Module 8 Assignment 1: Dogs vs. Cats Redux (Kernel Edition)

Claire Markey, Julia Granito, Manny Hurtado, and Steve Desilets

MSDS 422: Practical Machine Learning

May 21st, 2023

Introduction

Classification of images such as dogs or cats may be accomplished through the use of Convolutional Neural Networks (CNN). CNNs may be utilized to build classifiers that can assign the correct label to an image (in this case a dog or cat). To that end, we sought to explore how CNNs may be best used or modified to correctly classify the provided images.

Method

Kaggle data containing images of dogs and cats were downloaded and analyzed using Jupyter Notebooks (Cukierski, 2016). The images were cropped and image pixels were rescaled to prepare for input into the constructed CNN models. These models were trained and used to predict whether an image contains a dog or a cat (1 = dog, 0 = cat). Model performance metrics were examined and compared.

Results and Insights

First, an exploratory data analysis (EDA) revealed that the train data contains 25,000 images of dogs and cats. Each image has the label and a numeric identifier as part of the filename. 12,500 images were labeled “cat” and 12,500 images were labeled “dog”. The test data contains 12,500 images, named according to a numeric identifier. The images were resized to 150 x 150 pixels with 3 channels (RGB). For all CNN models, image pixel values were rescaled to range 0 to 1 and training and validation batches were prepared and augmented using Keras’ ImageDataGenerator (Geron, 2019). Augmenting the training images through rotation incorporates a level of variation in the dataset which can improve the generalization and robustness of the trained model. However, this will increase runtime for a large volume of data.

A sequential convolutional neural network model was then constructed with multiple 2D convolutional, max pooling, dense, and dropout layers. The ReLU activation function was used on the 2D convolutional layers to introduce non-linearity to the model with the exception of the final layer, where the sigmoid activation function was used for outputting binary classification probabilities for the images. Max pooling layers followed each of the convolutional layers with a pool size of 2, effectively dividing the spatial dimension of each image. This pattern was repeated three more times before introducing dense and dropout layers into the model. Filters were set to 64, 32, and 16 with kernel sizes of 5 x 5 for the first layer and 3 x 3 for subsequent layers to increase granularity. In this design, we chose to reduce the filter count for each layer and divide the filter size for subsequent layers by 2. However, we could explore a design with increasing detail for each sequential layer, such as doubling the filters after each pooling layer. Padding was not used, so the model loses pixels on the sides of the feature maps. The Adam optimizer with a learning rate of .001 was used to update the model on batches of size 32 for 20 epochs. Overall training time was about 90 minutes. The best model achieved an accuracy of 0.771 with a validation accuracy of 0.768. This model achieved a log loss of 0.436 in Kaggle. To reduce the computation time, we could explore using a higher learning rate with larger batch sizes, and reduce or eliminate our data augmentation. In addition, we could include bottleneck layers and explore reducing the dimensions of our input images using principal component or similar analysis. To further increase accuracy, we could use other methods to create same-sized images, as resizing differently sized images to 150x150 pixels distorts the underlying image.

To continue the analysis, a second CNN model was developed using a modified architecture. This model contains one 2D convolutional layer and one max pooling layer, then followed by two sets consisting of two convolutional layers and one max pooling layer. The number of filters used in each of these three sets increased, with filters being set to 64, 138, and then 256. This approach allowed the model to first detect smaller features and determine larger features from the pixel data. To prevent overfitting, two sets of layers consisting of one dense layer (ReLu activation function; units set to 128 and then 64; kernel sizes set to 7x7 and then kept at 3x3) and one dropout layer (dropout set to 50%) were implemented. A final sigmoid activation function layer was included at the end of architecture. This model architecture differed by using an RMSprop optimizer function (learning rate set to .01). The best model achieved lower accuracy (.50), validation accuracy (.49), and higher Kaggle log loss compared to the first model and its architecture. The adaptations of our models do not follow a structured experimental design, so the models cannot be directly compared to assess the impacts of specific architecture modifications. However, broadly it appears that the ordering and number of filters (and potentially filter size) meaningfully impact classification accuracy for these data.

A third CNN model was then created to determine whether slight changes to the first CNN model could result in improvements in predictive accuracy. This new model was also constructed with multiple 2D convolutional, max pooling, dense, and dropout layers. However, this model introduced an additional 2D convolutional and max pooling layer compared to the original model. The four 2D convolutional layers utilized 64, 32, 16, and 8 filters with 3x3 kernels and ReLU activation functions. The new max pooling layer also used a pool size of 2. All other parameters, including number of epochs, learning rate, use of no padding layers, structure of dense and dropout layers, batch sizes, and the output layer (sigmoid) activation function all remained unchanged compared to the original CNN model. The best model (over 6 hours of training time) achieved a training dataset prediction accuracy of 0.7920. We then applied the best CNN model to the validation and testing datasets. The validation dataset predictions were leveraged to construct Receiving Operating Characteristic (ROC) and Precision-Recall Curves, which did not look great. However, application of this model to the testing dataset resulted in a log loss of only 0.379, which suggests that the model performed moderately well.

The findings of these experiments suggest that CNNs may serve as useful tools for image classification problems and that neural network architectural and hyperparameter choices play important roles on prediction accuracy. CNN models tend to perform well at leveraging spatial correlations between data points in image data in order to identify useful features for classification. Accordingly, our models performed decently well at predicting cats and dogs based on their underlying pixel data. Notably, though, the prediction accuracy obtained from these three CNNs ranged widely, which confirms that researchers' choices for hyperparameters and CNN model architecture have very significant impacts on predictive performance of the model. Further research could attempt to improve the prediction accuracy to be higher than our maximum achieved testing dataset prediction accuracy of 0.79 by further refining the CNN model. This research may better equip data scientists interested in understanding how to maximize the predictive power of CNNs.

References

Geron, Aurelien. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. 2nd ed. Sebastopol, CA: O'Reilly

Will Cukierski. 2016. “Dogs vs. Cats Redux: Kernels Edition.” *Kaggle.* https://kaggle.com/competitions/dogs-vs-cats-redux-kernels-edition

.

Appendix 1 - Python Code and Outputs

Data Preparation

In [1]:

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

NOTE: extract images from zip file into train folder in the base of the current working directory of this notebook, this can be done manually or via zipfile. Example:

```
import zipfile
with zipfile.ZipFile('train.zip','r') as z:
    z.extractall("train")
    print('The train dataset is extracted into the train folder of
the current working directory')
```

Import Extracted Training Data

In [2]:

```
import os
# import extracted training files, in this case files were extracted to the train folc
Train_Path = "train"
train_files = os.listdir(Train_Path)
```

In [3]:

```
# number of training images
len(train_files)

# first ten image file names
train_files[0:10]
```

Out[3]:

```
25000
```

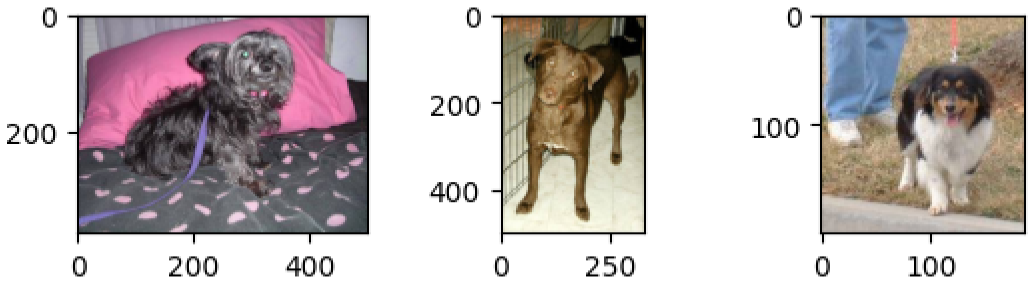
Out[3]:

```
['cat.0.jpg',
 'cat.1.jpg',
 'cat.10.jpg',
 'cat.100.jpg',
 'cat.1000.jpg',
 'cat.10000.jpg',
 'cat.10001.jpg',
 'cat.10002.jpg',
 'cat.10003.jpg',
 'cat.10004.jpg']
```

In [4]:

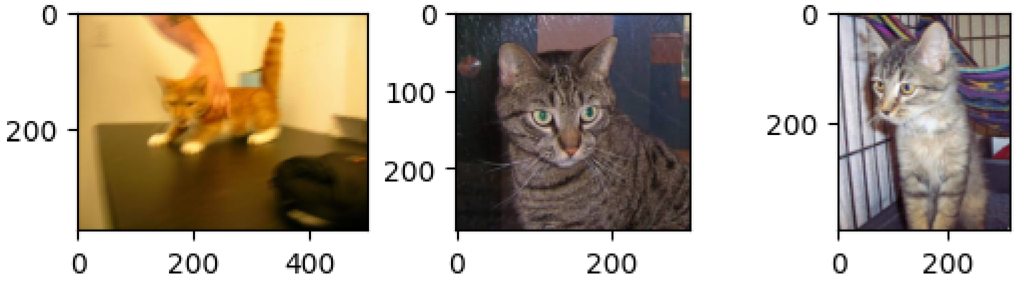
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread

# define location of dataset
folder = 'train/'

# plot first few images
for i in range(3):
    plt.subplot(330 + 1 + i) # define subplot
    filename = folder + 'dog.' + str(i) + '.jpg' # define filename
    image = imread(filename) # load image pixels
    plt.imshow(image) # plot raw pixel data

plt.show() # show the figure
```

```
Out[4]: <Axes: >

Out[4]: <matplotlib.image.AxesImage at 0x7f9234504ca0>

Out[4]: <Axes: >

Out[4]: <matplotlib.image.AxesImage at 0x7f92345593a0>

Out[4]: <Axes: >

Out[4]: <matplotlib.image.AxesImage at 0x7f9234489400>
```



```
In [5]: # plot first few images
        for i in range(3):
            plt.subplot(330 + 1 + i) # define subplot
            filename = folder + 'cat.' + str(i) + '.jpg' # define filename
            image = imread(filename) # load image pixels
            plt.imshow(image) # plot raw pixel data

        plt.show() # show the figure
```

```
Out[5]: <Axes: >

Out[5]: <matplotlib.image.AxesImage at 0x7f922abaf490>

Out[5]: <Axes: >

Out[5]: <matplotlib.image.AxesImage at 0x7f922ab78f40>

Out[5]: <Axes: >

Out[5]: <matplotlib.image.AxesImage at 0x7f922ab89fa0>
```



Extract Labels from File Names

In [6]:

```
# extract label from file name
label = []
identifier = []
for file in train_files:
    file_name = file.split(".")
    label.append(file_name[0])
    identifier.append(file_name[1])

# create df with id and label
train_df = pd.DataFrame(data={'id':identifier,'label':label})

# dummy encode label column
train_df["label_num"] = np.where(train_df["label"] == 'cat', 1, 0)

# number of labels should be 25000
train_df.shape

# first ten rows
train_df.head(10)
```

Out[6]:

```
(25000, 3)
```

Out[6]:

	id	label	label_num
0	0	cat	1
1	1	cat	1
2	10	cat	1
3	100	cat	1
4	1000	cat	1
5	10000	cat	1
6	10001	cat	1
7	10002	cat	1
8	10003	cat	1
9	10004	cat	1

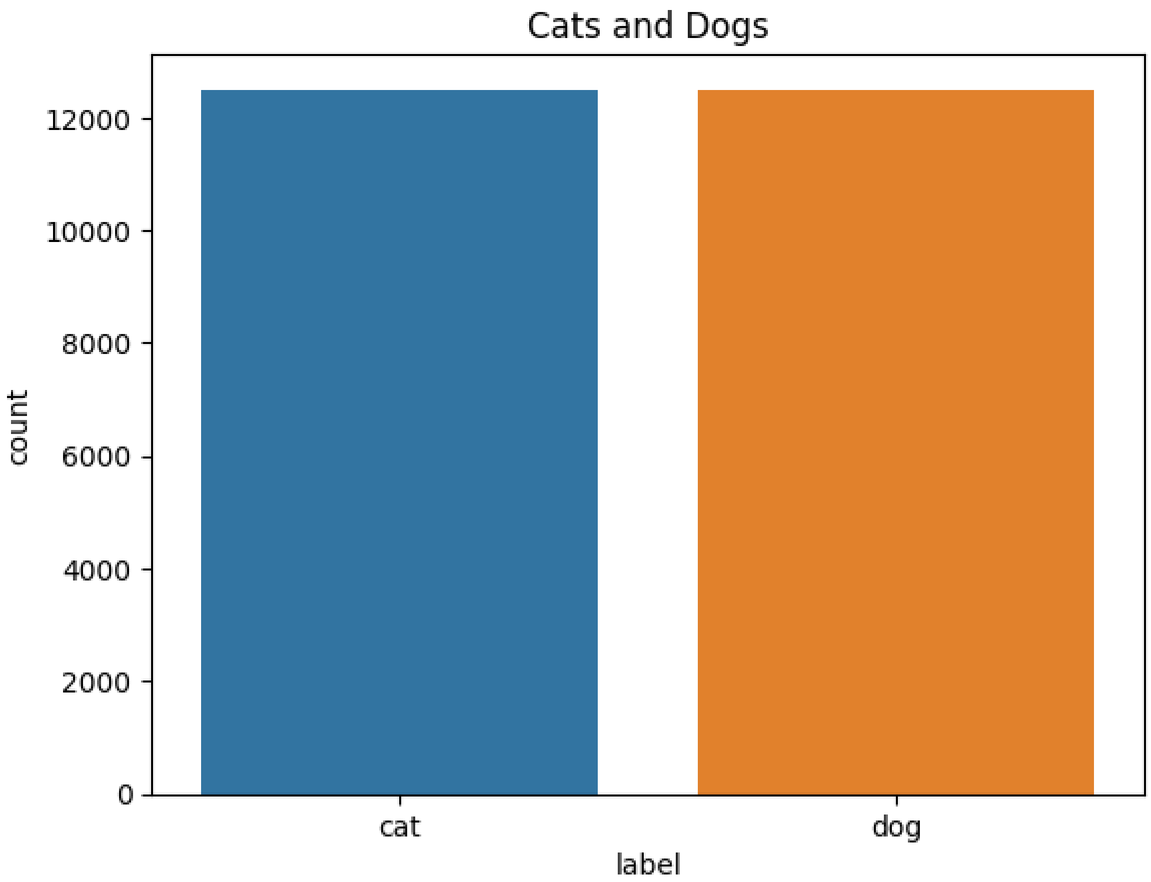
Plot label counts

In [7]:

```
import seaborn as sns
import plotly as plt
sns.countplot(x=train_df.label).set(title = 'Cats and Dogs')
```

Out[7]:

```
[Text(0.5, 1.0, 'Cats and Dogs')]
```



Extract Image Dimensions from First 200 Training Images

In [8]:
```
import pandas as pd
import cv2

# loop through training files to get image dimensions
img=[]
for file in train_files[0:200]:
    count=+1
    img.append(cv2.imread(os.path.join(Train_Path,file)).shape)

# create df with dim
dim_df = pd.DataFrame(data={'dimension':img})
dim_df.head(10)
```

Out[8]:

	dimension
0	(374, 500, 3)
1	(280, 300, 3)
2	(499, 489, 3)
3	(499, 403, 3)
4	(149, 150, 3)
5	(359, 431, 3)
6	(374, 500, 3)
7	(471, 499, 3)
8	(375, 499, 3)
9	(239, 320, 3)

Import Extracted Testing Data

In [9]:

```
# import extracted testing files, in this case files were extracted to the test folder
Test_Path = "test"
test_files = os.listdir(Test_Path)
```

In [10]:

```
# number of test images
len(test_files)

# first ten image file names
test_files[0:10]
```

Out[10]:

```
12500
```

Out[10]:

```
['1.jpg',
 '10.jpg',
 '100.jpg',
 '1000.jpg',
 '10000.jpg',
 '10001.jpg',
 '10002.jpg',
 '10003.jpg',
 '10004.jpg',
 '10005.jpg']
```

Crop and Resize Training Images to 150x150

In [11]:

```
import os
from PIL import Image

# make new directory for cropped pictures
Train_Cropped_Path = "train_cropped/"
#os.mkdir(Train_Cropped_Path)

# crop images and save in train_cropped folder
for file in train_files:
    im = Image.open(os.path.join(Train_Path,file))
    im = im.resize((150, 150))
    im = im.save(f"{Train_Cropped_Path}crop{file}")
```

Move cropped images to 'cat' or 'dog' folder based on label for the image data generator

In [12]:

```
import shutil, sys

# image labels
categories = ['cat' , 'dog']

# function to move cat images and dog images to folders
def move_images_to_specific_folder(new_path, category):
    for image_name in os.listdir(new_path):
        if category in image_name:
            if image_name.endswith('.jpg'):
                shutil.move(os.path.join(new_path,image_name), os.path.join(new_path,

# create folders for cats and dogs
for category in categories:
    path = os.path.join(Train_Cropped_Path, category)
    os.mkdir(path)

# move cropped files to appropriate folder based on label
for category in categories:
    move_images_to_specific_folder(Train_Cropped_Path, category)
```

Check Dimensions of Cropped Training Images

In [13]:

```
# loop through categories
img=[]
for category in categories:
    path = os.path.join(Train_Cropped_Path, category)
    train_cropped_files = os.listdir(path)
    for file in train_cropped_files[0:10]:
        img.append(cv2.imread(os.path.join(path,file)).shape)

# create df with dimensions, all images should be 150x150
dim_cropped_df = pd.DataFrame(data={'dimension':img})
dim_cropped_df.head(10)
```

Out[13]:

	dimension
0	(150, 150, 3)
1	(150, 150, 3)
2	(150, 150, 3)
3	(150, 150, 3)
4	(150, 150, 3)
5	(150, 150, 3)
6	(150, 150, 3)
7	(150, 150, 3)
8	(150, 150, 3)
9	(150, 150, 3)

The images are resized to 150x150

In [15]:

```
# load all libraries needed
import shutil, sys
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Input, Dense, Dropout, Flatten, MaxPool2D
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
%matplotlib inline
```

Use ImageDataGenerator to Prepare Training and Validation Batches

In [16]:

```
# create ImageDataGenerator to apply preprocessing for images and split data to batche
image_size = 150 # dimension of cropped image is 150x150
batch_size = 32 # start with relatively small batch size
epochs = 20 # start with 20 epochs

train_datagen = ImageDataGenerator(rescale = 1./255, # rescale the image pixels
                                   rotation_range=20,
                                   validation_split=0.2, # allocate 20% of the data as
                                   horizontal_flip=True,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2
                                  )

train_generator = train_datagen.flow_from_directory('train_cropped',
                                                    class_mode='binary',
                                                    batch_size = batch_size,
                                                    target_size=(image_size,image_size
                                                    subset='training',
                                                    shuffle=True,
                                                    seed=10)
validation_generator = train_datagen.flow_from_directory('train_cropped',
                                                         class_mode='binary',
                                                         batch_size = batch_size,
                                                         target_size=(image_size,image
                                                         subset='validation',
                                                         shuffle=True,
                                                         seed=10)
```

```
Found 20000 images belonging to 2 classes.
Found 5000 images belonging to 2 classes.
```

STILL WORKING ON ADJUSTING MODEL PARAMETERS - WORK IN PROGRESS

In [18]:

```
# Build Initial Sequential Model
model = Sequential()
# 2D convolutional layer w/64 filters, 5x5 kernel, and ReLU activation function. Model
model.add(Conv2D(input_shape=(150,150,3), filters = 64, kernel_size=(5,5), activation=
# max pooling layer
model.add(MaxPool2D(pool_size=(2,2)))

# Conv2D and MaxPooling2D layers with 32 filters and 3x3 kernel.
model.add(Conv2D(filters = 32, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))

# Conv2D and MaxPooling2D layers with 16 filters 3x3 kernel.
model.add(Conv2D(filters = 16, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))

# add dropout to avoid overfitting
model.add(Dropout(0.25))

# flatten output of the previous layer - converts to 1d layer
model.add(Flatten())

# add dense layer
model.add(Dense(32, activation='relu'))

# add dropout to avoid overfitting
model.add(Dropout(0.5))

# add dense layer
model.add(Dense(units=1, activation="sigmoid"))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_3 (Conv2D)           (None, 146, 146, 64)      4864      
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 73, 73, 64)       0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 71, 71, 32)        18464     
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 35, 35, 32)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 33, 33, 16)        4624      
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 16, 16, 16)       0         
 2D)                                                             
                                                                 
 dropout_2 (Dropout)         (None, 16, 16, 16)        0         
                                                                 
 flatten_1 (Flatten)         (None, 4096)              0         
                                                                 
 dense_2 (Dense)             (None, 32)                131104    
                                                                 
 dropout_3 (Dropout)         (None, 32)                0         
                                                                 
 dense_3 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 159,089
Trainable params: 159,089
Non-trainable params: 0
_________________________________________________________________
```

In [15]:

```python
# Adam solver optimizer with learning rate of 0.001
optimizer=tf.keras.optimizers.Adam(learning_rate=0.001)

# compile model
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

In [16]:

```python
import warnings
warnings.filterwarnings('ignore')
# early stopping based on validation loss (stops if model doesn't improve after 5 iter
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)

# save the best model as 'best_model.cnn' based on validation loss
save_best = ModelCheckpoint(filepath = 'best_model.cnn', verbose=1, save_best_only=Tru

# fit model using batches of training data and batches of testing data
history = model.fit(train_generator, steps_per_epoch=train_generator.samples // batch_
                              validation_data = validation_generator,
                              validation_steps = validation_generator.samples // batch
                              epochs = epochs,
                              callbacks=[save_best, early_stopping],
                              verbose=2)
```

```
Epoch 1/20
```

```
2023-05-19 16:15:47.299673: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'Placeholder/_0' with dtype int32
         [[{{node Placeholder/_0}}]]
2023-05-19 16:15:47.875890: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:9
54] layout failed: INVALID_ARGUMENT: Size of values 0 does not match size of permutat
ion 4 @ fanin shape insequential/dropout/dropout/SelectV2-2-TransposeNHWCToNCHW-Layou
tOptimizer
2023-05-19 16:15:53.695685: I tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.c
c:424] Loaded cuDNN version 8901
2023-05-19 16:16:02.147826: I tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.
cc:637] TensorFloat-32 will be used for the matrix multiplication. This will only be
logged once.
2023-05-19 16:16:02.213183: I tensorflow/compiler/xla/service/service.cc:169] XLA ser
vice 0x33f7e8d0 initialized for platform CUDA (this does not guarantee that XLA will
be used). Devices:
2023-05-19 16:16:02.214824: I tensorflow/compiler/xla/service/service.cc:177]   Strea
mExecutor device (0): NVIDIA GeForce RTX 3050 Ti Laptop GPU, Compute Capability 8.6
2023-05-19 16:16:02.376796: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_uti
l.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTO
RY` to enable.
2023-05-19 16:16:03.217350: I ./tensorflow/compiler/jit/device_compiler.h:180] Compil
ed cluster using XLA!  This line is logged at most once for the lifetime of the proce
ss.
2023-05-19 16:19:50.775011: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'Placeholder/_0' with dtype int32
         [[{{node Placeholder/_0}}]]
```

```
Epoch 1: val_loss improved from inf to 0.68270, saving model to best_model.cnn
```

```
2023-05-19 16:20:46.868148: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:20:46.883213: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 16:20:47.116625: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:20:47.147750: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
625/625 - 301s - loss: 0.6890 - accuracy: 0.5386 - val_loss: 0.6827 - val_accuracy:
0.5721 - 301s/epoch - 481ms/step
Epoch 2/20

Epoch 2: val_loss improved from 0.68270 to 0.65307, saving model to best_model.cnn
```

```
2023-05-19 16:25:14.614783: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:25:14.630243: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 16:25:15.519815: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:25:15.556845: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
625/625 - 268s - loss: 0.6728 - accuracy: 0.5864 - val_loss: 0.6531 - val_accuracy:
0.6178 - 268s/epoch - 429ms/step
Epoch 3/20

Epoch 3: val_loss improved from 0.65307 to 0.62187, saving model to best_model.cnn
```

```
2023-05-19 16:29:37.320760: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:29:37.336567: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 16:29:37.492521: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:29:37.516246: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
625/625 - 262s - loss: 0.6527 - accuracy: 0.6202 - val_loss: 0.6219 - val_accuracy:
0.6518 - 262s/epoch - 419ms/step
Epoch 4/20

Epoch 4: val_loss improved from 0.62187 to 0.59946, saving model to best_model.cnn
2023-05-19 16:34:03.087456: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:34:03.098684: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 16:34:03.252837: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:34:03.280496: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn/assets
INFO:tensorflow:Assets written to: best_model.cnn/assets
625/625 - 266s - loss: 0.6289 - accuracy: 0.6510 - val_loss: 0.5995 - val_accuracy:
0.6865 - 266s/epoch - 425ms/step
Epoch 5/20

Epoch 5: val_loss improved from 0.59946 to 0.57827, saving model to best_model.cnn
2023-05-19 16:38:18.224869: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:38:18.238906: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 16:38:18.399162: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:38:18.426660: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
INFO:tensorflow:Assets written to: best_model.cnn/assets
625/625 - 255s - loss: 0.6102 - accuracy: 0.6727 - val_loss: 0.5783 - val_accuracy: 0.6991 - 255s/epoch - 408ms/step
Epoch 6/20

Epoch 6: val_loss did not improve from 0.57827
625/625 - 259s - loss: 0.5978 - accuracy: 0.6837 - val_loss: 0.5888 - val_accuracy: 0.6845 - 259s/epoch - 415ms/step
Epoch 7/20

Epoch 7: val_loss improved from 0.57827 to 0.55521, saving model to best_model.cnn
2023-05-19 16:47:03.679394: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:47:03.697301: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 16:47:04.141093: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:47:04.169852: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn/assets
INFO:tensorflow:Assets written to: best_model.cnn/assets
625/625 - 266s - loss: 0.5871 - accuracy: 0.6973 - val_loss: 0.5552 - val_accuracy: 0.7198 - 266s/epoch - 426ms/step
Epoch 8/20

Epoch 8: val_loss improved from 0.55521 to 0.53402, saving model to best_model.cnn
```

```
2023-05-19 16:51:40.930446: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:51:40.944853: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 16:51:41.092140: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:51:41.116358: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
625/625 - 277s - loss: 0.5693 - accuracy: 0.7096 - val_loss: 0.5340 - val_accuracy:
0.7398 - 277s/epoch - 443ms/step
Epoch 9/20

Epoch 9: val_loss improved from 0.53402 to 0.51636, saving model to best_model.cnn
```

```
2023-05-19 16:56:12.736299: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:56:12.749123: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 16:56:12.901291: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 16:56:12.929765: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
625/625 - 272s - loss: 0.5559 - accuracy: 0.7260 - val_loss: 0.5164 - val_accuracy:
0.7516 - 272s/epoch - 435ms/step
Epoch 10/20

Epoch 10: val_loss did not improve from 0.51636
625/625 - 259s - loss: 0.5477 - accuracy: 0.7303 - val_loss: 0.5186 - val_accuracy:
0.7526 - 259s/epoch - 415ms/step
Epoch 11/20

Epoch 11: val_loss improved from 0.51636 to 0.50657, saving model to best_model.cnn
```

```
2023-05-19 17:04:58.936418: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:04:58.949972: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 17:04:59.095736: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:04:59.122846: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
625/625 - 267s - loss: 0.5379 - accuracy: 0.7388 - val_loss: 0.5066 - val_accuracy:
0.7562 - 267s/epoch - 427ms/step
Epoch 12/20

Epoch 12: val_loss improved from 0.50657 to 0.50519, saving model to best_model.cnn
```

```
2023-05-19 17:09:25.772791: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:09:25.788787: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 17:09:26.198490: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:09:26.227559: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
625/625 - 267s - loss: 0.5334 - accuracy: 0.7412 - val_loss: 0.5052 - val_accuracy:
0.7656 - 267s/epoch - 427ms/step
Epoch 13/20

Epoch 13: val_loss improved from 0.50519 to 0.49208, saving model to best_model.cnn
```

```
2023-05-19 17:13:54.312360: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:13:54.329135: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 17:13:54.479357: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:13:54.502887: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
625/625 - 268s - loss: 0.5290 - accuracy: 0.7436 - val_loss: 0.4921 - val_accuracy:
0.7714 - 268s/epoch - 429ms/step
Epoch 14/20

Epoch 14: val_loss improved from 0.49208 to 0.48775, saving model to best_model.cnn
```

```
2023-05-19 17:18:18.912751: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:18:18.926411: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 17:18:19.076868: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:18:19.102629: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
625/625 - 265s - loss: 0.5205 - accuracy: 0.7480 - val_loss: 0.4877 - val_accuracy:
0.7754 - 265s/epoch - 423ms/step
Epoch 15/20

Epoch 15: val_loss improved from 0.48775 to 0.47903, saving model to best_model.cnn
```

```
2023-05-19 17:22:38.752108: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:22:38.763135: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 17:22:38.923388: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:22:38.949746: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
```

```
INFO:tensorflow:Assets written to: best_model.cnn/assets
625/625 - 260s - loss: 0.5115 - accuracy: 0.7560 - val_loss: 0.4790 - val_accuracy:
0.7752 - 260s/epoch - 416ms/step
Epoch 16/20

Epoch 16: val_loss did not improve from 0.47903
625/625 - 264s - loss: 0.5049 - accuracy: 0.7606 - val_loss: 0.4937 - val_accuracy:
0.7636 - 264s/epoch - 423ms/step
Epoch 17/20

Epoch 17: val_loss did not improve from 0.47903
625/625 - 261s - loss: 0.5063 - accuracy: 0.7581 - val_loss: 0.4855 - val_accuracy:
0.7841 - 261s/epoch - 417ms/step
Epoch 18/20

Epoch 18: val_loss did not improve from 0.47903
625/625 - 270s - loss: 0.4957 - accuracy: 0.7634 - val_loss: 0.4937 - val_accuracy:
0.7652 - 270s/epoch - 432ms/step
Epoch 19/20

Epoch 19: val_loss improved from 0.47903 to 0.46612, saving model to best_model.cnn
2023-05-19 17:40:34.776138: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:40:34.790559: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 17:40:34.939172: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:40:34.968307: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn/assets
INFO:tensorflow:Assets written to: best_model.cnn/assets
625/625 - 281s - loss: 0.4909 - accuracy: 0.7691 - val_loss: 0.4661 - val_accuracy:
0.7889 - 281s/epoch - 449ms/step
Epoch 20/20

Epoch 20: val_loss improved from 0.46612 to 0.46608, saving model to best_model.cnn
```

```
2023-05-19 17:44:58.717570: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:44:58.734800: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
2023-05-19 17:44:58.883279: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,16,16,16]
         [[{{node inputs}}]]
2023-05-19 17:44:58.909669: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'inputs' with dtype float and shape [?,32]
         [[{{node inputs}}]]
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). Thes
e functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn/assets
INFO:tensorflow:Assets written to: best_model.cnn/assets
625/625 - 264s - loss: 0.4922 - accuracy: 0.7710 - val_loss: 0.4661 - val_accuracy:
0.7883 - 264s/epoch - 422ms/step
```

In [17]:

```
# show best model
best_model = tf.keras.models.load_model('best_model.cnn')
best_model.summary()
```

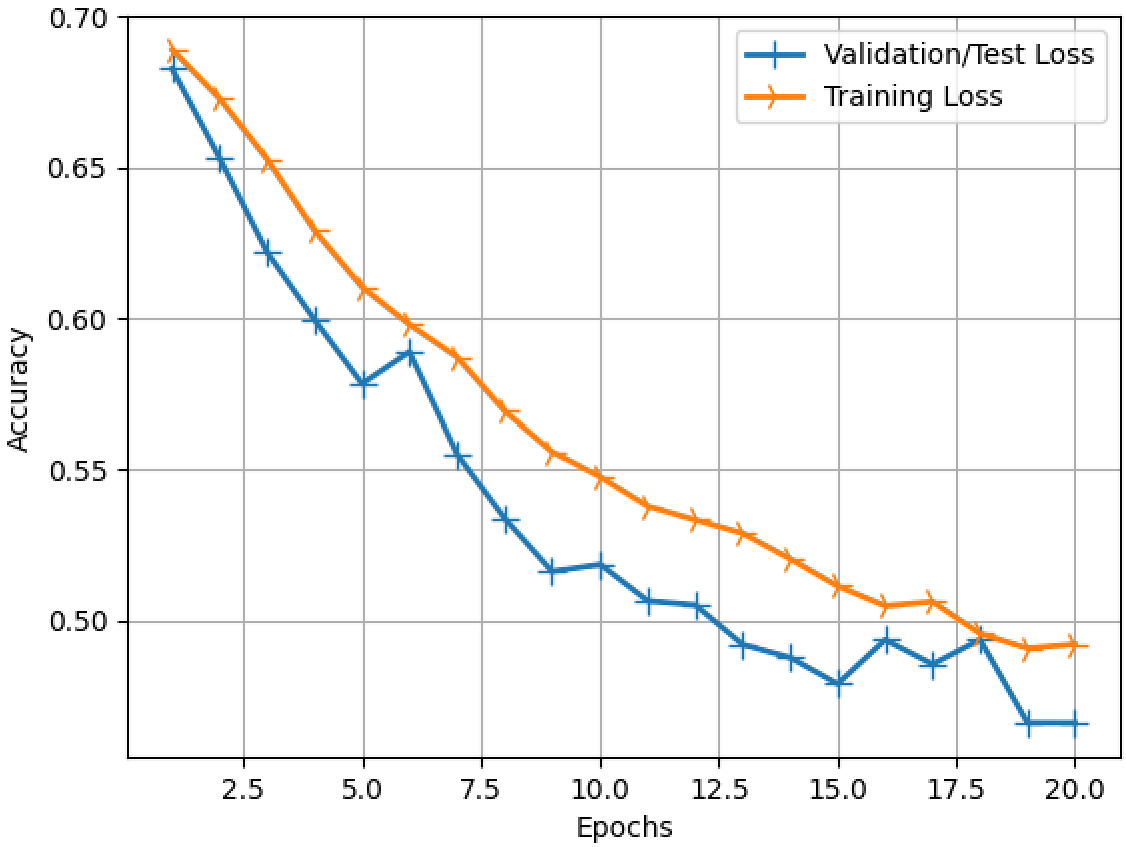
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 146, 146, 64)      4864      
                                                                 
 max_pooling2d (MaxPooling2D  (None, 73, 73, 64)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 71, 71, 32)        18464     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 35, 35, 32)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 33, 33, 16)        4624      
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 16, 16, 16)       0         
 2D)                                                             
                                                                 
 dropout (Dropout)           (None, 16, 16, 16)        0         
                                                                 
 flatten (Flatten)           (None, 4096)              0         
                                                                 
 dense (Dense)               (None, 32)                131104    
                                                                 
 dropout_1 (Dropout)         (None, 32)                0         
                                                                 
 dense_1 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 159,089
Trainable params: 159,089
Non-trainable params: 0
_________________________________________________________________
```

Loss Charts

In [18]:

```
history_dict = history.history

# extract the loss and validation losses
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

# save epochs
epochs = range(1, len(loss_values)+1)

# plot loss
line1 = plt.plot(epochs, val_loss_values, label ='Validation/Test Loss')
line2 = plt.plot(epochs, loss_values, label='Training Loss')

plt.setp(line1, linewidth = 2.0, marker='+', markersize=10.0)
plt.setp(line2, linewidth=2.0, marker='4', markersize=10.0)

# set labels
plt.xlabel('Epochs')
plt.ylabel('Accuracy')

plt.legend()
plt.grid(True)
plt.show()
```

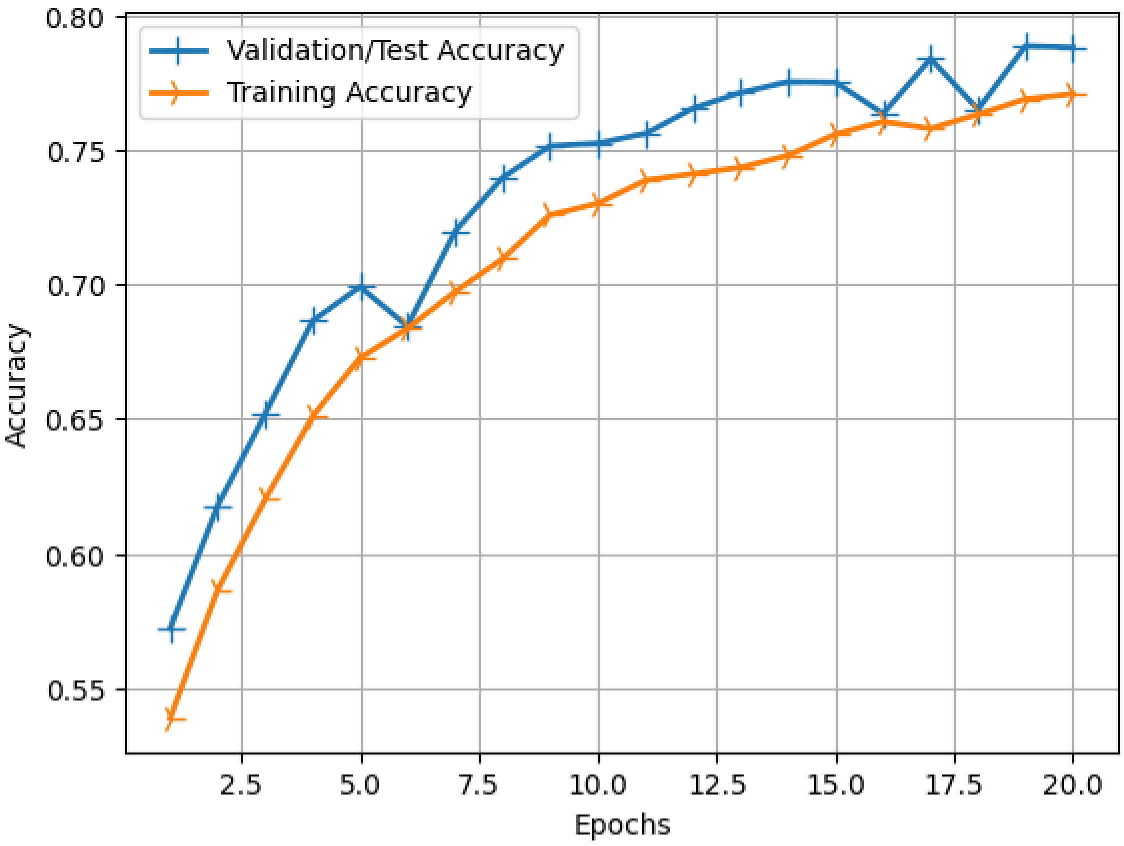
```
Out[18]: [None, None, None]

Out[18]: [None, None, None]

Out[18]: Text(0.5, 0, 'Epochs')

Out[18]: Text(0, 0.5, 'Accuracy')

Out[18]: <matplotlib.legend.Legend at 0x7f409c67bca0>
```



Accuracy Charts

```
In [19]: # extract acuuracy scores
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']

#plot accuracy scores
line1 = plt.plot(epochs, val_acc_values, label='Validation/Test Accuracy')
line2 = plt.plot(epochs, acc_values, label ='Training Accuracy')

plt.setp(line1, linewidth=2.0, marker='+', markersize=10.0)
plt.setp(line2, linewidth=2.0, marker='4', markersize=10.0)

# set labels
plt.xlabel('Epochs')
plt.ylabel('Accuracy')

plt.grid(True)
plt.legend()
plt.show()
```

```
Out[19]: [None, None, None]
Out[19]: [None, None, None]
Out[19]: Text(0.5, 0, 'Epochs')
Out[19]: Text(0, 0.5, 'Accuracy')
Out[19]: <matplotlib.legend.Legend at 0x7f40ac0becd0>
```



Crop and Resize Testing Images to 150x150

In [20]:

```
# make new directory for cropped images
Test_Cropped_Path = "test_cropped/"

os.mkdir(Test_Cropped_Path)

# crop images and save in train_cropped folder
for file in test_files:
    im = Image.open(os.path.join(Test_Path,file))
    im = im.resize((150, 150))
    im = im.save(f"{Test_Cropped_Path}{file}")

test_cropped_files = os.listdir(Test_Cropped_Path)
# create dataframe with filenames
test_df = pd.DataFrame(data = test_cropped_files, columns = ['filename'])
# extract id from filename
test_df['id'] = test_df['filename'].apply(lambda f: int(f.split('.')[0]))
test_df.sort_values(by = 'id', inplace = True, ignore_index = True)
```

Input Test Data into Testing Generator for Model Predictions

In [42]:

```
# rescale image pixels
test_gen = ImageDataGenerator(rescale = 1./255)
# create test generator for testing data
test_generator = test_gen.flow_from_dataframe(test_df,
                                               directory='test_cropped',
                                               x_col='filename',
                                               class_mode= None,
                                               target_size=(image_size,image_size),
                                               batch_size=batch_size,
                                               shuffle=False
)
```

```
Found 12500 validated image filenames.
```

Save Predictions into CSV file for Kaggle

In [23]:

```
# Apply the cnn model1 to the test dataset
cnn_pred1 = best_model.predict(test_generator, verbose = 1)

# Put the label predictions into a dataframe
cnn_pred1_df = pd.DataFrame(cnn_pred1, columns=['label'])

# Add the ID column to the front of the cnn predictions dataframe
cnn_pred1_df.insert(0, 'id', test_df['id'])

# Output predictions to csv
#cnn_pred1_df.to_csv('test_predictions_cnn_v3.csv', index=False)
```

```
2023-05-19 18:45:15.508063: I tensorflow/core/common_runtime/executor.cc:1197] [/devi
ce:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and y
ou can ignore this message): INVALID_ARGUMENT: You must feed a value for placeholder
tensor 'Placeholder/_0' with dtype int32
         [[{{node Placeholder/_0}}]]
391/391 [==============================] - 76s 194ms/step
```

Let's display the Kaggle results from the application of the CNN model on the test dataset

In [25]:

```
# Display the kaggle results (log loss) of CNN model
import matplotlib.pyplot as plt
plt.figure(figsize = (15, 15))
kaggle_results = plt.imread('Kaggle_results_cnn_v3.jpg')
plt.imshow(kaggle_results)
plt.axis("off")
plt.show()
```

Out[25]:

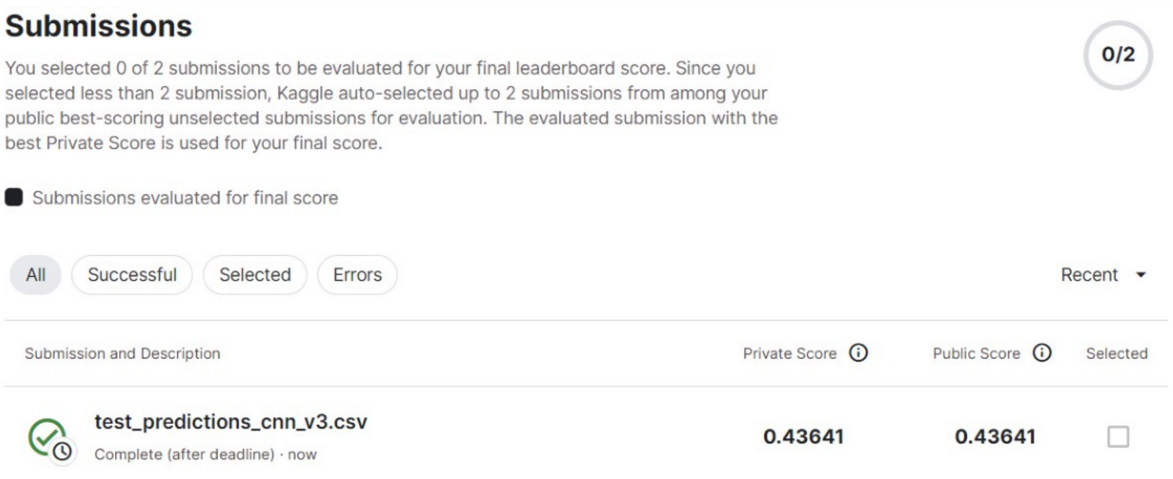
```
<Figure size 1500x1500 with 0 Axes>
```

Out[25]:

```
<matplotlib.image.AxesImage at 0x7f40f002e190>
```

Out[25]:

```
(-0.5, 1222.5, 481.5, -0.5)
```



Submissions

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

0/2

■ Submissions evaluated for final score

Submission and Description	Private Score ⓘ	Public Score ⓘ	Selected
test_predictions_cnn_v3.csv Complete (after deadline) · now	0.43641	0.43641	□

CNN Model 2

In [43]:

```
# Build Initial Sequential Model
model = Sequential()

# One 2D convolutional layer w/64 filters, 7x7 kernel, and ReLu activation function. M
model.add(Conv2D(input_shape=(150,150,3), filters = 64, kernel_size=(7,7), activation=
model.add(MaxPool2D(pool_size=(2,2))) #max pooling layer that divides spatial dimensio

# Two 2D convolutional layer w/128 filters, 3x3 kernel, and ReLu activation function.
model.add(Conv2D(input_shape=(150,150,3), filters = 128, kernel_size=(3,3), activation
model.add(Conv2D(input_shape=(150,150,3), filters = 128, kernel_size=(3,3), activation
model.add(MaxPool2D(pool_size=(2,2))) #max pooling layer that divides spatial dimensio

# Two 2D convolutional layer w/256 filters, 3x3 kernel, and ReLu activation function.
model.add(Conv2D(input_shape=(150,150,3), filters = 256, kernel_size=(3,3), activation
model.add(Conv2D(input_shape=(150,150,3), filters = 256, kernel_size=(3,3), activation
model.add(MaxPool2D(pool_size=(2,2))) #max pooling layer that divides spatial dimensio

# Flatten layers because the model expects a 1D array
model.add(Flatten())

# Add two sets of layers (one dense layer and one dropout layer) to reduce overfitting
model.add(Dense(units=128, activation="relu"))
model.add(Dropout(0.5))

model.add(Dense(units=64, activation="relu"))
model.add(Dropout(0.5))

# Add final softmax
model.add(Dense(units=1, activation ='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_5 (Conv2D)           (None, 144, 144, 64)      9472      
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 72, 72, 64)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 70, 70, 128)       73856     
                                                                 
 conv2d_7 (Conv2D)           (None, 68, 68, 128)       147584    
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 34, 34, 128)      0         
 2D)                                                             
                                                                 
 conv2d_8 (Conv2D)           (None, 32, 32, 256)       295168    
                                                                 
 conv2d_9 (Conv2D)           (None, 30, 30, 256)       590080    
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 15, 15, 256)      0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 57600)             0         
                                                                 
 dense_3 (Dense)             (None, 128)               7372928   
                                                                 
 dropout_2 (Dropout)         (None, 128)               0         
                                                                 
 dense_4 (Dense)             (None, 64)                8256      
                                                                 
 dropout_3 (Dropout)         (None, 64)                0         
                                                                 
 dense_5 (Dense)             (None, 1)                 65        
                                                                 
=================================================================
Total params: 8,497,409
Trainable params: 8,497,409
Non-trainable params: 0
_________________________________________________________________
```

Let's use an RMSprop optimizer set at a .01 learning rate and a binary cross-entropy loss function, and measure accuracy as a metric

In [44]:

```
# compile model
model.compile(loss='binary_crossentropy', optimizer= tf.keras.optimizers.RMSprop(learr

import warnings
warnings.filterwarnings('ignore')
# early stopping based on validation loss (stops if model doesn't improve after 10 ite
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
```

```
# save the best model as 'best_model.cnn' based on validation loss
save_best = ModelCheckpoint(filepath = 'best_model_2m.cnn', verbose=1, save_best_only=
```

Fit the model next:

In [45]:

```
# fit model using batches of training data and batches of testing data
history = model.fit(train_generator, steps_per_epoch=train_generator.samples // batch_
                              validation_data = validation_generator,
                              validation_steps = validation_generator.samples // batch
                              epochs = epochs,
                              callbacks=[save_best, early_stopping],
                              verbose=2)
```

```
Epoch 1/20

Epoch 1: val_loss improved from inf to 0.69357, saving model to best_model_2m.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit
_compiled_convolution_op while saving (showing 5 of 6). These functions will not be d
irectly callable after loading.
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
625/625 - 2472s - loss: 178.8774 - accuracy: 0.5000 - val_loss: 0.6936 - val_accurac
y: 0.5002 - 2472s/epoch - 4s/step
Epoch 2/20

Epoch 2: val_loss improved from 0.69357 to 0.69344, saving model to best_model_2m.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit
_compiled_convolution_op while saving (showing 5 of 6). These functions will not be d
irectly callable after loading.
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
625/625 - 2490s - loss: 0.6937 - accuracy: 0.5000 - val_loss: 0.6934 - val_accuracy:
0.5002 - 2490s/epoch - 4s/step
Epoch 3/20

Epoch 3: val_loss improved from 0.69344 to 0.69323, saving model to best_model_2m.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit
_compiled_convolution_op while saving (showing 5 of 6). These functions will not be d
irectly callable after loading.
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
```

```
625/625 - 2480s - loss: 0.6933 - accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy:
0.4996 - 2480s/epoch - 4s/step
Epoch 4/20

Epoch 4: val_loss did not improve from 0.69323
625/625 - 2509s - loss: 0.6933 - accuracy: 0.5000 - val_loss: 0.6933 - val_accuracy:
0.5000 - 2509s/epoch - 4s/step
Epoch 5/20

Epoch 5: val_loss did not improve from 0.69323
625/625 - 2481s - loss: 0.6933 - accuracy: 0.5000 - val_loss: 0.6934 - val_accuracy:
0.4998 - 2481s/epoch - 4s/step
Epoch 6/20

Epoch 6: val_loss improved from 0.69323 to 0.69320, saving model to best_model_2m.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit
_compiled_convolution_op while saving (showing 5 of 6). These functions will not be d
irectly callable after loading.
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
625/625 - 2464s - loss: 0.6934 - accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy:
0.5000 - 2464s/epoch - 4s/step
Epoch 7/20

Epoch 7: val_loss did not improve from 0.69320
625/625 - 2459s - loss: 0.6934 - accuracy: 0.5000 - val_loss: 0.6933 - val_accuracy:
0.5004 - 2459s/epoch - 4s/step
Epoch 8/20

Epoch 8: val_loss improved from 0.69320 to 0.69317, saving model to best_model_2m.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit
_compiled_convolution_op while saving (showing 5 of 6). These functions will not be d
irectly callable after loading.
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
625/625 - 2456s - loss: 0.6933 - accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy:
0.5000 - 2456s/epoch - 4s/step
Epoch 9/20

Epoch 9: val_loss did not improve from 0.69317
625/625 - 2402s - loss: 0.6933 - accuracy: 0.5000 - val_loss: 0.6935 - val_accuracy:
0.4996 - 2402s/epoch - 4s/step
Epoch 10/20

Epoch 10: val_loss improved from 0.69317 to 0.69315, saving model to best_model_2m.cn
n
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit
_compiled_convolution_op while saving (showing 5 of 6). These functions will not be d
irectly callable after loading.
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
INFO:tensorflow:Assets written to: best_model_2m.cnn\assets
625/625 - 2555s - loss: 0.6934 - accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy:
0.4998 - 2555s/epoch - 4s/step
Epoch 11/20
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_3120\2322926065.py in <module>
      1 # fit model using batches of training data and batches of testing data
----> 2 history = model.fit(train_generator, steps_per_epoch=train_generator.samples
// batch_size,
      3                               validation_data = validation_generator,
      4                               validation_steps = validation_generator.samples
// batch_size,
      5                               epochs = epochs,

~\anaconda3\lib\site-packages\keras\utils\traceback_utils.py in error_handler(*args,
**kwargs)
     63         filtered_tb = None
     64         try:
---> 65             return fn(*args, **kwargs)
     66         except Exception as e:
     67             filtered_tb = _process_traceback_frames(e.__traceback__)

~\anaconda3\lib\site-packages\keras\engine\training.py in fit(self, x, y, batch_size,
epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight,
sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_batch_siz
e, validation_freq, max_queue_size, workers, use_multiprocessing)
   1683                             ):
   1684                                 callbacks.on_train_batch_begin(step)
-> 1685                                 tmp_logs = self.train_function(iterator)
   1686                                 if data_handler.should_sync:
   1687                                     context.async_wait()

~\anaconda3\lib\site-packages\tensorflow\python\util\traceback_utils.py in error_hand
ler(*args, **kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = _process_traceback_frames(e.__traceback__)

~\anaconda3\lib\site-packages\tensorflow\python\eager\polymorphic_function\polymorphi
c_function.py in __call__(self, *args, **kwds)
    892 
    893       with OptionalXlaContext(self._jit_compile):
--> 894         result = self._call(*args, **kwds)
    895 
    896       new_tracing_count = self.experimental_get_tracing_count()

~\anaconda3\lib\site-packages\tensorflow\python\eager\polymorphic_function\polymorphi
c_function.py in _call(self, *args, **kwds)
    924       # In this case we have created variables on the first call, so we run t
he
    925       # defunned version which is guaranteed to never create variables.
--> 926       return self._no_variable_creation_fn(*args, **kwds)  # pylint: disable=
not-callable
    927     elif self._variable_creation_fn is not None:
    928       # Release the lock early so that multiple threads can perform the call

~\anaconda3\lib\site-packages\tensorflow\python\eager\polymorphic_function\tracing_co
mpiler.py in __call__(self, *args, **kwargs)
    141       (concrete_function,
    142        filtered_flat_args) = self._maybe_define_function(args, kwargs)
--> 143     return concrete_function._call_flat(
    144         filtered_flat_args, captured_inputs=concrete_function.captured_input
s)  # pylint: disable=protected-access
    145 

~\anaconda3\lib\site-packages\tensorflow\python\eager\polymorphic_function\monomorphi
c_function.py in _call_flat(self, args, captured_inputs, cancellation_manager)
   1755         and executing_eagerly):
   1756       # No tape is watching; skip to running the function.
-> 1757       return self._build_call_outputs(self._inference_function.call(
   1758           ctx, args, cancellation_manager=cancellation_manager))
   1759     forward_backward = self._select_forward_and_backward_functions(

~\anaconda3\lib\site-packages\tensorflow\python\eager\polymorphic_function\monomorphi
c_function.py in call(self, ctx, args, cancellation_manager)
    379       with _InterpolateFunctionError(self):
    380         if cancellation_manager is None:
--> 381           outputs = execute.execute(
    382               str(self.signature.name),
    383               num_outputs=self._num_outputs,

~\anaconda3\lib\site-packages\tensorflow\python\eager\execute.py in quick_execute(op_
name, num_outputs, inputs, attrs, ctx, name)
     50   try:
     51     ctx.ensure_initialized()
---> 52     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     53                                         inputs, attrs, num_outputs)
     54   except core._NotOkStatusException as e:

KeyboardInterrupt: 
```

In [46]:

```
# show best model
best_model = tf.keras.models.load_model('best_model_2m.cnn')
best_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_5 (Conv2D)           (None, 144, 144, 64)      9472      
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 72, 72, 64)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 70, 70, 128)       73856     
                                                                 
 conv2d_7 (Conv2D)           (None, 68, 68, 128)       147584    
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 34, 34, 128)      0         
 2D)                                                             
                                                                 
 conv2d_8 (Conv2D)           (None, 32, 32, 256)       295168    
                                                                 
 conv2d_9 (Conv2D)           (None, 30, 30, 256)       590080    
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 15, 15, 256)      0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 57600)             0         
                                                                 
 dense_3 (Dense)             (None, 128)               7372928   
                                                                 
 dropout_2 (Dropout)         (None, 128)               0         
                                                                 
 dense_4 (Dense)             (None, 64)                8256      
                                                                 
 dropout_3 (Dropout)         (None, 64)                0         
                                                                 
 dense_5 (Dense)             (None, 1)                 65        
                                                                 
=================================================================
Total params: 8,497,409
Trainable params: 8,497,409
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
import keras
from matplotlib import pyplot as plt
history = model1.fit(train_x, train_y,validation_split = 0.2, epochs=4, batch_size=4)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

In [50]:

```
# Apply the best model to the training / validation dataset
model_2m_pred_validation = best_model.predict(validation_generator, verbose = 1)

# Put the label predictions into a dataframe
model_2m_pred_validation_df = pd.DataFrame(model_2m_pred_validation, columns=['label']
```

```
157/157 [==============================] - 150s 951ms/step
```

In [51]:

```
model_2m_pred_validation_array = model_2m_pred_validation_df['label'].to_numpy()
```

In [56]:

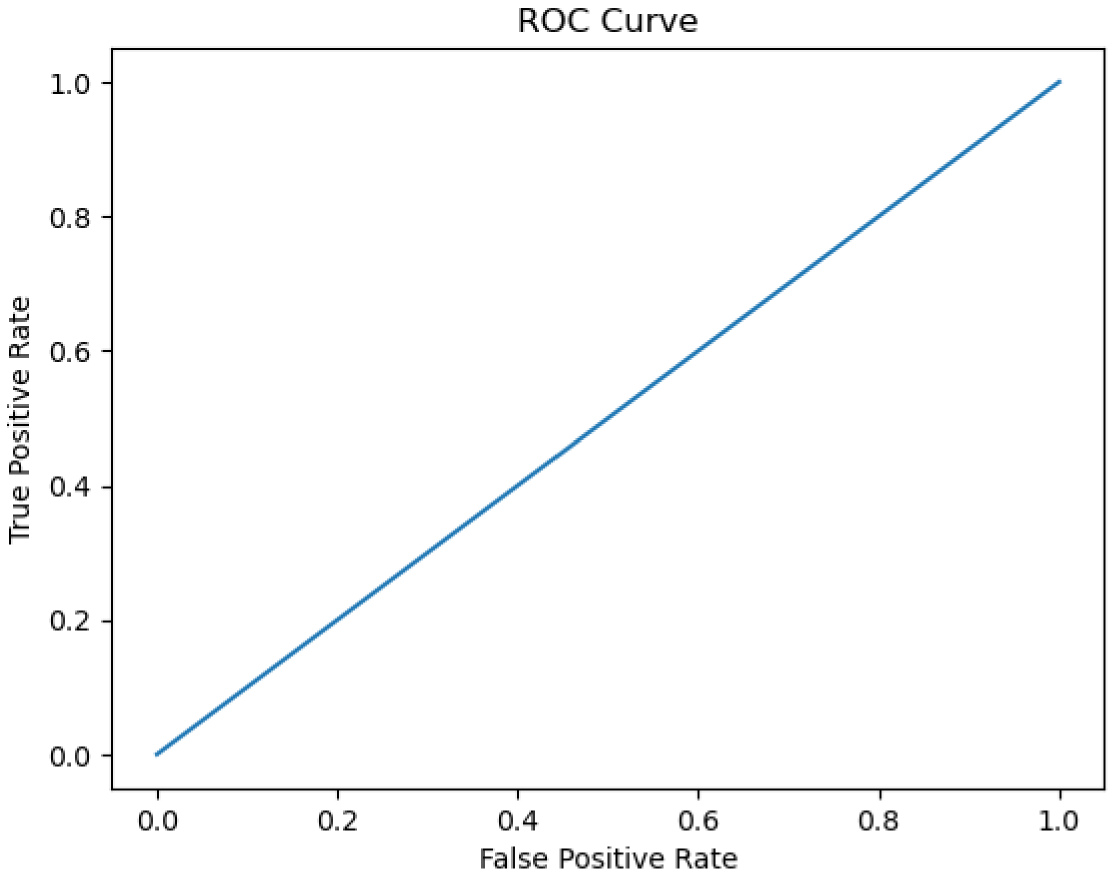
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import precision_recall_curve, auc
from sklearn.metrics import PrecisionRecallDisplay
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, roc_curve
from matplotlib import pyplot as plt


true_y = validation_generator.labels


# Curves

fpr, tpr, _ = roc_curve(true_y, model_2m_pred_validation_array)
roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr).plot()
plt.title('ROC Curve')
# roc auc score
auc1 = roc_auc_score(true_y, model_2m_pred_validation_array)
print("The roc auc score is:", auc1)

prec, recall, _ = precision_recall_curve(true_y, model_2m_pred_validation_array)
pr_display = PrecisionRecallDisplay(precision=prec, recall=recall).plot()
plt.title('Precision-Recall Curve')# precision-recall auc score
auc2 = auc(recall, prec)
print("The prec-recall auc score is:", auc2)
```

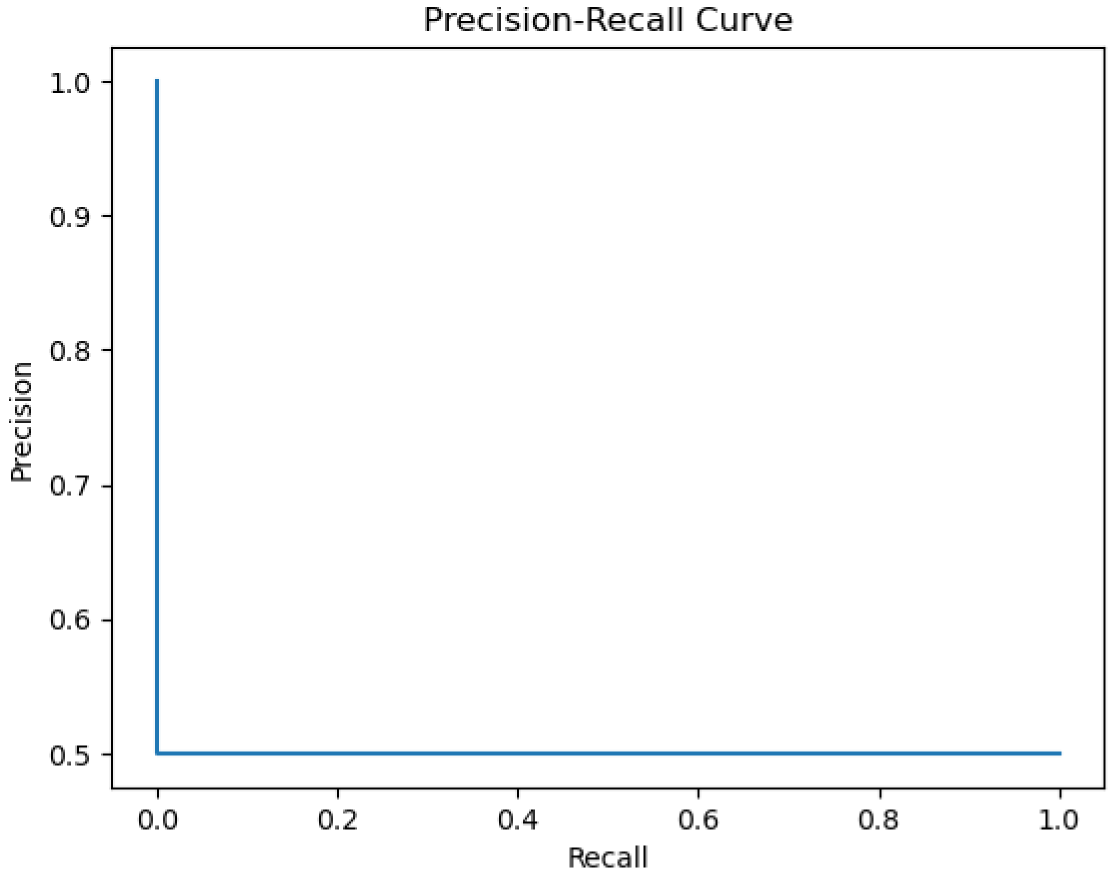
Out[56]:

```
Text(0.5, 1.0, 'ROC Curve')

The roc auc score is: 0.5
```

Out[56]:

```
Text(0.5, 1.0, 'Precision-Recall Curve')

The prec-recall auc score is: 0.75
```





In [57]:
```
# rescale image pixels
test_gen = ImageDataGenerator(rescale = 1./255)
# create test generator for testing data
```

```
test_generator = test_gen.flow_from_dataframe(test_df,
                                               directory='test_cropped',
                                               x_col='filename',
                                               class_mode= None,
                                               target_size=(image_size,image_size),
                                               batch_size=batch_size,
                                               shuffle=False
)
```

```
Found 12500 validated image filenames.
```

In [58]:
```
# Apply the cnn model1 to the test dataset
model_2m_pred2 = best_model.predict(test_generator, verbose = 1)

# Put the label predictions into a dataframe
model_2m_pred2_df = pd.DataFrame(model_2m_pred2, columns=['label'])

# Add the ID column to the front of the cnn predictions dataframe
model_2m_pred2_df.insert(0, 'id', test_df['id'])

# Output predictions to csv
model_2m_pred2_df.to_csv('model_2_test predictions.csv', index=False)
```

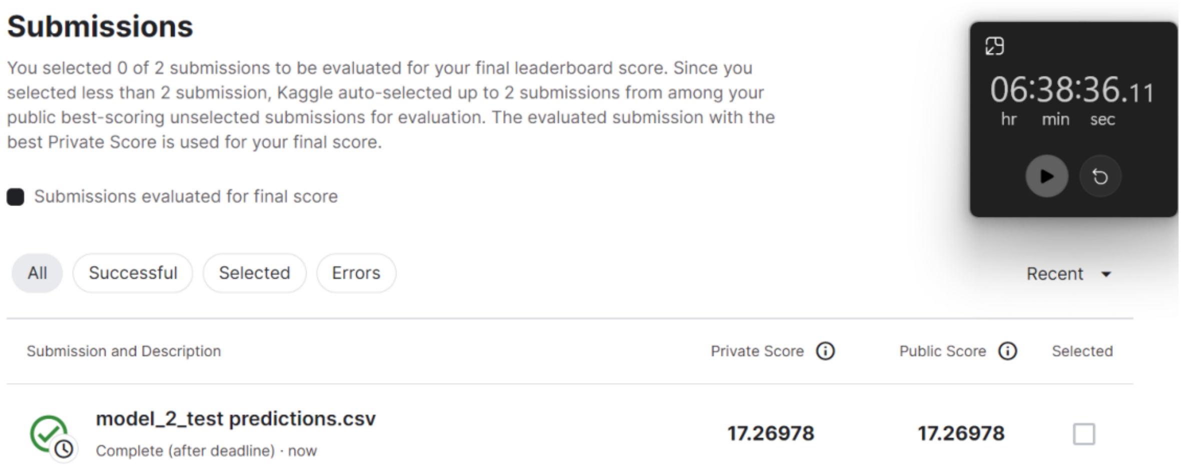
```
391/391 [==============================] - 415s 1s/step
```

In [60]:
```
# Display the kaggle results (log loss) of CNN model
import matplotlib.pyplot as plt
plt.figure(figsize = (15, 15))
kaggle_results = plt.imread('Kaggle_results_Model_2.png')
plt.imshow(kaggle_results)
plt.axis("off")
plt.show()
```

Out[60]:
```
<Figure size 1500x1500 with 0 Axes>
```

Out[60]:
```
<matplotlib.image.AxesImage at 0x2270338c3d0>
```

Out[60]:
```
(-0.5, 1476.5, 641.5, -0.5)
```



Submissions

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

● Submissions evaluated for final score

Submission and Description	Private Score ⓘ	Public Score ⓘ	Selected
model_2_test predictions.csv Complete (after deadline) · now	17.26978	17.26978	□

CNN Model 3

In [36]:
```
# Build Initial Sequential Model
model = Sequential()
# 2D convolutional layer w/64 filters, 3x3 kernel, and ReLU activation function. Model
model.add(Conv2D(input_shape=(150,150,3), filters = 64, kernel_size=(3,3), activation=
# max pooling layer
model.add(MaxPool2D(pool_size=(2,2)))

# Conv2D and MaxPooling2D layers with 32 filters and 3x3 kernel.
model.add(Conv2D(filters = 32, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))

# Conv2D and MaxPooling2D layers with 16 filters 3x3 kernel.
model.add(Conv2D(filters = 16, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))

# Conv2D and MaxPooling2D layers with 8 filters 3x3 kernel.
model.add(Conv2D(filters = 8, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))

# add dropout to avoid overfitting
model.add(Dropout(0.25))

# flatten output of the previous layer - converts to 1d layer
model.add(Flatten())

# add dense layer
model.add(Dense(32, activation='relu'))

# add dropout to avoid overfitting
model.add(Dropout(0.5))

# add dense layer
model.add(Dense(units=1, activation="sigmoid"))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 148, 148, 64)      1792      
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 74, 74, 64)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 72, 72, 32)        18464     
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 36, 36, 32)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 34, 34, 16)        4624      
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 17, 17, 16)       0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 15, 15, 8)         1160      
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 7, 7, 8)          0         
 2D)                                                             
                                                                 
 dropout_2 (Dropout)         (None, 7, 7, 8)           0         
                                                                 
 flatten_1 (Flatten)         (None, 392)               0         
                                                                 
 dense_2 (Dense)             (None, 32)                12576     
                                                                 
 dropout_3 (Dropout)         (None, 32)                0         
                                                                 
 dense_3 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 38,649
Trainable params: 38,649
Non-trainable params: 0
_________________________________________________________________
```

In [37]:

```python
# Adam solver optimizer with learning rate of 0.001
optimizer=tf.keras.optimizers.Adam(learning_rate=0.001)

# compile model
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

In [38]:

```python
import warnings
warnings.filterwarnings('ignore')
# early stopping based on validation loss (stops if model doesn't improve after 5 iter
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)

# save the best model as 'best_model.cnn' based on validation loss
save_best = ModelCheckpoint(filepath = 'best_model.cnn', verbose=1, save_best_only=Tru

# fit model using batches of training data and batches of testing data
history = model.fit(train_generator, steps_per_epoch=train_generator.samples // batch_
                              validation_data = validation_generator,
                              validation_steps = validation_generator.samples // batch
                              epochs = epochs,
                              callbacks=[save_best, early_stopping],
                              verbose=2)
```

```
Epoch 1/20

Epoch 1: val_loss improved from inf to 0.68135, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 282s - loss: 0.6904 - accuracy: 0.5263 - val_loss: 0.6814 - val_accuracy:
0.5563 - 282s/epoch - 450ms/step
Epoch 2/20

Epoch 2: val_loss improved from 0.68135 to 0.66792, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 266s - loss: 0.6795 - accuracy: 0.5624 - val_loss: 0.6679 - val_accuracy:
0.5895 - 266s/epoch - 426ms/step
Epoch 3/20

Epoch 3: val_loss improved from 0.66792 to 0.64925, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 280s - loss: 0.6667 - accuracy: 0.5946 - val_loss: 0.6492 - val_accuracy:
0.6082 - 280s/epoch - 448ms/step
Epoch 4/20

Epoch 4: val_loss improved from 0.64925 to 0.62552, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 282s - loss: 0.6581 - accuracy: 0.6147 - val_loss: 0.6255 - val_accuracy:
0.6534 - 282s/epoch - 452ms/step
Epoch 5/20

Epoch 5: val_loss improved from 0.62552 to 0.60479, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 341s - loss: 0.6326 - accuracy: 0.6520 - val_loss: 0.6048 - val_accuracy:
0.6803 - 341s/epoch - 545ms/step
Epoch 6/20

Epoch 6: val_loss improved from 0.60479 to 0.57591, saving model to best_model.cnn
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 341s - loss: 0.6079 - accuracy: 0.6747 - val_loss: 0.5759 - val_accuracy:
0.7113 - 341s/epoch - 546ms/step
Epoch 7/20

Epoch 7: val_loss improved from 0.57591 to 0.55527, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 360s - loss: 0.5928 - accuracy: 0.6916 - val_loss: 0.5553 - val_accuracy:
0.7135 - 360s/epoch - 576ms/step
Epoch 8/20

Epoch 8: val_loss improved from 0.55527 to 0.54201, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 343s - loss: 0.5670 - accuracy: 0.7107 - val_loss: 0.5420 - val_accuracy:
0.7302 - 343s/epoch - 548ms/step
Epoch 9/20

Epoch 9: val_loss improved from 0.54201 to 0.52909, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 332s - loss: 0.5505 - accuracy: 0.7269 - val_loss: 0.5291 - val_accuracy:
0.7402 - 332s/epoch - 531ms/step
Epoch 10/20

Epoch 10: val_loss did not improve from 0.52909
625/625 - 329s - loss: 0.5392 - accuracy: 0.7330 - val_loss: 0.5486 - val_accuracy:
0.7310 - 329s/epoch - 527ms/step
Epoch 11/20

Epoch 11: val_loss improved from 0.52909 to 0.50948, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
```

```
625/625 - 325s - loss: 0.5284 - accuracy: 0.7411 - val_loss: 0.5095 - val_accuracy:
0.7474 - 325s/epoch - 520ms/step
Epoch 12/20

Epoch 12: val_loss did not improve from 0.50948
625/625 - 319s - loss: 0.5201 - accuracy: 0.7487 - val_loss: 0.5163 - val_accuracy:
0.7482 - 319s/epoch - 511ms/step
Epoch 13/20

Epoch 13: val_loss improved from 0.50948 to 0.46880, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 349s - loss: 0.5032 - accuracy: 0.7581 - val_loss: 0.4688 - val_accuracy:
0.7871 - 349s/epoch - 558ms/step
Epoch 14/20

Epoch 14: val_loss improved from 0.46880 to 0.46452, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 366s - loss: 0.4968 - accuracy: 0.7675 - val_loss: 0.4645 - val_accuracy:
0.7817 - 366s/epoch - 585ms/step
Epoch 15/20

Epoch 15: val_loss did not improve from 0.46452
625/625 - 328s - loss: 0.4923 - accuracy: 0.7681 - val_loss: 0.4677 - val_accuracy:
0.7778 - 328s/epoch - 525ms/step
Epoch 16/20

Epoch 16: val_loss improved from 0.46452 to 0.45175, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 342s - loss: 0.4882 - accuracy: 0.7739 - val_loss: 0.4517 - val_accuracy:
0.7891 - 342s/epoch - 548ms/step
Epoch 17/20

Epoch 17: val_loss did not improve from 0.45175
625/625 - 368s - loss: 0.4737 - accuracy: 0.7819 - val_loss: 0.4813 - val_accuracy:
0.7720 - 368s/epoch - 588ms/step
Epoch 18/20

Epoch 18: val_loss improved from 0.45175 to 0.44220, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
```

```
625/625 - 308s - loss: 0.4665 - accuracy: 0.7838 - val_loss: 0.4422 - val_accuracy:
0.7943 - 308s/epoch - 492ms/step
Epoch 19/20

Epoch 19: val_loss improved from 0.44220 to 0.43580, saving model to best_model.cnn
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_comp
iled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while
saving (showing 4 of 4). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: best_model.cnn\assets
INFO:tensorflow:Assets written to: best_model.cnn\assets
625/625 - 323s - loss: 0.4616 - accuracy: 0.7896 - val_loss: 0.4358 - val_accuracy:
0.8017 - 323s/epoch - 516ms/step
Epoch 20/20

Epoch 20: val_loss did not improve from 0.43580
625/625 - 311s - loss: 0.4560 - accuracy: 0.7920 - val_loss: 0.4376 - val_accuracy:
0.8035 - 311s/epoch - 497ms/step
```

In [39]:

```
# show best model
best_model = tf.keras.models.load_model('best_model.cnn')
best_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 148, 148, 64)      1792      
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 74, 74, 64)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 72, 72, 32)        18464     
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 36, 36, 32)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 34, 34, 16)        4624      
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 17, 17, 16)       0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 15, 15, 8)         1160      
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 7, 7, 8)          0         
 2D)                                                             
                                                                 
 dropout_2 (Dropout)         (None, 7, 7, 8)           0         
                                                                 
 flatten_1 (Flatten)         (None, 392)               0         
                                                                 
 dense_2 (Dense)             (None, 32)                12576     
                                                                 
 dropout_3 (Dropout)         (None, 32)                0         
                                                                 
 dense_3 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 38,649
Trainable params: 38,649
Non-trainable params: 0
_________________________________________________________________
```

Generate Precision-Recall and Receiving Operating Characteristic Curves

In [73]:

```
# Apply the cnn model1 to the training / validation dataset
cnn_pred3_validation = best_model.predict(validation_generator, verbose = 1)

# Put the label predictions into a dataframe
cnn_pred3_validation_df = pd.DataFrame(cnn_pred3_validation, columns=['label'])
```

```
157/157 [==============================] - 33s 211ms/step
```

In [78]:

```
cnn_pred3_validation_array = cnn_pred3_validation_df['label'].to_numpy()
```

Out[78]:

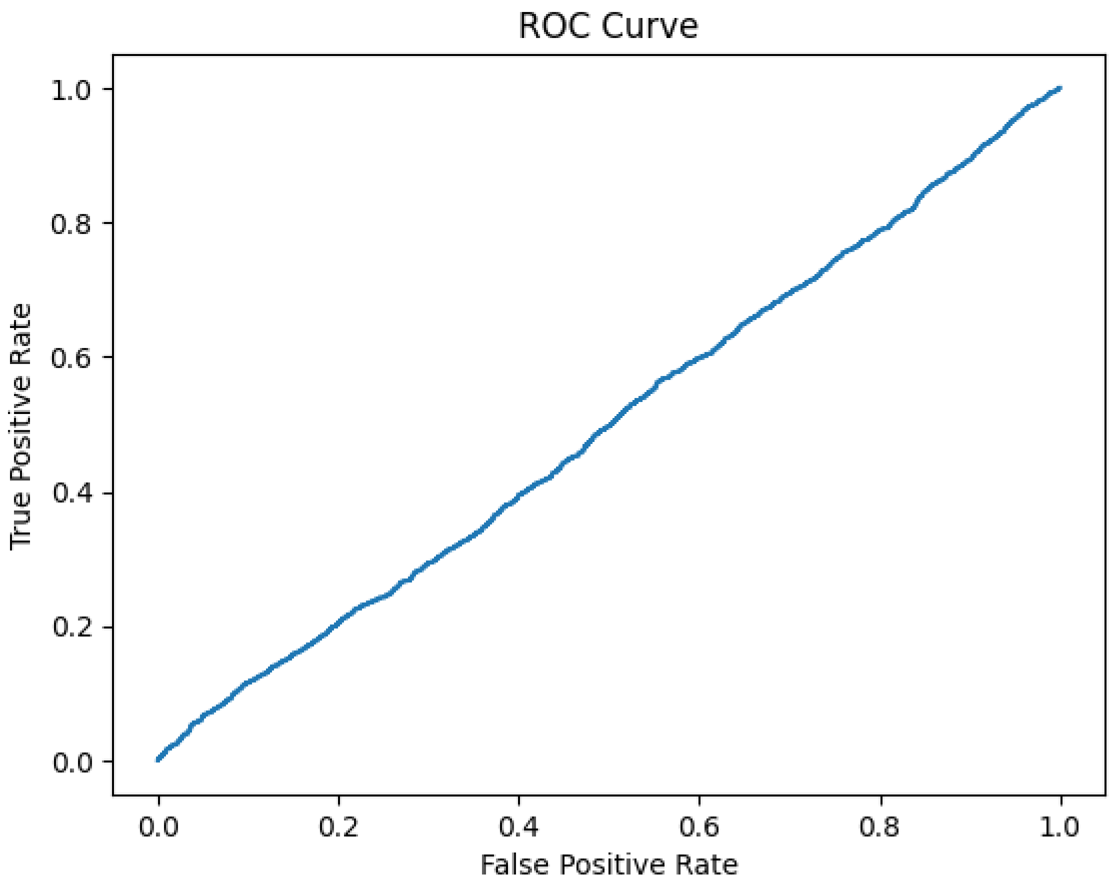
```
array([0.3702017 , 0.54691535, 0.9955137 , ..., 0.13030599, 0.39366305,
       0.7284142 ], dtype=float32)
```

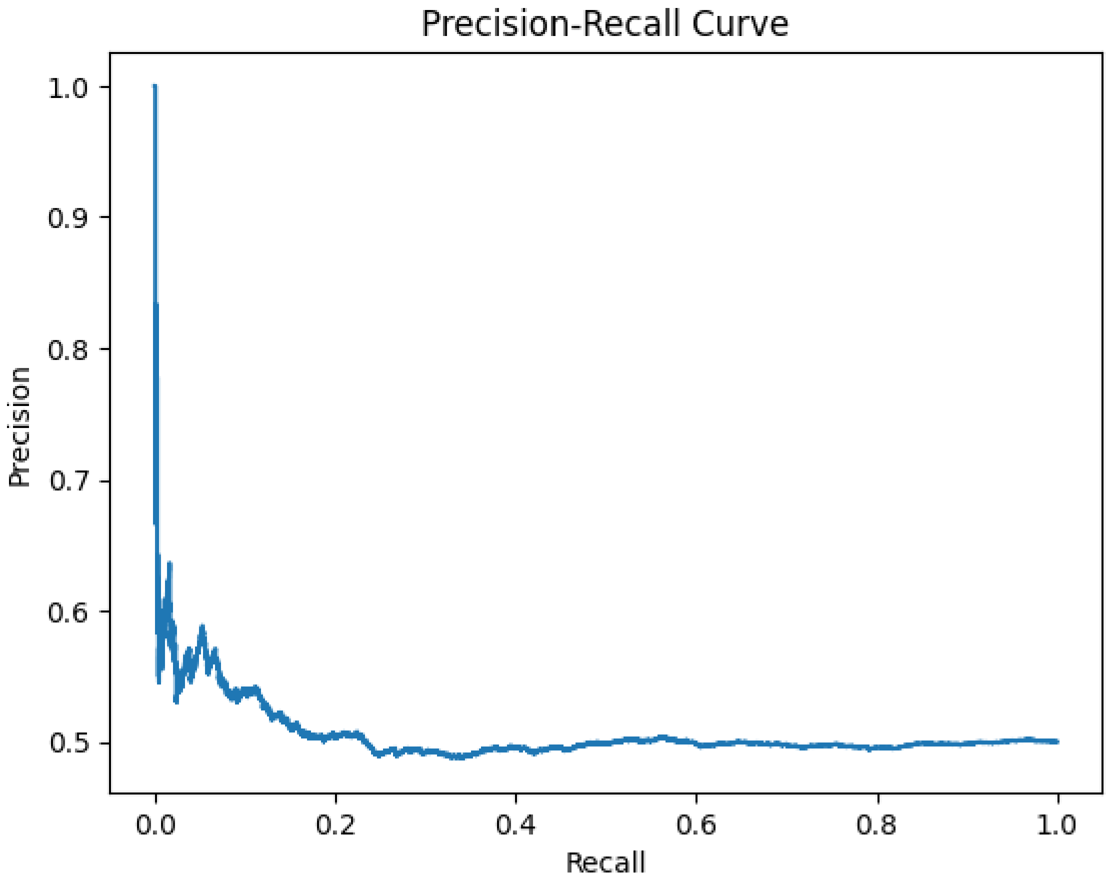
In [83]:

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import precision_recall_curve, auc
from sklearn.metrics import PrecisionRecallDisplay
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, roc_curve
```

```
true_y = validation_generator.labels


# Curves

fpr, tpr, _ = roc_curve(true_y, cnn_pred3_validation_array)
roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr).plot()
plt.title('ROC Curve')
# roc auc score
auc1 = roc_auc_score(true_y, cnn_pred3_validation_array)
print("The roc auc score is:", auc1)

prec, recall, _ = precision_recall_curve(true_y, cnn_pred3_validation_array)
pr_display = PrecisionRecallDisplay(precision=prec, recall=recall).plot()
plt.title('Precision-Recall Curve')# precision-recall auc score
auc2 = auc(recall, prec)
print("The prec-recall auc score is:", auc2)
```

```
Out[83]: Text(0.5, 1.0, 'ROC Curve')

The roc auc score is: 0.49759144
Out[83]: Text(0.5, 1.0, 'Precision-Recall Curve')

The prec-recall auc score is: 0.5063294941649535
```





Input Testing Data into Testing Generator for Model Predictions

In [40]:

```
# rescale image pixels
test_gen = ImageDataGenerator(rescale = 1./255)
# create test generator for testing data
test_generator = test_gen.flow_from_dataframe(test_df,
                                               directory='test_cropped',
                                               x_col='filename',
                                               class_mode= None,
                                               target_size=(image_size,image_size),
                                               batch_size=batch_size,
                                               shuffle=False
)
```

```
Found 12500 validated image filenames.
```

Save Predictions into CSV for Kaggle

In [41]:

```
# Apply the cnn model1 to the test dataset
cnn_pred3 = best_model.predict(test_generator, verbose = 1)

# Put the label predictions into a dataframe
cnn_pred3_df = pd.DataFrame(cnn_pred3, columns=['label'])

# Add the ID column to the front of the cnn predictions dataframe
cnn_pred3_df.insert(0, 'id', test_df['id'])

# Output predictions to csv
cnn_pred3_df.to_csv('test_predictions_cnn_v3.csv', index=False)
```

```
391/391 [==============================] - 49s 125ms/step
```

Display the Kaggle results from the application of the CNN model on the test dataset

In [43]:

```
# Display the kaggle results (log loss) of CNN model
import matplotlib.pyplot as plt
plt.figure(figsize = (15, 15))
kaggle_results = plt.imread('Kaggle_results_cnn_v3.jpg')
plt.imshow(kaggle_results)
plt.axis("off")
plt.show()
```

Out[43]:

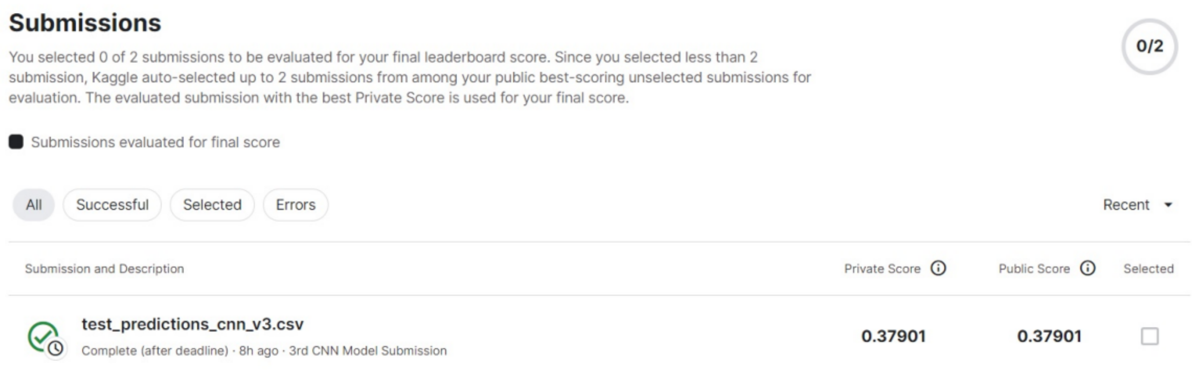
```
<Figure size 1500x1500 with 0 Axes>
```

Out[43]:

```
<matplotlib.image.AxesImage at 0x1f365c058b0>
```

Out[43]:

```
(-0.5, 1501.5, 506.5, -0.5)
```



Submissions

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

■ Submissions evaluated for final score

Submission and Description	Private Score	Public Score	Selected
test_predictions_cnn_v3.csv Complete (after deadline) · 8h ago · 3rd CNN Model Submission	0.37901	0.37901	□

In [ ]: