - c12_loss: 0.0347 - bb_loss: 0.0321 - pc_accuracy: 1.0000 - pc_mse:
0.0044 - pc_mae: 0.0612 - c12_accuracy: 0.9645 - c12_mse: 0.0347 - c12_mae:
0.1528 - bb_accuracy: 0.3096 - bb_mse: 0.0321 - bb_mae: 0.1553 - val_loss:
0.1338 - val_pc_loss: 0.0208 - val_c12_loss: 0.0764 - val_bb_loss: 0.0366 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0208 - val_pc_mae: 0.0796 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0764 - val_c12_mae: 0.1970 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0366 - val_bb_mae: 0.1609
Epoch 29/100
4/4 [==============================] - 3s 714ms/step - loss: 0.0679 - pc_loss:
0.0042 - c12_loss: 0.0313 - bb_loss: 0.0325 - pc_accuracy: 1.0000 - pc_mse:
0.0042 - pc_mae: 0.0601 - c12_accuracy: 0.9797 - c12_mse: 0.0313 - c12_mae:
0.1472 - bb_accuracy: 0.3401 - bb_mse: 0.0325 - bb_mae: 0.1558 - val_loss:
0.0941 - val_pc_loss: 0.0060 - val_c12_loss: 0.0565 - val_bb_loss: 0.0316 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0060 - val_pc_mae: 0.0721 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0565 - val_c12_mae: 0.1874 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0316 - val_bb_mae: 0.1558
Epoch 30/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0708 - pc_loss:
0.0043 - c12_loss: 0.0346 - bb_loss: 0.0320 - pc_accuracy: 1.0000 - pc_mse:
0.0043 - pc_mae: 0.0607 - c12_accuracy: 0.9695 - c12_mse: 0.0346 - c12_mae:
0.1546 - bb_accuracy: 0.3198 - bb_mse: 0.0320 - bb_mae: 0.1561 - val_loss:
```

```
0.1052 - val_pc_loss: 0.0109 - val_c12_loss: 0.0615 - val_bb_loss: 0.0327 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0109 - val_pc_mae: 0.0716 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0615 - val_c12_mae: 0.1822 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0327 - val_bb_mae: 0.1530
Epoch 31/100
4/4 [==============================] - 3s 713ms/step - loss: 0.0667 - pc_loss:
0.0041 - c12_loss: 0.0306 - bb_loss: 0.0319 - pc_accuracy: 1.0000 - pc_mse:
0.0041 - pc_mae: 0.0596 - c12_accuracy: 0.9797 - c12_mse: 0.0306 - c12_mae:
0.1459 - bb_accuracy: 0.3299 - bb_mse: 0.0319 - bb_mae: 0.1548 - val_loss:
0.1020 - val_pc_loss: 0.0060 - val_c12_loss: 0.0657 - val_bb_loss: 0.0304 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0060 - val_pc_mae: 0.0658 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0657 - val_c12_mae: 0.1886 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0304 - val_bb_mae: 0.1483
Epoch 32/100
4/4 [==============================] - 3s 719ms/step - loss: 0.0665 - pc_loss:
0.0042 - c12_loss: 0.0305 - bb_loss: 0.0318 - pc_accuracy: 1.0000 - pc_mse:
0.0042 - pc_mae: 0.0602 - c12_accuracy: 0.9848 - c12_mse: 0.0305 - c12_mae:
0.1452 - bb_accuracy: 0.3198 - bb_mse: 0.0318 - bb_mae: 0.1545 - val_loss:
0.1051 - val_pc_loss: 0.0124 - val_c12_loss: 0.0589 - val_bb_loss: 0.0339 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0124 - val_pc_mae: 0.0711 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0589 - val_c12_mae: 0.1779 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0339 - val_bb_mae: 0.1552
Epoch 33/100
4/4 [==============================] - 3s 721ms/step - loss: 0.0643 - pc_loss:
0.0040 - c12_loss: 0.0286 - bb_loss: 0.0316 - pc_accuracy: 1.0000 - pc_mse:
0.0040 - pc_mae: 0.0591 - c12_accuracy: 0.9848 - c12_mse: 0.0286 - c12_mae:
0.1416 - bb_accuracy: 0.3249 - bb_mse: 0.0316 - bb_mae: 0.1540 - val_loss:
0.1091 - val_pc_loss: 0.0148 - val_c12_loss: 0.0602 - val_bb_loss: 0.0341 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0148 - val_pc_mae: 0.0734 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0602 - val_c12_mae: 0.1782 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0341 - val_bb_mae: 0.1553
Epoch 34/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0639 - pc_loss:
0.0041 - c12_loss: 0.0283 - bb_loss: 0.0315 - pc_accuracy: 1.0000 - pc_mse:
0.0041 - pc_mae: 0.0590 - c12_accuracy: 0.9848 - c12_mse: 0.0283 - c12_mae:
0.1406 - bb_accuracy: 0.3299 - bb_mse: 0.0315 - bb_mae: 0.1536 - val_loss:
0.1246 - val_pc_loss: 0.0280 - val_c12_loss: 0.0612 - val_bb_loss: 0.0354 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0280 - val_pc_mae: 0.0828 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0612 - val_c12_mae: 0.1772 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0354 - val_bb_mae: 0.1561
Epoch 35/100
4/4 [==============================] - 3s 717ms/step - loss: 0.0642 - pc_loss:
0.0042 - c12_loss: 0.0285 - bb_loss: 0.0315 - pc_accuracy: 1.0000 - pc_mse:
0.0042 - pc_mae: 0.0594 - c12_accuracy: 0.9848 - c12_mse: 0.0285 - c12_mae:
0.1405 - bb_accuracy: 0.3198 - bb_mse: 0.0315 - bb_mae: 0.1535 - val_loss:
0.1198 - val_pc_loss: 0.0248 - val_c12_loss: 0.0598 - val_bb_loss: 0.0352 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0248 - val_pc_mae: 0.0800 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0598 - val_c12_mae: 0.1755 -
```

```
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0352 - val_bb_mae: 0.1560
Epoch 36/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0630 - pc_loss:
0.0039 - c12_loss: 0.0279 - bb_loss: 0.0312 - pc_accuracy: 1.0000 - pc_mse:
0.0039 - pc_mae: 0.0580 - c12_accuracy: 0.9848 - c12_mse: 0.0279 - c12_mae:
0.1394 - bb_accuracy: 0.3249 - bb_mse: 0.0312 - bb_mae: 0.1524 - val_loss:
0.1390 - val_pc_loss: 0.0176 - val_c12_loss: 0.0808 - val_bb_loss: 0.0406 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0176 - val_pc_mae: 0.1135 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0808 - val_c12_mae: 0.2219 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0406 - val_bb_mae: 0.1781
Epoch 37/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0836 - pc_loss:
0.0110 - c12_loss: 0.0379 - bb_loss: 0.0348 - pc_accuracy: 0.9949 - pc_mse:
0.0110 - pc_mae: 0.0794 - c12_accuracy: 0.9746 - c12_mse: 0.0379 - c12_mae:
0.1563 - bb_accuracy: 0.3299 - bb_mse: 0.0348 - bb_mae: 0.1604 - val_loss:
0.1330 - val_pc_loss: 0.0311 - val_c12_loss: 0.0660 - val_bb_loss: 0.0359 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0311 - val_pc_mae: 0.0851 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0660 - val_c12_mae: 0.1825 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0359 - val_bb_mae: 0.1568
Epoch 38/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0639 - pc_loss:
0.0042 - c12_loss: 0.0283 - bb_loss: 0.0313 - pc_accuracy: 1.0000 - pc_mse:
0.0042 - pc_mae: 0.0591 - c12_accuracy: 0.9746 - c12_mse: 0.0283 - c12_mae:
0.1392 - bb_accuracy: 0.3299 - bb_mse: 0.0313 - bb_mae: 0.1524 - val_loss:
0.1175 - val_pc_loss: 0.0257 - val_c12_loss: 0.0573 - val_bb_loss: 0.0346 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0257 - val_pc_mae: 0.0813 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0573 - val_c12_mae: 0.1716 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0346 - val_bb_mae: 0.1540
Epoch 39/100
4/4 [==============================] - 3s 717ms/step - loss: 0.0621 - pc_loss:
0.0039 - c12_loss: 0.0272 - bb_loss: 0.0311 - pc_accuracy: 1.0000 - pc_mse:
0.0039 - pc_mae: 0.0578 - c12_accuracy: 0.9797 - c12_mse: 0.0272 - c12_mae:
0.1371 - bb_accuracy: 0.3299 - bb_mse: 0.0311 - bb_mae: 0.1520 - val_loss:
0.1384 - val_pc_loss: 0.0255 - val_c12_loss: 0.0768 - val_bb_loss: 0.0360 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0255 - val_pc_mae: 0.0814 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0768 - val_c12_mae: 0.1928 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0360 - val_bb_mae: 0.1577
Epoch 40/100
4/4 [==============================] - 3s 724ms/step - loss: 0.0628 - pc_loss:
0.0040 - c12_loss: 0.0277 - bb_loss: 0.0310 - pc_accuracy: 1.0000 - pc_mse:
0.0040 - pc_mae: 0.0585 - c12_accuracy: 0.9797 - c12_mse: 0.0277 - c12_mae:
0.1380 - bb_accuracy: 0.3299 - bb_mse: 0.0310 - bb_mae: 0.1520 - val_loss:
0.1097 - val_pc_loss: 0.0193 - val_c12_loss: 0.0568 - val_bb_loss: 0.0336 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0193 - val_pc_mae: 0.0766 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0568 - val_c12_mae: 0.1711 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0336 - val_bb_mae: 0.1517
Epoch 41/100
4/4 [==============================] - 3s 727ms/step - loss: 0.0611 - pc_loss:
```

```
0.0039 - c12_loss: 0.0263 - bb_loss: 0.0309 - pc_accuracy: 1.0000 - pc_mse:
0.0039 - pc_mae: 0.0576 - c12_accuracy: 0.9848 - c12_mse: 0.0263 - c12_mae:
0.1352 - bb_accuracy: 0.3249 - bb_mse: 0.0309 - bb_mae: 0.1512 - val_loss:
0.1074 - val_pc_loss: 0.0160 - val_c12_loss: 0.0578 - val_bb_loss: 0.0336 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0160 - val_pc_mae: 0.0731 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0578 - val_c12_mae: 0.1723 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0336 - val_bb_mae: 0.1524
Epoch 42/100
4/4 [==============================] - 3s 724ms/step - loss: 0.0603 - pc_loss:
0.0039 - c12_loss: 0.0257 - bb_loss: 0.0307 - pc_accuracy: 1.0000 - pc_mse:
0.0039 - pc_mae: 0.0573 - c12_accuracy: 0.9797 - c12_mse: 0.0257 - c12_mae:
0.1336 - bb_accuracy: 0.3299 - bb_mse: 0.0307 - bb_mae: 0.1509 - val_loss:
0.0897 - val_pc_loss: 0.0064 - val_c12_loss: 0.0523 - val_bb_loss: 0.0309 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0064 - val_pc_mae: 0.0645 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0523 - val_c12_mae: 0.1700 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0309 - val_bb_mae: 0.1479
Epoch 43/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0606 - pc_loss:
0.0036 - c12_loss: 0.0263 - bb_loss: 0.0307 - pc_accuracy: 1.0000 - pc_mse:
0.0036 - pc_mae: 0.0563 - c12_accuracy: 0.9746 - c12_mse: 0.0263 - c12_mae:
0.1348 - bb_accuracy: 0.3147 - bb_mse: 0.0307 - bb_mae: 0.1509 - val_loss:
0.1165 - val_pc_loss: 0.0238 - val_c12_loss: 0.0582 - val_bb_loss: 0.0346 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0238 - val_pc_mae: 0.0784 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0582 - val_c12_mae: 0.1707 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0346 - val_bb_mae: 0.1539
Epoch 44/100
4/4 [==============================] - 3s 714ms/step - loss: 0.0596 - pc_loss:
0.0038 - c12_loss: 0.0252 - bb_loss: 0.0306 - pc_accuracy: 1.0000 - pc_mse:
0.0038 - pc_mae: 0.0567 - c12_accuracy: 0.9797 - c12_mse: 0.0252 - c12_mae:
0.1323 - bb_accuracy: 0.3299 - bb_mse: 0.0306 - bb_mae: 0.1506 - val_loss:
0.1048 - val_pc_loss: 0.0170 - val_c12_loss: 0.0547 - val_bb_loss: 0.0331 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0170 - val_pc_mae: 0.0731 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0547 - val_c12_mae: 0.1677 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0331 - val_bb_mae: 0.1508
Epoch 45/100
4/4 [==============================] - 3s 721ms/step - loss: 0.0587 - pc_loss:
0.0036 - c12_loss: 0.0248 - bb_loss: 0.0303 - pc_accuracy: 1.0000 - pc_mse:
0.0036 - pc_mae: 0.0560 - c12_accuracy: 0.9898 - c12_mse: 0.0248 - c12_mae:
0.1317 - bb_accuracy: 0.3249 - bb_mse: 0.0303 - bb_mae: 0.1499 - val_loss:
0.1094 - val_pc_loss: 0.0177 - val_c12_loss: 0.0583 - val_bb_loss: 0.0335 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0177 - val_pc_mae: 0.0734 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0583 - val_c12_mae: 0.1709 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0335 - val_bb_mae: 0.1522
Epoch 46/100
4/4 [==============================] - 3s 720ms/step - loss: 0.0583 - pc_loss:
0.0037 - c12_loss: 0.0243 - bb_loss: 0.0303 - pc_accuracy: 1.0000 - pc_mse:
0.0037 - pc_mae: 0.0560 - c12_accuracy: 0.9898 - c12_mse: 0.0243 - c12_mae:
0.1305 - bb_accuracy: 0.3299 - bb_mse: 0.0303 - bb_mae: 0.1497 - val_loss:
```

```
0.5739 - val_pc_loss: 0.3192 - val_c12_loss: 0.1550 - val_bb_loss: 0.0997 -
val_pc_accuracy: 0.5909 - val_pc_mse: 0.3192 - val_pc_mae: 0.3946 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.1550 - val_c12_mae: 0.3395 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0997 - val_bb_mae: 0.2652
Epoch 47/100
4/4 [==============================] - 3s 718ms/step - loss: 0.2013 - pc_loss:
0.0883 - c12_loss: 0.0627 - bb_loss: 0.0503 - pc_accuracy: 0.8883 - pc_mse:
0.0883 - pc_mae: 0.1556 - c12_accuracy: 0.9594 - c12_mse: 0.0627 - c12_mae:
0.1965 - bb_accuracy: 0.3299 - bb_mse: 0.0503 - bb_mae: 0.1833 - val_loss:
0.1456 - val_pc_loss: 0.0338 - val_c12_loss: 0.0760 - val_bb_loss: 0.0359 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0338 - val_pc_mae: 0.0852 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0760 - val_c12_mae: 0.1918 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0359 - val_bb_mae: 0.1559
Epoch 48/100
4/4 [==============================] - 3s 716ms/step - loss: 0.0623 - pc_loss:
0.0043 - c12_loss: 0.0275 - bb_loss: 0.0305 - pc_accuracy: 1.0000 - pc_mse:
0.0043 - pc_mae: 0.0586 - c12_accuracy: 0.9797 - c12_mse: 0.0275 - c12_mae:
0.1365 - bb_accuracy: 0.3299 - bb_mse: 0.0305 - bb_mae: 0.1503 - val_loss:
0.1895 - val_pc_loss: 0.0283 - val_c12_loss: 0.1227 - val_bb_loss: 0.0385 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0283 - val_pc_mae: 0.0867 -
val_c12_accuracy: 0.7727 - val_c12_mse: 0.1227 - val_c12_mae: 0.2312 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0385 - val_bb_mae: 0.1680
Epoch 49/100
4/4 [==============================] - 3s 714ms/step - loss: 0.0712 - pc_loss:
0.0044 - c12_loss: 0.0364 - bb_loss: 0.0304 - pc_accuracy: 1.0000 - pc_mse:
0.0044 - pc_mae: 0.0593 - c12_accuracy: 0.9695 - c12_mse: 0.0364 - c12_mae:
0.1483 - bb_accuracy: 0.3147 - bb_mse: 0.0304 - bb_mae: 0.1500 - val_loss:
0.1153 - val_pc_loss: 0.0244 - val_c12_loss: 0.0575 - val_bb_loss: 0.0334 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0244 - val_pc_mae: 0.0783 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0575 - val_c12_mae: 0.1699 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0334 - val_bb_mae: 0.1505
Epoch 50/100
4/4 [==============================] - 3s 717ms/step - loss: 0.0582 - pc_loss:
0.0037 - c12_loss: 0.0245 - bb_loss: 0.0300 - pc_accuracy: 1.0000 - pc_mse:
0.0037 - pc_mae: 0.0559 - c12_accuracy: 0.9898 - c12_mse: 0.0245 - c12_mae:
0.1301 - bb_accuracy: 0.3147 - bb_mse: 0.0300 - bb_mae: 0.1487 - val_loss:
0.1221 - val_pc_loss: 0.0251 - val_c12_loss: 0.0627 - val_bb_loss: 0.0343 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0251 - val_pc_mae: 0.0780 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0627 - val_c12_mae: 0.1747 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0343 - val_bb_mae: 0.1527
Epoch 51/100
4/4 [==============================] - 3s 717ms/step - loss: 0.0579 - pc_loss:
0.0037 - c12_loss: 0.0243 - bb_loss: 0.0299 - pc_accuracy: 1.0000 - pc_mse:
0.0037 - pc_mae: 0.0556 - c12_accuracy: 0.9898 - c12_mse: 0.0243 - c12_mae:
0.1296 - bb_accuracy: 0.3249 - bb_mse: 0.0299 - bb_mae: 0.1485 - val_loss:
0.1153 - val_pc_loss: 0.0246 - val_c12_loss: 0.0569 - val_bb_loss: 0.0338 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0246 - val_pc_mae: 0.0774 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0569 - val_c12_mae: 0.1673 -
```

```
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0338 - val_bb_mae: 0.1520
Epoch 52/100
4/4 [==============================] - 3s 720ms/step - loss: 0.0567 - pc_loss:
0.0036 - c12_loss: 0.0235 - bb_loss: 0.0297 - pc_accuracy: 1.0000 - pc_mse:
0.0036 - pc_mae: 0.0552 - c12_accuracy: 0.9898 - c12_mse: 0.0235 - c12_mae:
0.1279 - bb_accuracy: 0.3299 - bb_mse: 0.0297 - bb_mae: 0.1479 - val_loss:
0.1388 - val_pc_loss: 0.0405 - val_c12_loss: 0.0618 - val_bb_loss: 0.0365 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0405 - val_pc_mae: 0.0867 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0618 - val_c12_mae: 0.1727 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0365 - val_bb_mae: 0.1542
Epoch 53/100
4/4 [==============================] - 3s 720ms/step - loss: 0.0840 - pc_loss:
0.0146 - c12_loss: 0.0369 - bb_loss: 0.0325 - pc_accuracy: 0.9898 - pc_mse:
0.0146 - pc_mae: 0.0838 - c12_accuracy: 0.9695 - c12_mse: 0.0369 - c12_mae:
0.1510 - bb_accuracy: 0.3553 - bb_mse: 0.0325 - bb_mae: 0.1551 - val_loss:
0.1444 - val_pc_loss: 0.0429 - val_c12_loss: 0.0668 - val_bb_loss: 0.0347 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0429 - val_pc_mae: 0.0920 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0668 - val_c12_mae: 0.1831 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0347 - val_bb_mae: 0.1493
Epoch 54/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0633 - pc_loss:
0.0057 - c12_loss: 0.0274 - bb_loss: 0.0301 - pc_accuracy: 0.9949 - pc_mse:
0.0057 - pc_mae: 0.0600 - c12_accuracy: 0.9848 - c12_mse: 0.0274 - c12_mae:
0.1335 - bb_accuracy: 0.3299 - bb_mse: 0.0301 - bb_mae: 0.1490 - val_loss:
0.1241 - val_pc_loss: 0.0068 - val_c12_loss: 0.0842 - val_bb_loss: 0.0331 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0068 - val_pc_mae: 0.0653 -
val_c12_accuracy: 0.8182 - val_c12_mse: 0.0842 - val_c12_mae: 0.2004 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0331 - val_bb_mae: 0.1542
Epoch 55/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0609 - pc_loss:
0.0037 - c12_loss: 0.0273 - bb_loss: 0.0300 - pc_accuracy: 1.0000 - pc_mse:
0.0037 - pc_mae: 0.0557 - c12_accuracy: 0.9797 - c12_mse: 0.0273 - c12_mae:
0.1349 - bb_accuracy: 0.3147 - bb_mse: 0.0300 - bb_mae: 0.1488 - val_loss:
0.1315 - val_pc_loss: 0.0119 - val_c12_loss: 0.0905 - val_bb_loss: 0.0291 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0119 - val_pc_mae: 0.0729 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0905 - val_c12_mae: 0.1997 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0291 - val_bb_mae: 0.1420
Epoch 56/100
4/4 [==============================] - 3s 724ms/step - loss: 0.0706 - pc_loss:
0.0036 - c12_loss: 0.0373 - bb_loss: 0.0296 - pc_accuracy: 1.0000 - pc_mse:
0.0036 - pc_mae: 0.0554 - c12_accuracy: 0.9746 - c12_mse: 0.0373 - c12_mae:
0.1494 - bb_accuracy: 0.2944 - bb_mse: 0.0296 - bb_mae: 0.1474 - val_loss:
0.0849 - val_pc_loss: 0.0063 - val_c12_loss: 0.0484 - val_bb_loss: 0.0302 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0063 - val_pc_mae: 0.0731 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0484 - val_c12_mae: 0.1702 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0302 - val_bb_mae: 0.1523
Epoch 57/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0579 - pc_loss:
```

```
0.0042 - c12_loss: 0.0235 - bb_loss: 0.0303 - pc_accuracy: 1.0000 - pc_mse:
0.0042 - pc_mae: 0.0596 - c12_accuracy: 0.9848 - c12_mse: 0.0235 - c12_mae:
0.1287 - bb_accuracy: 0.3198 - bb_mse: 0.0303 - bb_mae: 0.1507 - val_loss:
0.0937 - val_pc_loss: 0.0090 - val_c12_loss: 0.0536 - val_bb_loss: 0.0310 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0090 - val_pc_mae: 0.0644 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0536 - val_c12_mae: 0.1648 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0310 - val_bb_mae: 0.1463
Epoch 58/100
4/4 [==============================] - 3s 720ms/step - loss: 0.0550 - pc_loss:
0.0035 - c12_loss: 0.0220 - bb_loss: 0.0295 - pc_accuracy: 1.0000 - pc_mse:
0.0035 - pc_mae: 0.0552 - c12_accuracy: 0.9898 - c12_mse: 0.0220 - c12_mae:
0.1241 - bb_accuracy: 0.3198 - bb_mse: 0.0295 - bb_mae: 0.1471 - val_loss:
0.1386 - val_pc_loss: 0.0109 - val_c12_loss: 0.0943 - val_bb_loss: 0.0334 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0109 - val_pc_mae: 0.0672 -
val_c12_accuracy: 0.8182 - val_c12_mse: 0.0943 - val_c12_mae: 0.2041 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0334 - val_bb_mae: 0.1543
Epoch 59/100
4/4 [==============================] - 3s 716ms/step - loss: 0.0578 - pc_loss:
0.0035 - c12_loss: 0.0245 - bb_loss: 0.0298 - pc_accuracy: 1.0000 - pc_mse:
0.0035 - pc_mae: 0.0546 - c12_accuracy: 0.9898 - c12_mse: 0.0245 - c12_mae:
0.1299 - bb_accuracy: 0.3249 - bb_mse: 0.0298 - bb_mae: 0.1488 - val_loss:
0.0814 - val_pc_loss: 0.0054 - val_c12_loss: 0.0477 - val_bb_loss: 0.0283 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0054 - val_pc_mae: 0.0593 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0477 - val_c12_mae: 0.1595 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0283 - val_bb_mae: 0.1410
Epoch 60/100
4/4 [==============================] - 3s 719ms/step - loss: 0.0546 - pc_loss:
0.0036 - c12_loss: 0.0218 - bb_loss: 0.0292 - pc_accuracy: 1.0000 - pc_mse:
0.0036 - pc_mae: 0.0549 - c12_accuracy: 0.9898 - c12_mse: 0.0218 - c12_mae:
0.1232 - bb_accuracy: 0.3096 - bb_mse: 0.0292 - bb_mae: 0.1465 - val_loss:
0.0891 - val_pc_loss: 0.0106 - val_c12_loss: 0.0475 - val_bb_loss: 0.0310 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0106 - val_pc_mae: 0.0641 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0475 - val_c12_mae: 0.1544 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0310 - val_bb_mae: 0.1473
Epoch 61/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0532 - pc_loss:
0.0034 - c12_loss: 0.0206 - bb_loss: 0.0292 - pc_accuracy: 1.0000 - pc_mse:
0.0034 - pc_mae: 0.0536 - c12_accuracy: 0.9898 - c12_mse: 0.0206 - c12_mae:
0.1208 - bb_accuracy: 0.3249 - bb_mse: 0.0292 - bb_mae: 0.1466 - val_loss:
0.0959 - val_pc_loss: 0.0132 - val_c12_loss: 0.0508 - val_bb_loss: 0.0319 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0132 - val_pc_mae: 0.0666 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0508 - val_c12_mae: 0.1566 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0319 - val_bb_mae: 0.1490
Epoch 62/100
4/4 [==============================] - 3s 716ms/step - loss: 0.0527 - pc_loss:
0.0034 - c12_loss: 0.0203 - bb_loss: 0.0291 - pc_accuracy: 1.0000 - pc_mse:
0.0034 - pc_mae: 0.0534 - c12_accuracy: 0.9898 - c12_mse: 0.0203 - c12_mae:
0.1198 - bb_accuracy: 0.3249 - bb_mse: 0.0291 - bb_mae: 0.1461 - val_loss:
```

```
0.0823 - val_pc_loss: 0.0056 - val_c12_loss: 0.0470 - val_bb_loss: 0.0297 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0056 - val_pc_mae: 0.0602 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0470 - val_c12_mae: 0.1559 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0297 - val_bb_mae: 0.1463
Epoch 63/100
4/4 [==============================] - 3s 719ms/step - loss: 0.0531 - pc_loss:
0.0034 - c12_loss: 0.0206 - bb_loss: 0.0291 - pc_accuracy: 1.0000 - pc_mse:
0.0034 - pc_mae: 0.0542 - c12_accuracy: 0.9848 - c12_mse: 0.0206 - c12_mae:
0.1206 - bb_accuracy: 0.3299 - bb_mse: 0.0291 - bb_mae: 0.1463 - val_loss:
0.0877 - val_pc_loss: 0.0119 - val_c12_loss: 0.0451 - val_bb_loss: 0.0307 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0119 - val_pc_mae: 0.0660 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0451 - val_c12_mae: 0.1505 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0307 - val_bb_mae: 0.1457
Epoch 64/100
4/4 [==============================] - 3s 719ms/step - loss: 0.0522 - pc_loss:
0.0034 - c12_loss: 0.0199 - bb_loss: 0.0290 - pc_accuracy: 1.0000 - pc_mse:
0.0034 - pc_mae: 0.0533 - c12_accuracy: 0.9898 - c12_mse: 0.0199 - c12_mae:
0.1187 - bb_accuracy: 0.3249 - bb_mse: 0.0290 - bb_mae: 0.1457 - val_loss:
0.0789 - val_pc_loss: 0.0073 - val_c12_loss: 0.0419 - val_bb_loss: 0.0296 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0073 - val_pc_mae: 0.0606 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0419 - val_c12_mae: 0.1473 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0296 - val_bb_mae: 0.1425
Epoch 65/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0517 - pc_loss:
0.0034 - c12_loss: 0.0196 - bb_loss: 0.0288 - pc_accuracy: 1.0000 - pc_mse:
0.0034 - pc_mae: 0.0535 - c12_accuracy: 0.9898 - c12_mse: 0.0196 - c12_mae:
0.1179 - bb_accuracy: 0.3249 - bb_mse: 0.0288 - bb_mae: 0.1451 - val_loss:
0.0894 - val_pc_loss: 0.0133 - val_c12_loss: 0.0457 - val_bb_loss: 0.0304 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0133 - val_pc_mae: 0.0672 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0457 - val_c12_mae: 0.1517 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0304 - val_bb_mae: 0.1433
Epoch 66/100
4/4 [==============================] - 3s 725ms/step - loss: 0.0515 - pc_loss:
0.0034 - c12_loss: 0.0194 - bb_loss: 0.0287 - pc_accuracy: 1.0000 - pc_mse:
0.0034 - pc_mae: 0.0535 - c12_accuracy: 0.9898 - c12_mse: 0.0194 - c12_mae:
0.1172 - bb_accuracy: 0.3198 - bb_mse: 0.0287 - bb_mae: 0.1448 - val_loss:
0.0923 - val_pc_loss: 0.0127 - val_c12_loss: 0.0484 - val_bb_loss: 0.0311 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0127 - val_pc_mae: 0.0654 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0484 - val_c12_mae: 0.1528 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0311 - val_bb_mae: 0.1461
Epoch 67/100
4/4 [==============================] - 3s 717ms/step - loss: 0.0509 - pc_loss:
0.0033 - c12_loss: 0.0190 - bb_loss: 0.0286 - pc_accuracy: 1.0000 - pc_mse:
0.0033 - pc_mae: 0.0528 - c12_accuracy: 0.9898 - c12_mse: 0.0190 - c12_mae:
0.1160 - bb_accuracy: 0.3299 - bb_mse: 0.0286 - bb_mae: 0.1444 - val_loss:
0.0989 - val_pc_loss: 0.0126 - val_c12_loss: 0.0549 - val_bb_loss: 0.0314 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0126 - val_pc_mae: 0.0657 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0549 - val_c12_mae: 0.1600 -
```

```
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0314 - val_bb_mae: 0.1468
Epoch 68/100
4/4 [==============================] - 3s 714ms/step - loss: 0.0510 - pc_loss:
0.0033 - c12_loss: 0.0193 - bb_loss: 0.0284 - pc_accuracy: 1.0000 - pc_mse:
0.0033 - pc_mae: 0.0528 - c12_accuracy: 0.9898 - c12_mse: 0.0193 - c12_mae:
0.1166 - bb_accuracy: 0.3299 - bb_mse: 0.0284 - bb_mae: 0.1439 - val_loss:
0.0849 - val_pc_loss: 0.0079 - val_c12_loss: 0.0470 - val_bb_loss: 0.0300 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0079 - val_pc_mae: 0.0602 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0470 - val_c12_mae: 0.1512 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0300 - val_bb_mae: 0.1444
Epoch 69/100
4/4 [==============================] - 3s 719ms/step - loss: 0.0505 - pc_loss:
0.0033 - c12_loss: 0.0188 - bb_loss: 0.0284 - pc_accuracy: 1.0000 - pc_mse:
0.0033 - pc_mae: 0.0527 - c12_accuracy: 0.9898 - c12_mse: 0.0188 - c12_mae:
0.1152 - bb_accuracy: 0.3299 - bb_mse: 0.0284 - bb_mae: 0.1438 - val_loss:
0.0900 - val_pc_loss: 0.0110 - val_c12_loss: 0.0483 - val_bb_loss: 0.0307 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0110 - val_pc_mae: 0.0633 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0483 - val_c12_mae: 0.1519 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0307 - val_bb_mae: 0.1452
Epoch 70/100
4/4 [==============================] - 3s 721ms/step - loss: 0.0500 - pc_loss:
0.0032 - c12_loss: 0.0185 - bb_loss: 0.0283 - pc_accuracy: 1.0000 - pc_mse:
0.0032 - pc_mae: 0.0522 - c12_accuracy: 0.9898 - c12_mse: 0.0185 - c12_mae:
0.1146 - bb_accuracy: 0.3299 - bb_mse: 0.0283 - bb_mae: 0.1434 - val_loss:
0.0862 - val_pc_loss: 0.0115 - val_c12_loss: 0.0448 - val_bb_loss: 0.0299 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0115 - val_pc_mae: 0.0646 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0448 - val_c12_mae: 0.1492 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0299 - val_bb_mae: 0.1414
Epoch 71/100
4/4 [==============================] - 3s 725ms/step - loss: 0.0498 - pc_loss:
0.0032 - c12_loss: 0.0185 - bb_loss: 0.0281 - pc_accuracy: 1.0000 - pc_mse:
0.0032 - pc_mae: 0.0523 - c12_accuracy: 0.9898 - c12_mse: 0.0185 - c12_mae:
0.1142 - bb_accuracy: 0.3198 - bb_mse: 0.0281 - bb_mae: 0.1428 - val_loss:
0.0935 - val_pc_loss: 0.0116 - val_c12_loss: 0.0511 - val_bb_loss: 0.0308 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0116 - val_pc_mae: 0.0637 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0511 - val_c12_mae: 0.1546 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0308 - val_bb_mae: 0.1453
Epoch 72/100
4/4 [==============================] - 3s 720ms/step - loss: 0.0495 - pc_loss:
0.0032 - c12_loss: 0.0183 - bb_loss: 0.0280 - pc_accuracy: 1.0000 - pc_mse:
0.0032 - pc_mae: 0.0521 - c12_accuracy: 0.9898 - c12_mse: 0.0183 - c12_mae:
0.1136 - bb_accuracy: 0.3299 - bb_mse: 0.0280 - bb_mae: 0.1428 - val_loss:
0.0910 - val_pc_loss: 0.0122 - val_c12_loss: 0.0483 - val_bb_loss: 0.0306 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0122 - val_pc_mae: 0.0640 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0483 - val_c12_mae: 0.1512 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0306 - val_bb_mae: 0.1443
Epoch 73/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0490 - pc_loss:
```

```
0.0032 - c12_loss: 0.0178 - bb_loss: 0.0280 - pc_accuracy: 1.0000 - pc_mse:
0.0032 - pc_mae: 0.0517 - c12_accuracy: 0.9898 - c12_mse: 0.0178 - c12_mae:
0.1124 - bb_accuracy: 0.3299 - bb_mse: 0.0280 - bb_mae: 0.1424 - val_loss:
0.0780 - val_pc_loss: 0.0065 - val_c12_loss: 0.0426 - val_bb_loss: 0.0289 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0065 - val_pc_mae: 0.0577 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0426 - val_c12_mae: 0.1463 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0289 - val_bb_mae: 0.1405
Epoch 74/100
4/4 [==============================] - 3s 721ms/step - loss: 0.0487 - pc_loss:
0.0031 - c12_loss: 0.0177 - bb_loss: 0.0278 - pc_accuracy: 1.0000 - pc_mse:
0.0031 - pc_mae: 0.0517 - c12_accuracy: 0.9898 - c12_mse: 0.0177 - c12_mae:
0.1123 - bb_accuracy: 0.3350 - bb_mse: 0.0278 - bb_mae: 0.1421 - val_loss:
0.0924 - val_pc_loss: 0.0090 - val_c12_loss: 0.0531 - val_bb_loss: 0.0303 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0090 - val_pc_mae: 0.0608 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0531 - val_c12_mae: 0.1573 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0303 - val_bb_mae: 0.1436
Epoch 75/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0486 - pc_loss:
0.0032 - c12_loss: 0.0176 - bb_loss: 0.0279 - pc_accuracy: 1.0000 - pc_mse:
0.0032 - pc_mae: 0.0519 - c12_accuracy: 0.9898 - c12_mse: 0.0176 - c12_mae:
0.1116 - bb_accuracy: 0.3249 - bb_mse: 0.0279 - bb_mae: 0.1422 - val_loss:
0.0902 - val_pc_loss: 0.0125 - val_c12_loss: 0.0472 - val_bb_loss: 0.0304 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0125 - val_pc_mae: 0.0641 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0472 - val_c12_mae: 0.1498 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0304 - val_bb_mae: 0.1429
Epoch 76/100
4/4 [==============================] - 3s 723ms/step - loss: 0.0483 - pc_loss:
0.0032 - c12_loss: 0.0174 - bb_loss: 0.0277 - pc_accuracy: 1.0000 - pc_mse:
0.0032 - pc_mae: 0.0516 - c12_accuracy: 0.9898 - c12_mse: 0.0174 - c12_mae:
0.1110 - bb_accuracy: 0.3350 - bb_mse: 0.0277 - bb_mae: 0.1418 - val_loss:
0.0787 - val_pc_loss: 0.0070 - val_c12_loss: 0.0430 - val_bb_loss: 0.0287 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0070 - val_pc_mae: 0.0583 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0430 - val_c12_mae: 0.1467 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0287 - val_bb_mae: 0.1400
Epoch 77/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0479 - pc_loss:
0.0031 - c12_loss: 0.0173 - bb_loss: 0.0275 - pc_accuracy: 1.0000 - pc_mse:
0.0031 - pc_mae: 0.0513 - c12_accuracy: 0.9898 - c12_mse: 0.0173 - c12_mae:
0.1110 - bb_accuracy: 0.3249 - bb_mse: 0.0275 - bb_mae: 0.1413 - val_loss:
0.0859 - val_pc_loss: 0.0097 - val_c12_loss: 0.0466 - val_bb_loss: 0.0296 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0097 - val_pc_mae: 0.0610 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0466 - val_c12_mae: 0.1496 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0296 - val_bb_mae: 0.1419
Epoch 78/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0476 - pc_loss:
0.0031 - c12_loss: 0.0171 - bb_loss: 0.0274 - pc_accuracy: 1.0000 - pc_mse:
0.0031 - pc_mae: 0.0510 - c12_accuracy: 0.9898 - c12_mse: 0.0171 - c12_mae:
0.1102 - bb_accuracy: 0.3350 - bb_mse: 0.0274 - bb_mae: 0.1410 - val_loss:
```

```
0.1110 - val_pc_loss: 0.0287 - val_c12_loss: 0.0496 - val_bb_loss: 0.0326 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0287 - val_pc_mae: 0.1361 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0496 - val_c12_mae: 0.1766 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0326 - val_bb_mae: 0.1619
Epoch 79/100
4/4 [==============================] - 3s 718ms/step - loss: 0.2939 - pc_loss:
0.1059 - c12_loss: 0.1319 - bb_loss: 0.0561 - pc_accuracy: 0.8883 - pc_mse:
0.1059 - pc_mae: 0.1904 - c12_accuracy: 0.7259 - c12_mse: 0.1319 - c12_mae:
0.2546 - bb_accuracy: 0.3452 - bb_mse: 0.0561 - bb_mae: 0.1932 - val_loss:
0.1625 - val_pc_loss: 0.0420 - val_c12_loss: 0.0874 - val_bb_loss: 0.0332 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0420 - val_pc_mae: 0.0938 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0874 - val_c12_mae: 0.2159 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0332 - val_bb_mae: 0.1461
Epoch 80/100
4/4 [==============================] - 3s 723ms/step - loss: 0.0835 - pc_loss:
0.0089 - c12_loss: 0.0478 - bb_loss: 0.0268 - pc_accuracy: 0.9898 - pc_mse:
0.0089 - pc_mae: 0.0628 - c12_accuracy: 0.9645 - c12_mse: 0.0478 - c12_mae:
0.1664 - bb_accuracy: 0.3350 - bb_mse: 0.0268 - bb_mae: 0.1386 - val_loss:
0.1214 - val_pc_loss: 0.0226 - val_c12_loss: 0.0680 - val_bb_loss: 0.0308 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0226 - val_pc_mae: 0.0778 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0680 - val_c12_mae: 0.1804 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0308 - val_bb_mae: 0.1435
Epoch 81/100
4/4 [==============================] - 3s 719ms/step - loss: 0.0527 - pc_loss:
0.0034 - c12_loss: 0.0227 - bb_loss: 0.0266 - pc_accuracy: 1.0000 - pc_mse:
0.0034 - pc_mae: 0.0541 - c12_accuracy: 0.9898 - c12_mse: 0.0227 - c12_mae:
0.1247 - bb_accuracy: 0.3401 - bb_mse: 0.0266 - bb_mae: 0.1394 - val_loss:
0.1332 - val_pc_loss: 0.0291 - val_c12_loss: 0.0719 - val_bb_loss: 0.0322 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0291 - val_pc_mae: 0.0807 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0719 - val_c12_mae: 0.1797 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0322 - val_bb_mae: 0.1444
Epoch 82/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0500 - pc_loss:
0.0033 - c12_loss: 0.0199 - bb_loss: 0.0268 - pc_accuracy: 1.0000 - pc_mse:
0.0033 - pc_mae: 0.0532 - c12_accuracy: 0.9898 - c12_mse: 0.0199 - c12_mae:
0.1180 - bb_accuracy: 0.3299 - bb_mse: 0.0268 - bb_mae: 0.1396 - val_loss:
0.1091 - val_pc_loss: 0.0212 - val_c12_loss: 0.0579 - val_bb_loss: 0.0300 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0212 - val_pc_mae: 0.0744 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0579 - val_c12_mae: 0.1650 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0300 - val_bb_mae: 0.1405
Epoch 83/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0491 - pc_loss:
0.0033 - c12_loss: 0.0191 - bb_loss: 0.0267 - pc_accuracy: 1.0000 - pc_mse:
0.0033 - pc_mae: 0.0526 - c12_accuracy: 0.9898 - c12_mse: 0.0191 - c12_mae:
0.1162 - bb_accuracy: 0.3503 - bb_mse: 0.0267 - bb_mae: 0.1395 - val_loss:
0.1028 - val_pc_loss: 0.0172 - val_c12_loss: 0.0576 - val_bb_loss: 0.0281 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0172 - val_pc_mae: 0.0732 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0576 - val_c12_mae: 0.1609 -
```

```
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0281 - val_bb_mae: 0.1352
Epoch 84/100
4/4 [==============================] - 3s 726ms/step - loss: 0.0512 - pc_loss:
0.0032 - c12_loss: 0.0215 - bb_loss: 0.0266 - pc_accuracy: 1.0000 - pc_mse:
0.0032 - pc_mae: 0.0520 - c12_accuracy: 0.9848 - c12_mse: 0.0215 - c12_mae:
0.1192 - bb_accuracy: 0.3401 - bb_mse: 0.0266 - bb_mae: 0.1388 - val_loss:
0.1024 - val_pc_loss: 0.0242 - val_c12_loss: 0.0478 - val_bb_loss: 0.0304 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0242 - val_pc_mae: 0.0752 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0478 - val_c12_mae: 0.1505 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0304 - val_bb_mae: 0.1393
Epoch 85/100
4/4 [==============================] - 3s 720ms/step - loss: 0.0482 - pc_loss:
0.0032 - c12_loss: 0.0183 - bb_loss: 0.0267 - pc_accuracy: 1.0000 - pc_mse:
0.0032 - pc_mae: 0.0516 - c12_accuracy: 0.9898 - c12_mse: 0.0183 - c12_mae:
0.1130 - bb_accuracy: 0.3299 - bb_mse: 0.0267 - bb_mae: 0.1393 - val_loss:
0.1126 - val_pc_loss: 0.0177 - val_c12_loss: 0.0643 - val_bb_loss: 0.0305 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0177 - val_pc_mae: 0.0691 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0643 - val_c12_mae: 0.1692 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0305 - val_bb_mae: 0.1431
Epoch 86/100
4/4 [==============================] - 3s 726ms/step - loss: 0.0475 - pc_loss:
0.0031 - c12_loss: 0.0179 - bb_loss: 0.0266 - pc_accuracy: 1.0000 - pc_mse:
0.0031 - pc_mae: 0.0510 - c12_accuracy: 0.9898 - c12_mse: 0.0179 - c12_mae:
0.1124 - bb_accuracy: 0.3401 - bb_mse: 0.0266 - bb_mae: 0.1388 - val_loss:
0.0971 - val_pc_loss: 0.0224 - val_c12_loss: 0.0445 - val_bb_loss: 0.0302 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0224 - val_pc_mae: 0.0726 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0445 - val_c12_mae: 0.1450 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0302 - val_bb_mae: 0.1403
Epoch 87/100
4/4 [==============================] - 3s 717ms/step - loss: 0.0466 - pc_loss:
0.0031 - c12_loss: 0.0169 - bb_loss: 0.0266 - pc_accuracy: 1.0000 - pc_mse:
0.0031 - pc_mae: 0.0510 - c12_accuracy: 0.9898 - c12_mse: 0.0169 - c12_mae:
0.1097 - bb_accuracy: 0.3401 - bb_mse: 0.0266 - bb_mae: 0.1388 - val_loss:
0.0990 - val_pc_loss: 0.0192 - val_c12_loss: 0.0497 - val_bb_loss: 0.0301 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0192 - val_pc_mae: 0.0694 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0497 - val_c12_mae: 0.1517 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0301 - val_bb_mae: 0.1414
Epoch 88/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0461 - pc_loss:
0.0030 - c12_loss: 0.0166 - bb_loss: 0.0264 - pc_accuracy: 1.0000 - pc_mse:
0.0030 - pc_mae: 0.0506 - c12_accuracy: 0.9898 - c12_mse: 0.0166 - c12_mae:
0.1088 - bb_accuracy: 0.3401 - bb_mse: 0.0264 - bb_mae: 0.1385 - val_loss:
0.0783 - val_pc_loss: 0.0101 - val_c12_loss: 0.0402 - val_bb_loss: 0.0280 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0101 - val_pc_mae: 0.0615 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0402 - val_c12_mae: 0.1418 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0280 - val_bb_mae: 0.1363
Epoch 89/100
4/4 [==============================] - 3s 725ms/step - loss: 0.0458 - pc_loss:
```

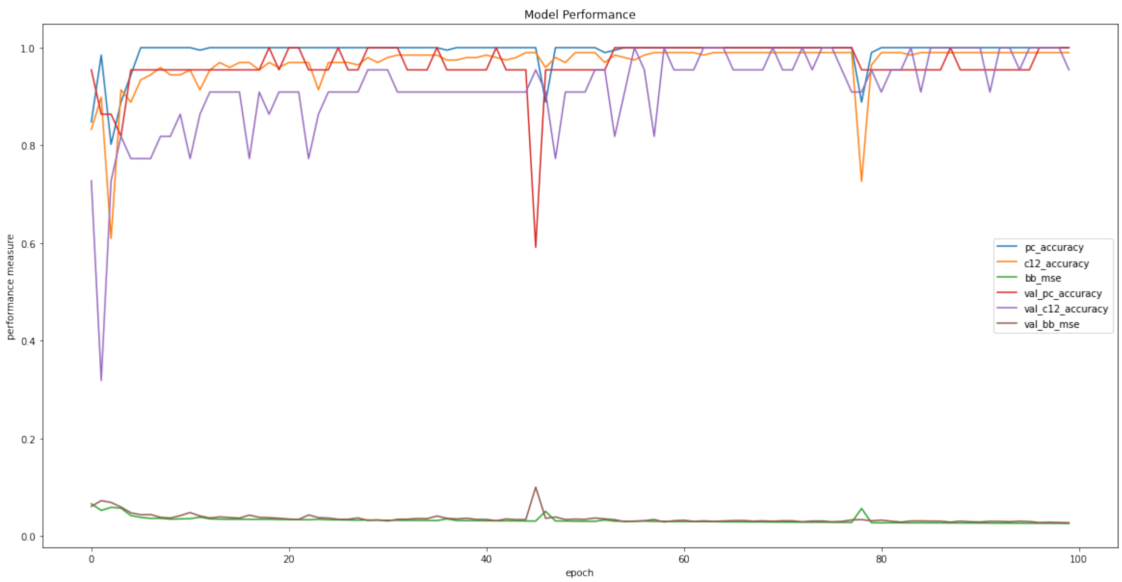
```
0.0030 - c12_loss: 0.0164 - bb_loss: 0.0264 - pc_accuracy: 1.0000 - pc_mse:
0.0030 - pc_mae: 0.0503 - c12_accuracy: 0.9898 - c12_mse: 0.0164 - c12_mae:
0.1082 - bb_accuracy: 0.3401 - bb_mse: 0.0264 - bb_mae: 0.1382 - val_loss:
0.0992 - val_pc_loss: 0.0194 - val_c12_loss: 0.0498 - val_bb_loss: 0.0301 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0194 - val_pc_mae: 0.0689 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0498 - val_c12_mae: 0.1512 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0301 - val_bb_mae: 0.1410
Epoch 90/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0453 - pc_loss:
0.0030 - c12_loss: 0.0161 - bb_loss: 0.0263 - pc_accuracy: 1.0000 - pc_mse:
0.0030 - pc_mae: 0.0501 - c12_accuracy: 0.9898 - c12_mse: 0.0161 - c12_mae:
0.1071 - bb_accuracy: 0.3401 - bb_mse: 0.0263 - bb_mae: 0.1380 - val_loss:
0.0868 - val_pc_loss: 0.0147 - val_c12_loss: 0.0432 - val_bb_loss: 0.0289 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0147 - val_pc_mae: 0.0652 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0432 - val_c12_mae: 0.1442 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0289 - val_bb_mae: 0.1379
Epoch 91/100
4/4 [==============================] - 3s 722ms/step - loss: 0.0451 - pc_loss:
0.0030 - c12_loss: 0.0159 - bb_loss: 0.0262 - pc_accuracy: 1.0000 - pc_mse:
0.0030 - pc_mae: 0.0500 - c12_accuracy: 0.9898 - c12_mse: 0.0159 - c12_mae:
0.1066 - bb_accuracy: 0.3401 - bb_mse: 0.0262 - bb_mae: 0.1378 - val_loss:
0.0835 - val_pc_loss: 0.0143 - val_c12_loss: 0.0408 - val_bb_loss: 0.0284 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0143 - val_pc_mae: 0.0649 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0408 - val_c12_mae: 0.1411 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0284 - val_bb_mae: 0.1366
Epoch 92/100
4/4 [==============================] - 3s 723ms/step - loss: 0.0448 - pc_loss:
0.0029 - c12_loss: 0.0158 - bb_loss: 0.0261 - pc_accuracy: 1.0000 - pc_mse:
0.0029 - pc_mae: 0.0498 - c12_accuracy: 0.9898 - c12_mse: 0.0158 - c12_mae:
0.1061 - bb_accuracy: 0.3401 - bb_mse: 0.0261 - bb_mae: 0.1373 - val_loss:
0.1047 - val_pc_loss: 0.0182 - val_c12_loss: 0.0568 - val_bb_loss: 0.0298 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0182 - val_pc_mae: 0.0673 -
val_c12_accuracy: 0.9091 - val_c12_mse: 0.0568 - val_c12_mae: 0.1575 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0298 - val_bb_mae: 0.1412
Epoch 93/100
4/4 [==============================] - 3s 717ms/step - loss: 0.0448 - pc_loss:
0.0029 - c12_loss: 0.0159 - bb_loss: 0.0260 - pc_accuracy: 1.0000 - pc_mse:
0.0029 - pc_mae: 0.0495 - c12_accuracy: 0.9898 - c12_mse: 0.0159 - c12_mae:
0.1064 - bb_accuracy: 0.3452 - bb_mse: 0.0260 - bb_mae: 0.1372 - val_loss:
0.0956 - val_pc_loss: 0.0178 - val_c12_loss: 0.0481 - val_bb_loss: 0.0296 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0178 - val_pc_mae: 0.0672 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0481 - val_c12_mae: 0.1486 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0296 - val_bb_mae: 0.1399
Epoch 94/100
4/4 [==============================] - 3s 723ms/step - loss: 0.0442 - pc_loss:
0.0029 - c12_loss: 0.0153 - bb_loss: 0.0259 - pc_accuracy: 1.0000 - pc_mse:
0.0029 - pc_mae: 0.0495 - c12_accuracy: 0.9898 - c12_mse: 0.0153 - c12_mae:
0.1046 - bb_accuracy: 0.3452 - bb_mse: 0.0259 - bb_mae: 0.1369 - val_loss:
```

```
0.0885 - val_pc_loss: 0.0158 - val_c12_loss: 0.0438 - val_bb_loss: 0.0289 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0158 - val_pc_mae: 0.0657 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0438 - val_c12_mae: 0.1440 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0289 - val_bb_mae: 0.1379
Epoch 95/100
4/4 [==============================] - 3s 717ms/step - loss: 0.0439 - pc_loss:
0.0029 - c12_loss: 0.0151 - bb_loss: 0.0258 - pc_accuracy: 1.0000 - pc_mse:
0.0029 - pc_mae: 0.0495 - c12_accuracy: 0.9898 - c12_mse: 0.0151 - c12_mae:
0.1040 - bb_accuracy: 0.3401 - bb_mse: 0.0258 - bb_mae: 0.1366 - val_loss:
0.1086 - val_pc_loss: 0.0180 - val_c12_loss: 0.0605 - val_bb_loss: 0.0300 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0180 - val_pc_mae: 0.0675 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0605 - val_c12_mae: 0.1645 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0300 - val_bb_mae: 0.1395
Epoch 96/100
4/4 [==============================] - 3s 724ms/step - loss: 0.0441 - pc_loss:
0.0029 - c12_loss: 0.0155 - bb_loss: 0.0257 - pc_accuracy: 1.0000 - pc_mse:
0.0029 - pc_mae: 0.0497 - c12_accuracy: 0.9898 - c12_mse: 0.0155 - c12_mae:
0.1051 - bb_accuracy: 0.3452 - bb_mse: 0.0257 - bb_mae: 0.1361 - val_loss:
0.0956 - val_pc_loss: 0.0180 - val_c12_loss: 0.0483 - val_bb_loss: 0.0293 -
val_pc_accuracy: 0.9545 - val_pc_mse: 0.0180 - val_pc_mae: 0.0674 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0483 - val_c12_mae: 0.1496 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0293 - val_bb_mae: 0.1380
Epoch 97/100
4/4 [==============================] - 3s 720ms/step - loss: 0.0434 - pc_loss:
0.0029 - c12_loss: 0.0148 - bb_loss: 0.0257 - pc_accuracy: 1.0000 - pc_mse:
0.0029 - pc_mae: 0.0494 - c12_accuracy: 0.9898 - c12_mse: 0.0148 - c12_mae:
0.1027 - bb_accuracy: 0.3401 - bb_mse: 0.0257 - bb_mae: 0.1361 - val_loss:
0.0793 - val_pc_loss: 0.0107 - val_c12_loss: 0.0414 - val_bb_loss: 0.0273 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0107 - val_pc_mae: 0.0614 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0414 - val_c12_mae: 0.1421 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0273 - val_bb_mae: 0.1329
Epoch 98/100
4/4 [==============================] - 3s 718ms/step - loss: 0.0433 - pc_loss:
0.0029 - c12_loss: 0.0148 - bb_loss: 0.0256 - pc_accuracy: 1.0000 - pc_mse:
0.0029 - pc_mae: 0.0492 - c12_accuracy: 0.9898 - c12_mse: 0.0148 - c12_mae:
0.1028 - bb_accuracy: 0.3401 - bb_mse: 0.0256 - bb_mae: 0.1358 - val_loss:
0.0821 - val_pc_loss: 0.0125 - val_c12_loss: 0.0418 - val_bb_loss: 0.0278 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0125 - val_pc_mae: 0.0626 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0418 - val_c12_mae: 0.1418 -
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0278 - val_bb_mae: 0.1349
Epoch 99/100
4/4 [==============================] - 3s 720ms/step - loss: 0.0429 - pc_loss:
0.0029 - c12_loss: 0.0145 - bb_loss: 0.0255 - pc_accuracy: 1.0000 - pc_mse:
0.0029 - pc_mae: 0.0491 - c12_accuracy: 0.9898 - c12_mse: 0.0145 - c12_mae:
0.1017 - bb_accuracy: 0.3452 - bb_mse: 0.0255 - bb_mae: 0.1356 - val_loss:
0.0816 - val_pc_loss: 0.0132 - val_c12_loss: 0.0410 - val_bb_loss: 0.0273 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0132 - val_pc_mae: 0.0636 -
val_c12_accuracy: 1.0000 - val_c12_mse: 0.0410 - val_c12_mae: 0.1413 -
```

```
val_bb_accuracy: 0.3636 - val_bb_mse: 0.0273 - val_bb_mae: 0.1330
Epoch 100/100
4/4 [==============================] - 3s 715ms/step - loss: 0.0427 - pc_loss:
0.0029 - c12_loss: 0.0144 - bb_loss: 0.0254 - pc_accuracy: 1.0000 - pc_mse:
0.0029 - pc_mae: 0.0492 - c12_accuracy: 0.9898 - c12_mse: 0.0144 - c12_mae:
0.1015 - bb_accuracy: 0.3401 - bb_mse: 0.0254 - bb_mae: 0.1351 - val_loss:
0.0843 - val_pc_loss: 0.0060 - val_c12_loss: 0.0514 - val_bb_loss: 0.0269 -
val_pc_accuracy: 1.0000 - val_pc_mse: 0.0060 - val_pc_mae: 0.0558 -
val_c12_accuracy: 0.9545 - val_c12_mse: 0.0514 - val_c12_mae: 0.1550 -
val_bb_accuracy: 0.3182 - val_bb_mse: 0.0269 - val_bb_mae: 0.1360
```

[ ]:
```
# Plotting the performance of the model

plot_performance(history)
```



[ ]:
```
# Evaluating the model

test_metrics = model.evaluate(test_x,y={'pc':test_y_pc,'c12':test_y_c12,'bb':
 ↪test_y_bb},verbose=0)
test_metrics = np.round(np.multiply(test_metrics,100), 4)
print("% pc Accuracy on test set: ",test_metrics[4])
print("% c12 Accuracy on test set: ", test_metrics[7])
print("% bb Mean Suared Error on test set:  ", test_metrics[11])
```

```
% pc Accuracy on test set:  100.0
% c12 Accuracy on test set:  88.0
% bb Mean Suared Error on test set:   3.0454
```

1.10 Future Scope:

Linear expansion of classes of screws is easily possible as we only need to add additional outputs corresponding to new classes. As the model gets more and more data, its performance will become better and better.

1.11 Conclusion:

Using Deep Neural Network, We detected screws from images with 100 % accuracy on test set and classified screws with 88 % accuracy on test set. Also, we got mean squared error of 3 % on bounding box regression.

1.12 References:

1. https://www.tensorflow.org/guide/keras
2. Inkscape software
3. https://www.tensorflow.org/guide/keras/functional
4. https://keras.io/api/layers/activations/
5. https://colab.research.google.com/

[ ]:

```

```