Mask Recognition with Computer Vision in the Age of a Pandemic

Daniel Jones, Christoforos Christoforou
Division of Computer Science, Mathematics and Science, St. John's University, NY, U.S.A.
daniel.jones13@my.stjohns.edu, christoc@stjohns.edu

Abstract

In the current age of coronavirus, monitoring and enforcing correct mask-wearing regulation in public spaces is of paramount importance. Specifically, there is a need to monitor whether people wear masks and whether they wear them correctly. However, there is a lack of automated systems to recognize correct mask-wearing compliance. In this paper, we propose a computer-vision-based solution to the problem of mask-wearing monitoring. In particular, we propose a convolutional neural network to recognize images of people wearing masks correctly, people wearing masks incorrectly, and people not wearing masks at all. Our proposed model is shown to predict correct mask-wearing practices with over 98% accuracy. The model can be easily deployed as an automated system to screen people entering indoor spaces, and can replace current manual, time-consuming, temperature-screening practices. Such applications can serve as an important tool to help reduce transmission rates during the current pandemic.

Introduction

Because of the 2020 pandemic, throughout the US and around the world, mask-wearing is required in order to enter stores, restaurants, trains, places of work, and many other public venues. The CDC recommends that all people over the age of two wear masks in public spaces ("COVID-19: Considerations", 2020[1]). According to WebMD (Ellis 2020), the risk of transmitting COVID-19 decreases by 65% when wearing a mask, and thus mask-wearing is considered one of the most effective tools to reduce the spread of COVID-19. Because of the significant impact of mask-wearing in reducing transmission, incompliance with these requirements comes with high fines for the individuals violating the regulations and with store-closures for the venue owners. Yet, many people, either intentionally or unintentionally, fail to comply with the regulations and enter stores without wearing a mask correctly or wearing a mask at all. Thus, monitoring and enforcing mask-wearing regulations is of paramount importance that has significant financial and global health implications. However, it is challenging to monitor the thousands of people entering a venue for compliance.

Currently, businesses have human-workers stationed outside the stores or venues to remind people to wear a mask, or to fix their mask so that it sits on a person's face correctly. However, this manual approach is costly, time-consuming, prone to human error and results in long waiting lines at the venue. For example, hiring and training human workers to monitor mask-wearing is both time consuming and costly for businesses. Moreover, given the high volume of people entering a venue at any given time, the monitoring process is typically slow, resulting in long waiting lines. Importantly, the process runs the risk of human error, resulting to some individuals "fall-through-the-cracks", entering the venue without a mask and potentially spreading the virus. Therefore, it is important to look into automated systems for monitoring and enforcing mask-wearing regulation compliance.

Machine Learning approaches have been used in several application domains ranging for computer vision to natural-language processing to neuroscience (Christoforou et al., 2013; Christoforou et al. 2018) and neuro-technologies (Christoforou et al. 2010). Particularly in the Computer Vision domain, machine learning-based approaches have been proposed to automate several object recognitions tasks. For example, in (Fang et al. 2018), the authors aim to use object recognition to detect construction workers that are not wearing hardhats in order to reduce workplace accidents. Similarly, in (Mneymneh et al. 2019), the authors use object recognition in order to identify humans on construction sites in video, and then to determine whether or not the humans are wearing hardhats. Their goal is to replace the tedious safety monitoring process.

Besides safety on worksites, object recognition has also proven to be useful for public safety. In (Verma et al. 2017),

[1] https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/cloth-face-cover-guidance.html

the author's show that object recognition can be used for automated gun detection, possibly leading to applications that can help reduce crime. In a similar fashion, in (Gelana et al. 2019), the authors state that gun detection is necessary in order to identify "active shooters," and that human monitoring cannot keep up with the amount of CCTV footage. Therefore, they demonstrate how guns can be detected using CV object recognition techniques. Public and worksite safety are just a few of many examples in which object recognition can be utilized in the real world.

Modern CV solutions rely on deep-convolutional neural network models to achieve high-performance recognition. For example, deep-neural networks have been proposed to identify plant and animal species recognition. In (Lasseck 2017), the authors are able to use deep-convolutional neural networks in order to predict ten thousand plant species with ninety-six percent accuracy. Furthermore, in (Nguyen et al. 2016), the authors use different deep neural networks to classify images of 967 flower species and achieve results of up to 90.82% accuracy. In fact, a couple of CV- based solutions towards mask-wearing monitoring have been proposed in the literature which we review in the next couple of paragraphs.

One approach towards automated mask-wearing monitoring, was proposed in (Chavda et al. 2020). Their approach involved a multi-stage Convolutional Neural Network (CNN) architecture to detect face-masks in images. In particular, they first used a pre-trained model to detect faces in an image. Next, they designed a convolutional neural network to predict if each face that is detected contains a mask or not. Their models report just over 99% accuracy in identifying faces with masks. However, their model does not differentiate whether a mask is worn correctly or incorrectly, which is an important factor in the effectiveness of mask-wearing. We decide to skip the multi-stage approach in our research, as face detection is already abundantly discussed in literature.

A second approach was proposed in (Hammoudi et al. 2020), which attempts to differentiate between masks being worn correctly or incorrectly. There, the authors proposed a method that exploits Haar-like features to detect faces in an image as well as key features of the face (namely, eyes, mouth, nose and chin) from a camera-based acquisition of a mobile phone. They then used the presence or absence of those facial features to infer whether a person wears a mask correctly or incorrectly. If all of these features are detected, the application alerts that the user is wearing their mask incorrectly, otherwise it notifies the user is wearing the mask correctly. There are several limitations associated with the proposed approach. First, it uses an ad-hoc decision-rule (i.e. presence or absence of facial features) that relies on the accuracy of the face-feature detection algorithm; failure to detect facial features results to a decision that the mask is worn correctly. Moreover, the proposed method is designed to take as input "selfie" image from mobile-phones and cannot be generalized to venue-entrance monitoring environments. Importantly, the performance of the proposed algorithm is not validated on any real dataset, nor any quantification of the algorithm's accuracy is reported on the paper.

In an effort to differentiate between correctly vs incorrectly worn face masks, in this paper we propose a fully convolutional neural network architecture. The model performance is validated on a large dataset and is shown to achieve high accuracy scores. Our results could lead to applications that could hopefully limit the spread of COVID-19.

The rest of the paper is organized as in the following sections. In the methods section we introduce our proposed model, discuss the dataset used to validate our approaches and the procedure to evaluate our method. Next, we discuss some of the implementation details and report results on the assessment of our method. Finally, we conclude with the significance of our methods and discuss future work.

Methods

Face-mask Dataset creation and pre-processing

In order to validate our proposed approach, the first step was to access many images of people wearing masks correctly, images of people wearing masks incorrectly, and images of people not wearing masks at all. In order to do this, we collected data from the MaskedFace-Net dataset, which stems from the paper *MaskedFace-Net – A Dataset of Correctly/Incorrectly Masked Face Images in the Context of COVID-19* (Cabani et al. 2020). This dataset comprises of 137,016 images of people wearing masks. Each image is 1024x1024 pixels, RGB, and in jpg format. These images come from the Flickr-Faces-HQ dataset (Karras et al. 2019), but are edited with masks pasted over the faces either correctly or incorrectly. About half of the MaskedFace-Net dataset contains images of people wearing masks correctly, while the other half contains images of people wearing masks incorrectly. The incorrect mask wearing portion of the dataset consists of different ways to incorrectly wear masks. For example, in some of the images the person is not covering his nose. In other images, the person may not be covering his mouth or nose, with his mask around the chin. To generate the dataset, we gathered the first 5,000 images from the correctly worn mask portion of the dataset, and then gathered the next 5,000 images from the incorrectly worn mask dataset. Finally, to obtain the no mask portion of the dataset, we sampled 5,000 of the images from the original Flickr-Faces-HQ dataset. Overall, the dataset comprised 15,000 images from the three categories.

To prepare the images as an input to our model, we perform the following pre-processing steps. First each image was normalized (i.e. in the range 0 to 1) and rescaled to

224x224 pixels. The dataset was then split into separate folders; one folder included 70% of the images to be used for training, and 30% of the images to be used for model validation. To facilitate dynamic image loading during model training, the image data was stored in labeled folders. Specifically, two folders were created to store training and validation images. Within each of these folders, three subfolders were created, each storing images for the three categories (i.e. unmasked images; masked images worn correctly; masked images worn incorrectly).

Figure 1: Example Images from Dataset

Model Architecture

To recognize images into the three classes, we designed a convolutional neural network. The model is organized as follows. The first block of the model consisted of a convolution layer with thirty-two kernels of size five by five. These kernels operate at strides of four by four in order increase the speed of training. Padding is also included to return an image of the same dimensions. Next, a batch normalization layer is included to normalize the convolution and reduce the chance of overfitting. This convolution layer is then activated using a relu activation function. A two by two max pooling layer is then used to downsize the feature maps. Finally, a dropout layer set at .5 is included in order to prevent overfitting.

After this first convolution block, the model next contains a flatten layer, followed by a dense block. The dense layer in the block has 32 neurons and is activated by a relu function. Again, overfitting is prevented by utilizing batch normalization and a dropout layer is set at .2 in this block. Finally, the model ends with a dense layer activated by a softmax activation function in order for the model to appropriately make predictions on the three classes: correct mask, incorrect mask, or no mask. The model architecture is outlined figure 2.

Model Training, regularization and evaluation

To train the model architecture described above , we utilized a categorical cross entropy as a loss function and the adam optimizer as implemented in tensorflow. The performance metrics was set to accuracy score; the batch size was set to 32 observations and the maximum epochs was set to ten, with early stopping callback on the validation score set to 0.001 and patience of 5 epochs. Checkpoint callbacks were employed to monitor the best performing model weights.

Image augmentation was employed as a regularization mechanism during training. In particular, input data images were augmented by random transformations including (zooming, horizontally flipping, and changing shear). The ImageDataGenerator module from Tensorflow library was used to provide the augmented training dataset during training. Data and validation batches were generated dynamically during the training process using the 'from_directory' method of the ImageDataGenerator object. To assess our proposed model, we report validation accuracy during training of the model. Moreover, we report on the training speed of the model.

Results

Training Time Metrics

The training went on for eight out of the ten epochs, so early stopping was utilized during training. Epochs had to go

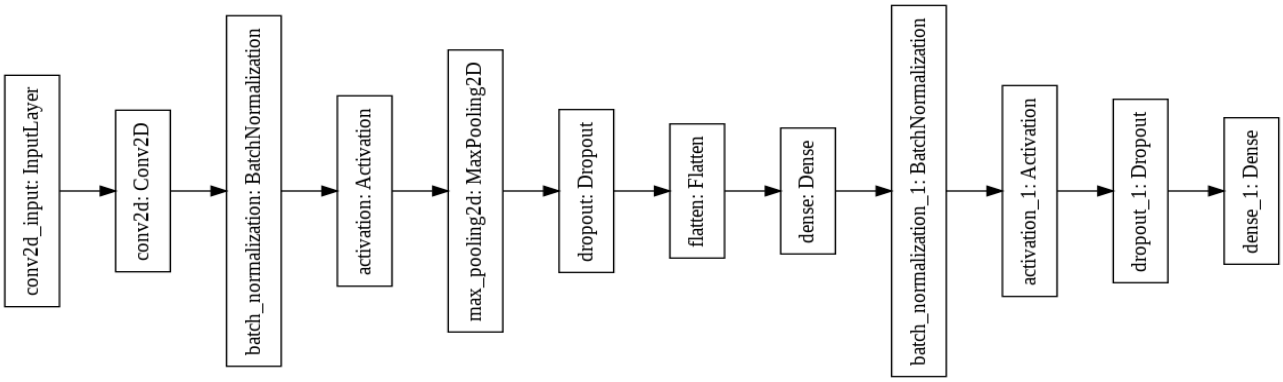

Figure 2: Model Architecture.

through 323 steps due to a batch size of 32, and each step ended up taking around two seconds for the model to fit the batch and update weights. Each epoch took a total of about six hundred seconds, so the training process took around a total of an hour and a half.

Training Process

Training loss and accuracy were monitored and displayed throughout the training process. Training loss steadily decreased as the training went through each epoch. It is also notable that training accuracy went up as the epochs ran. Validation accuracy and loss were also monitored and displayed during training. Validation accuracy started out high after the first epoch at 94%, and the validation loss started out low at .25. At the eighth and final epoch, the validation accuracy was at its highest point of 98.4% with a loss of .069. We save the weights from this final epoch, as it achieved the highest validation accuracy score. The loss and accuracy curves during training can be visualized in figure 3 below.

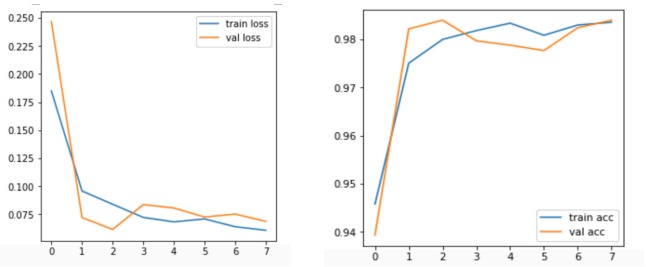

Figure 3: Training and Validation loss (left) and accuracy (right)

It can be seen from figure 3 that validation accuracy and validation loss match training accuracy and training loss very closely. This means that the model was successful at preventing overfitting, as the model does well in predicting both training data and validation data. This can be attributed to including image augmentation, bath normalization layers, max pooling layers, and dropout layers. Our model of choice that is saved uses the weights that perform the best on predicting validation data. This means that we use the model weights from the eighth epoch, which was able to predict the validation images at 98.4% accuracy. This is the model that is necessary to evaluate and see where it failed.

Evaluation

Using the prediction generator method from tensorflow, we were able to evaluate the completed model on the validation set and save predictions for each image on the validation set. The final model was able to predict the validation classes with 98.5% accuracy. This means that the model classified images of people wearing masks correctly, incorrectly, or not at all with only 1.5% error. Obviously, the predictions were not perfect, but the model predicted the classes well. However, it was still necessary to investigate as to why the model could not classify the images with one hundred percent accuracy.

In order to do this, I created a pandas dataframe to store the filename, along with the actual labels of the image and the predicted labels of the image. The first part of the dataframe is shown below.

	Filename	Predictions	Actual
0	MaskCorrect/00000_Mask.jpg	MaskCorrect	MaskCorrect
1	MaskCorrect/00001_Mask.jpg	MaskCorrect	MaskCorrect
2	MaskCorrect/00006_Mask.jpg	MaskCorrect	MaskCorrect
3	MaskCorrect/00008_Mask.jpg	MaskCorrect	MaskCorrect
4	MaskCorrect/00013_Mask.jpg	MaskCorrect	MaskCorrect
5	MaskCorrect/00016_Mask.jpg	MaskCorrect	MaskCorrect
6	MaskCorrect/00018_Mask.jpg	MaskCorrect	MaskCorrect
7	MaskCorrect/00021_Mask.jpg	MaskCorrect	MaskCorrect
8	MaskCorrect/00022_Mask.jpg	MaskCorrect	MaskCorrect
9	MaskCorrect/00023_Mask.jpg	MaskCorrect	MaskCorrect
10	MaskCorrect/00025_Mask.jpg	MaskCorrect	MaskCorrect

Figure 4: Pandas Dataframe Storage

We can use this dataframe to discover where our model incorrectly predicted the class of an image by getting the filenames where the predictions column does not match the actual column. This showed me that the model only misclassified 65 out of 4,434 images, confirming the high accuracy of the model. Using this dataframe, we can also create a confusion matrix to see how the model performed relative to each class. The confusion matrix can be seen in figure 5.

Actual \ Predicted	Mask Correct	Mask Incorrect	Mask None
Mask Correct	1432	6	0
Mask Incorrect	18	1448	30
Mask None	0	11	1489

Figure 5: Confusion Matrix (4,434 validation images)

From the confusion matrix, we can determine that the model correctly predicted that people were wearing masks the right way 99.6% of the time, that people were wearing masks the wrong way 96.8% of the time, and that people were not wearing masks at all 99.3% of the time.

The next thing to do was to load some of the images to see what went wrong with these examples. We visually inspected the image examples where the classifier made wrong predictions, and three representative examples are shown of the error types are shown figure 6.

(a) (b) (c)

Figure 6: Visual inspection of incorrectly predicted images

For the input image in figure 6.a, the model predicted that the baby was wearing his mask correctly. However, according to the dataset, he was wearing his mask incorrectly. This obviously could be up for debate, as his nose, mouth, and chin all seem to be covered up by his mask. It's possible that the image could have been mislabeled, and that the model actually predicted the image correctly as someone wearing a mask correctly. The error by the model stems from this image being a difficult example.

In the second image example in figure 6.b we observe similar situation. The model predicted that the baby displayed in figure 6.b was wearing her mask incorrectly, but the image is labeled as not wearing a mask at all. This is another image that is a difficult example, as the baby has a binky over her face, that the model most likely mistakenly classifies as an incorrect wearing of a mask.

When the image in in figure 6.c is used as an input, the model predicts this person is incorrectly wearing a mask, while the image is labeled as correct wearing of a mask. This is another difficult example, as the person's nose and mouth are not covered at all. This example is clearly an error within the dataset and the model was correct when predicting that this person was not wearing her mask correctly.

After examining these images, we came to the conclusion that most of the incorrect predictions came from difficult examples that could just be errors within the dataset. Therefore, we did not find it necessary to go back and change the model any further, as we were content with the high accuracy rate of 98.5%.

Discussion

Overall, the model did a great job in predicting if people are wearing masks correctly, incorrectly, or not wearing masks at all. It was able to predict the validation set with 98.5% accuracy. The examples that were mis-classified were difficult examples that could have potentially been mislabeled. There have been implementations on classifying incorrect and correct wearing of masks, and wearing or not wearing masks, but I have not seen any implementations on all three categories.

The extensions to this kind of classification are endless. People could be monitored while entering places of work to make sure they are wearing masks properly, and they could also be monitored at work if this is applied to video. Hopefully, an application such as this could have tremendous impacts on society and help slow down the pandemic, which is now picking up faster than ever before. Computer vision can also be useful for the pandemic in other ways, such as social distancing violations, occupancy counting, and even detecting COVID-19 patients through chest x-ray imaging. This is just one of many examples in which computer vision would be useful for the pandemic.

Reference

Cabani, A.; Hammoudi, K.; Benhabiles, H.; and Melkemi, M. 2020. MaskedFace-Net - A dataset of correctly/incorrectly masked face images in the context of COVID-19, *Smart Health*, Elsevier

Chavda, A.; Dsouza, J.; Badgujar, S.; and Damani, A. 2020. Multi-Stage CNN Architecture for Face Mask Detection. arXiv:2009.07627

Christoforou, C.; Constantinidou, F.; Shoshilou, P.; and Simos, P. (2013). Single-trial linear correlation analysis: application to characterization of stimulus modality effects. Frontiers in Computational Neuroscience 7, 15

Christoforou, C.; Haralick, R.M.; Sajda, P.; Parra, L.C. 2010. The bilinear brain: towards subject-invariant analysis, *In 2010 4th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pp. 1–6. IEEE, 2010.

Christoforou, C.; Hatzipanayioti, A.; and Avraamides, M. 2018 Perspective Taking vs Mental Rotation: CSP-based single-trial analysis for cognitive process disambiguation. In Wang, S., Yamamoto, V., Jianzhong S., Yang Y., Jones, E., Iasemidis, L., Mitchell, T., (Eds.) *Proceedings of International Conference, Brain Informatics* (pp. 109-199). Arlington, TX, USA

Ellis, R. 2020. Face Masks Reduce COVID Infection Risk by 65%. Retrieved November 15, 2020, from https://www.webmd.com/lung/news/20200710/face-masks-reduce-covid-risk-by-65-percent

Fang, Qi;Li, H.; Luo, X.; Ding, L.; Luo, H.; Rose, T.; and An,W. 2018. Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. *Automation in Construction*. 85. 1-9. 10.1016/j.autcon.2017.09.018

Germa, G.; and Dhillon, A. 2017. A Handheld Gun Detection using Faster R-CNN Deep Learning. *Proceedings of the 7th International Conference on Computer Vision and Communications Technology—ICCCT2017)* 84-88. 10.1145/3154979.3154988

Gelana, F.; and Yadav, A. 2019. Firearm Detection from Surveillance Cameras Using Image Processing and Machine Learning Techniques: *Proceedings of ICSICCS-2018*. 10.1007/978-981-13-2414-7_3

Hammoudi, K., Cabani, A.; Benhabiles, H.; and Melkemi, M. 2020. Validating the correct wearing of protection mask by taking

a selfie: design of a mobile application "CheckYourMask" to limit the spread of COVID-19, *CMES-Computer Modeling in Engineering & Sciences*, Vol.124, No.3, pp. 1049-1059, 2020, DOI:10.32604/cmes.2020.011663

Karras, T.; Laine, S.; and Aila, T. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR-2019), Long Beach, CA, USA*, 2019, pp. 4396-4405, doi: 10.1109/CVPR.2019.00453.

Lasseck, M. 2017. Image-based Plant Species Identification with Deep Convolutional Neural Networks. *CLEF.*

Mneymneh, B. E.; Abbas, M.; and Khoury, H. 2019. Vision-Based Framework for Intelligent Monitoring of Hardhat Wearing on Construction Sites. *Journal of Computing in Civil Engineering*. 33. 10.1061/(ASCE)CP.1943-5487.0000813

Nguyen, N.; Le, V.; Le, T.; Hai, V.; Pantuwong, N.; and Yagi, Y. 2016. Flower species identification using deep convolutional neural networks. *Regional Conference on Computer and Information Engieering (RCCIE-2016*).