Mask Detection using Convolutional Neural Networks

Riken Patel
ECE 491: Neural Networks
University of Illinois at Chicago
rpate322@uic.edu
Jakub Marek
ECE 491: Neural Networks
University of Illinois at Chicago
jmarek6@uic.edu

***Abstract*— This project is centered around COVID-19, the major health concern the world is facing currently. The focus of reducing the spread of the virus is around face masks, and the laws that many world institutions are putting in place mandating their wear in public places. This project proposes a method of image analysis that uses Convolution Neural Networks (CNN) to classify images into three separate categories: face masks, no face mask, or incorrectly worn face mask. In this project, a tool is created that anyone can use to gather information if the individuals are wearing face masks, not wearing a face mask, or wearing their face mask incorrectly quickly and easily.**

Keywords- COVID-19, Convolutional Neural Networks, CNN, face masks, image processing.

I. INTRODUCTION

As of April 26th, 2021, there have been a total of 146,841,882 confirmed cases of COVID-19, including 3,104,743 deaths [1]. Due to the nature of COVD-19 and its mechanism of transmission, world health professionals urged the world population to begin wearing face masks. These masks play a crucial role for protecting the health of individuals as it is the main barrier against COVID-19 and containing the virus. Knowing that COVID-19 is a disease that is transmitted through air (airborne), masks are a great first barrier of protection. Therefore, world governments began mandating the use of face masks. Currently, most states in the United States have enacted a mandate that requires face mask use in public areas. Therefore, it is important to create a system that can easily and quickly determine if individuals are wearing masks allowing for the enforcement of the mandates.

In this paper, a method is proposed that can take images and conclude whether the individuals in the image are wearing their face masks correctly, not wearing a face mask, or are wearing the face mask incorrectly. The method used for this specific application is Convolutional Neural Networks (CNN) which is a discriminative supervised deep learning algorithm and model that can extract features from images and one-dimensional signals and classify them automatically. Using this model, this project can determine whether individuals are wearing masks correctly in public environments helping them control the spread of the virus by enforcing individuals to wear face masks correctly.

II. METHOD DESCRIPTION

Dataset

The dataset that was used for this project was obtained from Kaggle. The dataset is named "Face Mask Detection" and it contains two sets of data: images and annotations [2]. The dataset contains 853 images that belong to 3 different classes which include: with mask, without mask, and mask worn incorrectly. The images are in the .png file type and the annotations are in the .xml. These .xml annotation files contained information regarding the images that they were tied to. The images contained people that met the description of one of the three different classifications. Some of these images had more than one individual, and each were made as an object under the annotation file. In addition, each one of these objects, sometimes more than one per image, contained coordinates of a box that focused on the face and a label designating the with mask, without mask, and mask worn incorrectly labels.

For example, using the image shown in Figure 1, two individuals are pictured, both have face masks that are worn correctly. Now referring to Figure 2, which is the corresponding plantation file that matches the image, the file states the image title, image size and then it shows the object data. Under the object designation labels follow stating whether the object has a mask and some other non-relevant data for this project. Since there are only two objects in people in this file two objects were created and boxes were drawn around the objects faces with a coordinate of this box specified as the: xmin, ymin, xmax, and ymax.

Figure 1: "maksssksksss208.png" from the Dataset.

```
- <annotation>
    <folder>images</folder>
    <filename>maksssksksss208.png</filename>
  - <size>
      <width>400</width>
      <height>200</height>
      <depth>3</depth>
    </size>
    <segmented>0</segmented>
  - <object>
      <name>with_mask</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <occluded>0</occluded>
      <difficult>0</difficult>
    - <bndbox>
        <xmin>84</xmin>
        <ymin>82</ymin>
        <xmax>194</xmax>
        <ymax>170</ymax>
      </bndbox>
    </object>
  - <object>
      <name>with_mask</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <occluded>0</occluded>
      <difficult>0</difficult>
    - <bndbox>
        <xmin>192</xmin>
        <ymin>71</ymin>
        <xmax>298</xmax>
        <ymax>187</ymax>
      </bndbox>
    </object>
</annotation>
```

Figure 2: "maksssksksss208.xml" form the Dataset

Preprocessing Data

For this project, preprocessing of the data was necessary so that the data of the images can be entered into the CNN. The data needed to be separated out and labeled individually to allow CNN to run flawlessly. Inspiration for this section was obtained from a Kaggle code published under the original dataset named "Pytorch - FasterRCNN" [3]. This first step of the code was to generate a matrix that would hold the image title with all its corresponding objects and their labels of mask. The mask data of without mask, with mask, and mask worn incorrectly was labeled 0,1,2, respectively. To do this, boxes, labels, and image ID matrices were created with all the aforementioned information and appended into a target matrix with all their information under the correct images. Next, the boxes from the objects that were found in the images, were extracted as a .png file of only the boxes and saved into a separate file, box images. Once all the objects were gathered, they were then labeled with their correct original image number, box/object number, and the label corresponding to its mask classification. Lastly, the images were all resized to a 50x50x3 image so that the CNN could work properly, and the white box was removed from the images.

Figure 3: Preprocessed image

CNN

Convolutional Neural Networks (CNN) is a discriminative supervised deep learning algorithm and model that can extract features from images and one-dimensional signals and classify them automatically [4]. CNN's can assign weights and biases to many aspects in the image to be able to differentiate one feature from the other. This is dependent by pixels/parameters of the image. It is based on many linear filters and non-linear operations [4]. CNN's are like multi-layer perceptrons except they have hidden layers called convolutional layers. These convolutional layers receive and transform inputs. This transformation is referred to as a convolutional process which enables the convolutional layers to detect patterns in images [5]. For example, in this application, the layers were to detect a mask -- whether the person is wearing a face mask correctly, incorrectly, or not wearing a mask at all within the image. Thus, this model was trained to detect this specific pattern of the object (face masks) in several images. The layers are square matrices of a certain size and it will convolve across these matrices as it receives input (image). The square matrices are a certain feature detector (i.e., face masks). In mathematical terms, the dot product will be taken between the feature detector/filter/kernel (square matrices) and image (input). The convolving also depends on the stride value. The stride is basically the value the filter/kernel slides across on the input image.

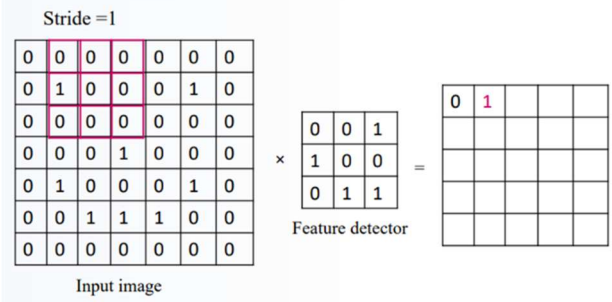

Figure 4: Feature detector kernel convolves across the input image with a stride of 1 taking the dot product to output the new image.

In this project, CNN's were utilized to depict, extract, and classify whether the person in the image is wearing a mask, not wearing a mask, or wearing the mask incorrectly. Tensorflow and Keras were used within Python 3.0 to train the model. Specifically, the sequential model was used from Tensorflow because this model stacks the layers layer-by-layer. There was only one input tensor and one output tensor [6]. A tensor is a multidimensional array. A first order tensor is a one-dimensional vector, whereas a two-dimensional matrix is a second order tensor [7]. In this model, conv2d layer, max pooling 2d, flatten, and dense were utilized. Also, cross-entropy loss function and adam optimizer was used to compile the model. Conv2d is a two-dimensional convolutional layer that does a spatial convolution across the images. This layer produces a convolutional kernel that moves across the input images to create an output tensor. Conv2d has the ability to create a bias vector that can be added to the outputs [8]. In this project, an activation of ReLU (rectified linear unit) and an input shape of 50x50x3 is used. Also, the filters of 32 and 64 are used. This is the dimensionality of the output space. A 3x3 kernel size is used for Conv2d. The input shape is 50x50x3 because those are the dimensions for each of the images in the dataset. ReLU generates an image out of the filtered image, then it combines both images. ReLU is preferred in deep learning neural network models than tanh or sigmoid as the activation function because ReLU is faster in computation and derivation [9]. Thus, it is quicker in inference and training time for neural networks and convergence of randomly determined gradient descents. ReLU is also a much easier activation function to implement into a system (inexpensive operation) as it simply needs to threshold a matrix of zeros. Max pooling 2d was also used in the model to determine the pool size (which is the window size of max pooling, in this case it is a 2x2). Max pooling also includes strides which is how far the pooling moves for each pooling step [10]. There were strides specified so a stride of 1 is utilized by the max pooling window. There is no padding on the input image or kernel. Furthermore, flatten was added to this model as well and this simply flattens the input image. Moreover, dense is also added to the mode. Dense layer is a regular deeply connected neural network layer [11]. In this case, the input of 64 and 10 are used which is the input image data; the kernel represents the weight data; and the activation is the ReLU function as already explained. A loss/cost function of cross-entropy was used to compute the loss between the predictions and labels [12]. As we had more than one labels (labels: 0 for no mask, 1 for correct mask, and 2 for incorrect mask) and this loss function is best to use when there are multiple labels in use. The training parameters/optimizer being used is adam. Adam optimizer is an adaptive learning rate algorithm that is used for training deep neural networks such as CNN's. Adam optimizer utilizes the advantages of Adagrad and RMSprop optimizers [13]. Model evaluation was also used to take the average of the accuracies from the 25 epochs and the average of the loss function; that is the accuracy and loss that is outputted in the terminal. This designed convolutional neural network model gave us the best accuracy for this dataset.

```
model = Sequential()
model.add(Conv2D(32,(3,3), activation='relu', input_shape=(50,50,3)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dense(10))
print(model.summary())
model.compile(optimizer='adam',
              loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
monitor = EarlyStopping(monitor='accuracy',patience = 3 , verbose=1, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=5, callbacks=[monitor])
loss,acc = model.evaluate(X_test,y_test,verbose = 2)
print('Accuracy of the model: ',round(acc,4)*100,'%')

random_array = np.random.randint(len(y_test), size=(1,5))[0]
```

Figure 5: Convolutional Neural Network (CNN) model used for "Face Mask Detection" dataset.

Programming process

Python 3.0 was used to create a convolutional neural network of the "Face Mask Detection" dataset. The following libraries were utilized to program and train the neural network: NumPy, Pandas, Tensorflow, Keras, Scikit learn (Sklearn), Matplotlib, torchvision, beautifulsoup4, os, cv2 and random.

First, the preprocessing of the dataset was done in the preprocessing.py file. There were two files for the dataset: images and annotations. The dataset was read into our IDE (Visual Studio Code or Spyder) and organized. The images file included all 853 images, and the annotations file included all the labels for the images. The annotations were labeled per person per image, so if the image had multiple people, it would have multiple labels for that one image. The annotations would give each person in the image a label of 0 if the person had no mask on, 1 if the person wore the mask correctly, and 2 if the person wore the mask incorrectly. As some images included more than one person in the image, each person's face in each image was extracted out using the labels from the annotations. Thus, the 853 images became 3882 images saved in another file. These images were refined and restructured into a 50x50x3 to make all 3882 extracted images the same size. This allows the CNN model to run more smoothly via each image. All this preprocessing data was saved in the preprocessing.py file.

Next, this preprocessing.py file was imported into the 491project.py file which is our main file with the CNN model. The extracted images and extracted labels were called upon and organized into two different arrays: images data and labels. Then, the images data was split into 80% training and 20% test sets with a random state of 42. Lastly, the CNN model was applied on the training and test datasets outputting the accuracy after 25 epochs. An early stopping mechanism was put in place so if the accuracy of the model did not change for 3 iterations of epoch, then it would stop the epoch and output the evaluated accuracy and loss. This also outputted images of whether the person in the image did not wear a mask (0), wore a mask (1), or wore the mask incorrectly (2). The model was able to output the actual and predicted labels in the terminal that correspond to the respective image.

III. Experimental Results

The accuracy of the model varies between 90-92% with a mean and median of approximately 91% after 25 epochs (figure 10). Furthermore, 20 images were also outputted to test the model's accuracy. As you can see in figures 6 to 9 below, the actual and predicted labels are outputted along with its respective image. In most cases, the actual and predicted values matched with the model (figures 6 to 8), but it does predict wrong sometimes if the image is harder to depict (figure 9). The accuracy of the model depends on the random images it takes to evaluate and the number of epochs it can run through. The figures below demonstrate some sample outputs that were obtained running the algorithmic program.

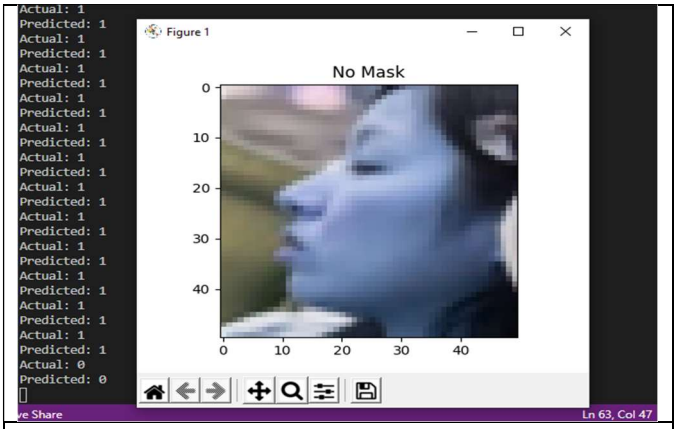

Figure 6: No mask (0)

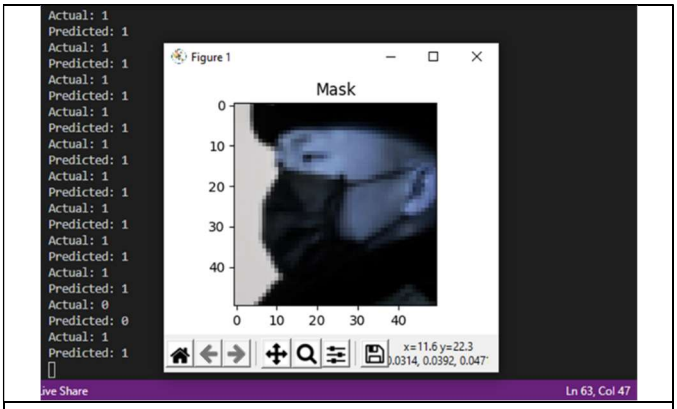

Figure 7: Mask (1)

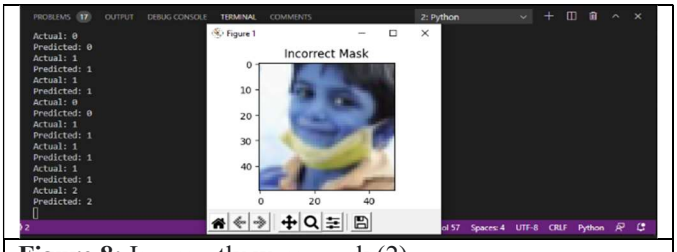

Figure 8: Incorrectly worn mask (2)

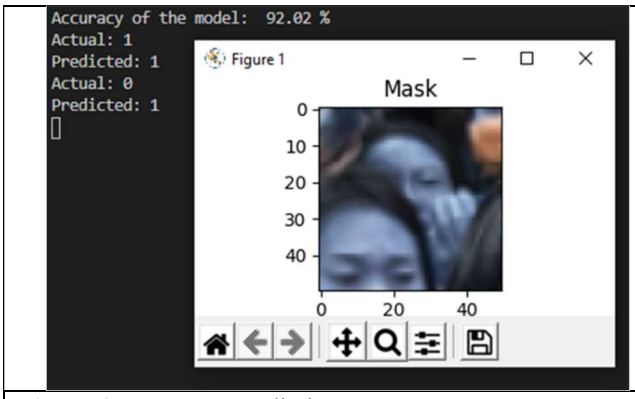

Figure 9: Incorrect prediction

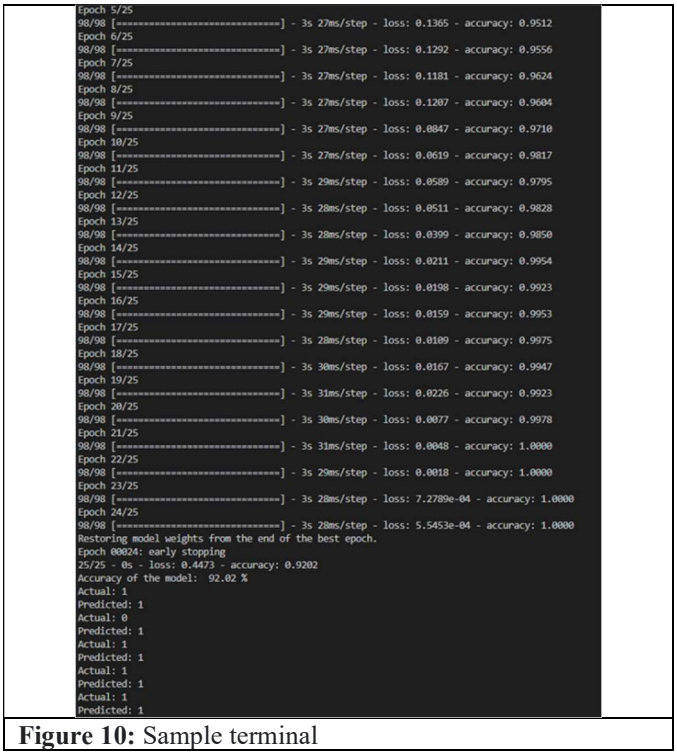

Figure 10: Sample terminal

IV. Conclusion

COVID-19 is currently the greatest health concern in the world. Therefore, all preventive measures should be taken until it is under control and every single individual in the world is vaccinated. According to the institutions around the world, the best precautionary measure is to wear face masks. Many governments around the world made it a law that individuals must wear masks when they enter public spaces. This project allows for the classification of individuals found within images into three categories: no mask, mask, incorrect mask. This tool allows for the classification of these individuals and gives institutions the ability to classify very quickly and easily from images taken by a camera. This method uses CNN to analyze images and classifies them under the three categories mentioned above with an accuracy of ~91%, providing a great method for the differentiation and classification with high

accuracy. Methods that involve image classification are invaluable for the future of police and military work because of its ability to very quickly be able to analyze a huge number of individuals and conclude with high accuracy. CNN's are widely used in image processing as well as other machine learning applications.

V. CONTRIBUTION OF EACH MEMBER

Riken and Jakub worked on the code together. Jakub wrote the abstract, introduction, and conclusion of the paper. Riken wrote the method description and experimental results. We both worked on the presentation and video together focusing on similar components as the paper. We helped each other out to complete the project, 50:50 work split as a group.

VI. REFERENCES

[1] Covid19.who.int. 2021. *WHO Coronavirus (COVID-19) Dashboard*. [online] Available at: <https://covid19.who.int/> [Accessed 27 April 2021].

[2] Kaggle.com. 2021. *Face Mask Detection*. [online] Available at: <https://www.kaggle.com/andrewmvd/face-mask-detection> [Accessed 27 April 2021]..

[3] Abouzeid, Y., 2021. *Pytorch - FasterRCNN*. [online] Kaggle.com. Available at: <https://www.kaggle.com/yehiakhaledabouzeid/pytorch-fasterrcnn> [Accessed 27 April 2021].

[4] A. Cetin, Class Lecture, Topic: "CNN" ECE 491, College of Engineering, University of Illinois at Chicago, Apr.,27, 2021.

[5] Dickson, B., 2021. What are convolutional neural networks (CNN)?. [online] TechTalks. Available at:<https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/> [Accessed 27 April 2021].

[6] Chollet, F., 2021. Keras documentation: The Sequential model. [online] Keras.io. Available at: <https://keras.io/guides/sequential_model/> [Accessed 27 April 2021]. https://keras.io/guides/sequential_model/

[7] Brownlee, J., 2021. A Gentle Introduction to Tensors for Machine Learning with NumPy. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/introduction-to-tensors-for-machine-learning/#:~:text=Tensors%20are%20a%20type%20of,them%20in%20Python%20with%20NumPy> [Accessed 27 April 2021].

[8] Team, K., 2021. *Keras documentation: Conv2D layer*. [online] Keras.io. Available at: <https://keras.io/api/layers/convolution_layers/convolution2d/> [Accessed 27 April 2021].

[9] Sharma, S., 2021. Activation Functions in Neural Networks. [online] Medium. Available at: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> [Accessed 27 April 2021].

[10] Team, K., 2021. *Keras documentation: MaxPooling2D layer*. [online] Keras.io. Available at: <https://keras.io/api/layers/pooling_layers/max_pooling2d/> [Accessed 27 April 2021].

[11] Tutorialspoint.com. 2021. Keras - Dense Layer - Tutorialspoint. [online] Available at: <https://www.tutorialspoint.com/keras/keras_dense_layer.htm> [Accessed 27 April 2021].

[12] TensorFlow. 2021. tf.keras.losses.SparseCategoricalCrossentropy | TensorFlow Core v2.4.1. [online] Available at: <https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy> [Accessed 27 April 2021].

[13] Saha, S., 2021. A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. [online] Medium. Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> [Accessed 27 April 2021].

VII. Appendix: code list

```
'''
Written by Riken Patel & Jakub Marek
Date: 4/24/2021
ECE 491 project pre-processing data
'''
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup
import torchvision
from torchvision import transforms, datasets, models
import torch
import os
import cv2
import warnings

warnings.filterwarnings("ignore") #ignores the warnings

## Take the objects in the image and labels them
def generate_label(obj):
    if obj.find('name').text == "with_mask":
        return 1
    elif obj.find('name').text == "mask_weared_incorrect":
        return 2
    return 0 #No mask

## Gathers the box coordinate data from the file as variables
def generate_box(obj):
    xmin = int(obj.find('xmin').text)
    ymin = int(obj.find('ymin').text)
    xmax = int(obj.find('xmax').text)
    ymax = int(obj.find('ymax').text)
    return [xmin, ymin, xmax, ymax]

## Create a new matrix that houses box, labels, and image id information
def generate_target(image_id, file):
    with open(file) as f:
        data = f.read()
        soup = BeautifulSoup(data,features= 'lxml')
        objects = soup.find_all('object')
        boxes = []
        labels = []
        for i in objects:
            boxes.append(generate_box(i))
            labels.append(generate_label(i))
        boxes = torch.as_tensor(boxes, dtype=torch.float32)
        labels = torch.as_tensor(labels, dtype=torch.int64)
        img_id = torch.tensor([image_id])
        target = {}
        target["boxes"] = boxes
        target["labels"] = labels
        target["image_id"] = img_id
        return target

img_dir = "c:/Users/Owner/Documents/UIC/ECE 491/Project/images"
ano_dir = "c:/Users/Owner/Documents/UIC/ECE 491/Project/annotations"
save_dir = "c:/Users/Owner/Documents/UIC/ECE 491/Project/box_images"
```

```
data_mask = []
num_imgs = len(os.listdir(img_dir))
for i in range(num_imgs):
    image_file = 'maksssksksss'+ str(i) + '.png'
    label_file = 'maksssksksss'+ str(i) + '.xml'
    img_path = os.path.join(img_dir,image_file)
    label_path = os.path.join(ano_dir,label_file)
    target = generate_target(i, label_path)
    data_mask.append(target)

images = list(sorted(os.listdir(img_dir)))
annotations = list(sorted(os.listdir(ano_dir)))

## Create new images form the boxes found in the original images, save then in a new folder
def image_class(i, data_mask):
    image_file = 'maksssksksss'+ str(i) + '.png'
    img_path = os.path.join(img_dir,image_file)
    image = cv2.imread(img_path)
    temp_img = image
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) #original image
    for j in range(len(data_mask[i]["boxes"])):
        box = data_mask[i]['boxes'][j]
        xmin, ymin, xmax, ymax = box
        cv2.rectangle(temp_img, (int(xmin-10), int(ymin-10)), (int(xmax+10), int(ymax+10)), (255,255,255) ,2)
        temp_img = cv2.cvtColor(temp_img, cv2.COLOR_BGR2RGB) #image with the box
        img_box = image[int(ymin-10):int(ymax+10), int(xmin-10):int(xmax+10)]
        label = int(data_mask[i]['labels'][j])
        label_path = 'img_{}_{}_{}.png'.format(i,j,label) #img, box num, label
        try:
            save_path = os.path.join(save_dir,label_path) #.../box_images/img_0_0_0.png
            img_box = cv2.resize(img_box,(50,50),interpolation = cv2.INTER_AREA)
            img_box = cv2.cvtColor(img_box, cv2.COLOR_BGR2RGB) #original image
            cv2.imwrite(save_path,img_box) #saves the image into the location
        except:
            pass

def preprocessing_images(img_dir = img_dir,data_mask = data_mask):
    num_imgs = len(os.listdir(img_dir))
    for i in range(num_imgs):
        image_class(i,data_mask)

preprocessing_images(img_dir,data_mask)


'''
Written by Riken Patel & Jakub Marek
Date: 04/24/2021
ECE 491 project
'''

import numpy as np
import os
import pandas as pd
import importlib
import warnings
from sklearn.model_selection import train_test_split
```

```
import cv2
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import keras
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.callbacks import EarlyStopping
from sklearn.metrics import accuracy_score
import random

warnings.filterwarnings("ignore") #ignores the warnings

save_dir = "c:/Users/Owner/Documents/UIC/ECE 491/Project/box_images"

def create_dict(save_dir):
    images_data = []
    labels = []
    for i in range(len(os.listdir(save_dir))):
        img = cv2.imread(os.path.join(save_dir,os.listdir(save_dir)[i])) #reading in the image
        img = img/255.0 #normalizing
        images_data.append(img)
        #extracting the labels
        label = os.listdir(save_dir)[i]
        label = label.split('_')[3]
        label = label.split('.')[0]
        labels.append(int(label)) #appending to the array
    return images_data,labels
img_data,labels = create_dict(save_dir)

X_train,X_test,y_train,y_test = train_test_split(np.array(img_data),np.array(labels),test_size=0.20, random_state=42)

model = Sequential()
model.add(Conv2D(32,(3,3), activation='relu', input_shape=(50,50,3)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dense(10))
print(model.summary())
model.compile(optimizer='adam',
          loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
          metrics=['accuracy'])
monitor = EarlyStopping(monitor='accuracy',patience = 3 , verbose=1, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=25, callbacks=[monitor])
loss,acc = model.evaluate(X_test,y_test,verbose = 2)
print('Accuracy of the model: ',round(acc,4)*100,'%')

random_array = np.random.randint(len(y_test), size=(1,5))[0]
for i in random_array:
    print('Actual:',y_test[i])
    test_img =  X_test[i]
```

```
expanded_image = np.expand_dims(test_img, axis=0)
prediction = model.predict(expanded_image)
print('Predicted:',np.argmax(prediction))
if np.argmax(prediction) == 1:
    plt.imshow(X_test[i])
    plt.title('Mask')
    plt.show()
elif np.argmax(prediction) == 0:
    plt.imshow(X_test[i])
    plt.title('No Mask')
    plt.show()
else: #2
    plt.imshow(X_test[i])
    plt.title('Incorrect Mask')
    plt.show()
```