الجامعة السعودية الالكترونية
Saudi Electronic University
2011 1432
College of Computing and Informatics

2020/2021 Second Semester

Course Code	DS520
Course Name	Artificial Intelligence for Data Science
CRN	24543
Assignment type	Project
Module	All
Assignment Points	10

Student ID	G200002614	G200007615
Student Name	Enad S. Alotaibi	Abdulaziz M. Alqumayzi

1. Introduction:

In this project, we will be developing a face mask detector that can distinguish between faces with masks and faces with no masks. In this report, we have proposed a detector that employs SSD for face detection and a neural network to detect the presence of a face mask. The implementation of the algorithm is on images, videos, and live video streams.

2. Literature Review

Object detection is one of the trending topics in the field of image processing and computer vision. Ranging from small-scale personal applications to large-scale industrial applications, object detection, and recognition is employed in a wide range of industries. Some examples include image retrieval, security and intelligence, OCR, medical imaging, and agricultural monitoring. In object detection, an image is read and one or more objects in that image are categorized. The location of those objects is also specified by a boundary called the bounding box. Traditionally, researchers used pattern recognition to predict faces based on prior face models. A breakthrough face detection technology then was developed named as Viola-Jones detector that was an optimized technique of using Haar, digital image features used in object recognition. However, it failed because it did not perform well on faces in dark areas and non-frontal faces. Since then, researchers are eager to develop new algorithms based on deep learning to improve the models. Deep learning allows us to learn features in an end-to-end manner and removing the need to use prior knowledge for forming feature extractors. There are various methods of object detection based on deep learning which are divided into two categories: one-stage and two-stage object detectors. Two-stage detectors use two neural networks to detect objects, for instance, region-based convolutional neural networks (R-CNN) and Faster R-CNN. The first neural network is used to generate

region proposals and the second one refines these region proposals; performing a coarse-to-fine detection. This strategy results in high detection performance compromising on speed. The seminal work R-CNN is proposed by R. Girshick etal. R-CNN uses selective search to propose some candidate regions which may contain objects. After that, the proposals are fed into a CNN model to extract features, and a support vector machine (SVM) is used to recognize classes of objects. However, the second stage of R-CNN is computationally expensive since the network has to detect proposals in a one-by-one manner and uses a separate SVM for final classification. Fast R-CNN solves this problem by introducing a region of interest (ROI) pooling layer to input all proposal regions at once. Faster RCNN is the evolution of R-CNN and Fast R-CNN, and as the name implies its training and testing speed is greater than those of its predecessors. While R-CNN and fast R-CNN use selective search algorithms limiting the detection speed, Faster R-CNN learns the proposed object regions itself using a region proposal network (RPN).

Like object detection, face detection adopts the same architectures as one-stage and two-stage detectors, but to improve face detection accuracy, more face-like features are being added. However, there is occasional research focusing on face mask detection. Some already existing facemask detectors have been modeled using OpenCV, Pytorch Lightning, MobileNet, RetinaNet, and Support Vector Machines. Here, we will be discussing two projects. One project used Real WorldMasked Face Dataset (RMFD) which contains 5,000 masked faces of 525 people and 90,000 normal faces [8]. These images are 250 x 250 in dimensions and cover all races and ethnicities and are unbalanced. This project took 100 x 100 images as input, and therefore, transformed each sample image when querying it, by resizing it to 100x100. Moreover, this project uses PyTorch then they convert images to Tensors, which is the base data type that PyTorch can work with. RMFD is im-balanced

(5,000 masked faces vs 90,000 non-masked faces). Therefore, the ratio of the samples in train/validation while splitting the dataset was kept equal using the train test split function of sklearn. Moreover, to deal with unbalanced data, they passed this information to the loss function to avoid unproportioned step sizes of the optimizer. They did this by assigning a weight to each class, according to its representability in the dataset. They assigned more weight to classes with a small number of samples so that the network will be penalized more if it makes mistakes predicting the label of these classes. While classes with large numbers of samples, they assigned to them a smaller weight. This makes their network training agnostic to the proportion of classes. The weights for each class were chosen using the formula below:

$$\textit{Class Weight} = 1 - (\ \textit{Class Cardinality} / \textit{Cardinalities of all classes}\ )$$

3. Body Section:

a. **Methods:** The working of the Single Shot Detector algorithm relies on an input image with a specified bound-ing box against the objects. The methodology of predicting an object in an image depends upon a very renowned convolution fashion. For each pixel of a given image, a set of default bounding boxes (usually 4) with different sizes and aspect ratios are evaluated. Moreover, for all the pixels, aconfidence score for all possible objects is calculated with an additional label of 'No Object'. This calculation is repeated for many different feature maps. To extract feature maps, we usually use the predefined trained techniques which are used for high-quality classification problems. We call this part of the model a base model. For the SSD, we have the VGG-16 network as our base model. At the training time, the bounding boxes were evaluated compared with the

ground truth boxes, and in the backpropagation, the trainable parameters are altered as per requirement. We truncate the VGG-16 model just before the classification layer and add feature layers that keep on decreasing in size. At each feature space, we use a kernel to produce outcomes that depict corresponding scores for each pixel whether there exists any objector not, and the corresponding dimensions of the resulting bounding box.VGG-16 is a very dense network having 16 layers of convolution which are useful in extracting features to classify and detect objects. The reason for the selection is because the architecture 4 consists of stacks of convolutions with 3x3 kernel size which thoroughly extract numerous feature information along with max-pooling and ReLU to pass the information flow in the model and adding linearity respectively from the given image. For additional nonlinearity, it uses 1x1 convolution blocks which does not change the spatial dimension of the input. Due to the small size filters striding over the image, many weight parameters end up giving improved performance. The block diagram [10] shows the working functionality of SSD. At the input end, we can see the-16 being used as the base model. Some additional feature layers are added at the end of the base model to take care of offsets and confidence scores of different bounding boxes. At the end part of the figure, we can see the layers being flattened to make predictions for different bounding boxes. In the end, non-maximum suppression is used whose purpose is to remove duplicate or quite similar bounding boxes around the same objects. There may be situations where the neighboring pixel also predicts a bounding box for an object with a bit less confidence which is finally rejected.

b. **Training:** Training At the training time, for each pixel, we compare the default bounding boxes having different sizes and aspect ratios with ground truth boxes and finally use the Intersection over Union (IoU) method to select the best matching box.

IoU evaluates how much part of our predicted box matches with the ground reality. The values range from 0 to 1 and increasing values of IoU determine the accuracies in the prediction; the best value being the highest value of IoU. The equation and pictorial description of IoU are given as follow:

IoU (B1, B2) = B1∩B2 / B1∪B2

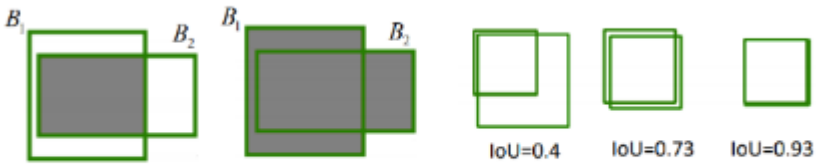


Figure 4: Pictorial representation of IoU.

c. **Loss functions:** Loss functions loss of the overall detection problem can be broken into two main subsets: localization loss andconfidence loss. The localization loss is just the difference between the default predicted bounding box and the ground truth bounding box (g). For a given center (cx, cy), we try to alter the width and height of the box such as to decrease the loss. Respectively, the equations for the localization andconfidence losses can be defined as:

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log(\frac{g_j^w}{d_i^w}) \qquad \hat{g}_j^h = \log(\frac{g_j^h}{d_i^h})$$

and

$$L_{locf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

The notations used in the equations are as follows:

- *g: ground truth bounding box*
- *l: predicted box*
- *d: default bounding box*
- *xpij : matching ith predicted box with jth default box with "p" category.*
- *cx, cy: distance from the center of the box in both x and y-direction*
- *w, h = width, and height concerning image size*
- *c: confidence of the presence of object or not.*

Confidence loss is the just measure of how high the probability of the presence of an object is when there exists an object. Similarly, the localization loss is the measure of how much a predicted box differs from the ground truth box in dimensions. Our model will try to push down both losses by predicting the presence of an object and then correctly classifying it to the right class.

d. DataSets: The dataset which we have used consists of 1509 total images out of which 755 are of masked faces and 754 are of unmasked faces. The masked faces images are augmented images of masks of all the surgical masks.

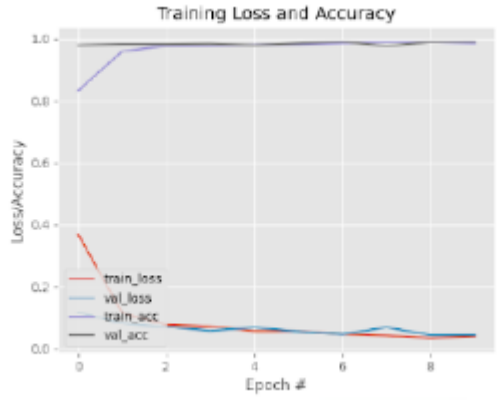
We need to split our dataset into three parts: training dataset, test dataset, and validation dataset. The purpose of splitting data is to avoid overfitting which is paying attention to minor details/noise which is not necessary and only optimizes the training dataset accuracy. We need a model that performs well on a dataset that it has never seen (test data), which is called generalization. The training set is the actual subset of the dataset that we use to train the model. The model observes and learns from this data and then optimizes its parameters. The validation dataset is used to select hyperparameters

(learning rate, regularization parameters). When the model is performing well enough on our validation dataset, we can stop learning using a training dataset. The test set is the remaining subset of data used to provide an unbiased evaluation of a final model fit on the training dataset. Data is split as per a split ratio which is highly dependent on the type of model we are building and the dataset itself. If our dataset and model are such that a lot of training is required, then we use a larger chunk of the data just for training which is our case. If the model has a lot of hyperparameters that can be tuned, then we need to take a higher amount of validation dataset. Models with a smaller number of hyperparameters are easy to tune and update, and so we can take a smaller validation dataset. In our approach, we have dedicated 80% of the dataset as the training data and the remaining 20%as the testing data, which makes the split ratio 0.8:0.2 of the train to test set. Out of the training data, we have used 20% as a validation data set. Overall, 64% of the dataset is used for training, 16% for validation, and 20% for testing.

e. **Result or Analysis:** We tried using three different base models for detecting ‘mask’ or ‘no mask’. The exercise was done to find the best fit model in our scenario. The evaluation process consists of first looking at theclassification report which gives us insight into precision, recall, and F1 score. The equations of these three metrics are as follows:

- Precision = True Positives / Positives + False Positives
- Recall = True Positives / Positives + False Negatives
- Accuracy =True Positives +True Negatives Positives + Negatives

Using these three metrics, we can conclude which model is performing most efficiently. The second part consists of plotting the training loss, validation loss, train accuracy, and validation accuracy which also proves helpful in choosing a final model.



f. Limitations: There were not many challenges faced but the two problems that were time-consuming and made the tasks tedious are discussed as follows. One was the excessive data loading time in Google ColabNotebook while loading the dataset into it. Since the runtime restarting refreshes all the cells, the cell for dataset loading took most of the time while running. Secondly, the access problem in GoogleColab Notebook: it did not allow the access of webcam which posed a hurdle in testing images and live video stream through Google Colab Notebook. Therefore, we had to run the code locally on the computer through which we tested the code on the live video stream

4. Conclusion:

To mitigate the spread of the COVID-19 pandemic, measures must be taken. We have modelled a facemask detector using SSD architecture and transfer learning methods in neural networks. To train, validate and test the model, we used the dataset that consisted of 1916 masked faces images and1919 unmasked faces images. These images were taken from various resources like Kaggle and RMFD datasets. The model was inferred on images and live video streams. To select a base model, we evaluated the metrics like accuracy, precision, and recall and selected MobileNetV2 architecture with the best performance having 100% precision and 99% recall. It is also computationally efficientusing MobileNetV2 that makes it easier to install the model to embedded systems. This face mask detector can be deployed in many areas like shopping malls, airports, and other heavy traffic places to monitor the public and to avoid the spread of the disease by checking who is following basic rules and who is not.

References:

[1] D. Ciresan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: Proceedings of the CVPR, 2012.

[2] K. He, J. Sun, Convolutional neural networks at constrained time cost, in: Proceedings of the CVPR, 2015.

[3] Y.L. Boureau, J. Ponce, Y. LeCun, A theoretical analysis of feature pooling in visual recognition, in: Proceedings of the ICML, 2010.

[4] B. QIN, D. LI, Identifying facemask-wearing condition using image super-resolution with classi_cation network to prevent covid-19 (2020).

[5] N.C. Ristea, R.T. Ionescu, Are you wearing a mask? improving mask detection from speech using aug- mentation by cycle-consistent gans, arXiv preprint arXiv:2006.10147 (2020).

[6] D. Matthias, M. Chidozie, Face mask detection applica- tion and dataset