Face Mask Detection

COMP 6721: Applied Artificial Intelligence Project

Group Name: SS_G06
Data Specialist: Marwa Abdallah (40187247)
Training Specialist: Nima Sarang (40156778)
Evaluation Specialist: Shahrzad AminRanjbar (40087038)

Concordia University
Montreal, Canada

Fall- 2020

Figure 1: *Samples from the dataset.*

1 Introduction

In the light of COVID-19 pandemic, wearing a face mask became a crucial daily routine that all people are getting used to [1]. Wearing face masks reduces the transmission of the virus. However, sometimes people fail to wear them properly (or at all). Accordingly, determining if an individual is wearing a face mask or not is an important task. In this project, we employ deep convolutional neural networks (CNNs) to analyzes face images and classify whether a person is wearing a face mask or not.

2 Dataset

To train the model, we carefully selected a dataset consisting of three classes: 1) Person with face-mask 2) Person without face-mask 3) Not a person. The dataset should be large enough for the model to learn the underlying features of each class. However, too large datasets require powerful hardware (i.e., GPU) to train the model. A compromise should be obtained between the learning capacity and the training time.

For each of the three classes, we collected the data images from the sources as shown in table 1. We split the dataset to three subsets, namely, train, validation, and test sets.

Table 1: *Different sources for the dataset classes.*

Class	Source
Class 1: Person with face mask	**MaFA** Dataset from NVIDIA Developer [2]
Class 2: Person without face mask	Flickr-Faces-HQ dataset **(FFHQ)** [3]
Class 3: Not a person	used 18 class from **ImageNet** dataset [4]

Table 2: *Number of images in each class.*

Class	Train	Validation	Testing	Total
Class 1: Person with face-mask	3200	800	1000	5000
Class 2: Person without face-mask	3200	800	1000	5000
Class 3: Not a person	3200	800	1000	5000

The number of the images in each class are shown in table 2.
Then we split the dataset with 15,000 data points into 10 subsets with 1500 data points on each subset equally to be ready for 10-fold cross-validation as explained in the coming sections.

As shown in table 2, we sampled the same number of images for each class. This simplifies the training and alleviate the problems of an imbalanced dataset [5]. The images in each class are not of the same size, however, we resize the images to 128×128 resolution in the pre-processing stage, which keeps good feature representation and speeds-up the training time; it also used in many literature work [6, 7, 8].

We also standardize the images for each of their color channels, based on mean and standard deviation of the color values in the training dataset.

3 Network Architecture

In this section, we introduce our neural network architecture that we used to classify images. Since our data are in the form of images, we've be mainly used convolutional layers to capture spatial features. We define a convolutional block as a convolutional layer, followed by Batch Normalization [9] and ReLU activation function. We employ batch normalization in order to stabilizing the training process by re-centering and re-scaling the inputs of the layers. It has been shown that tbis also has an effect of making the training faster. For downsampling the feature maps, strided convolutions were used instead of pooling layers, since they have learnable filters that can be adjusted, versus the pooling layers which are fixed operations.

We also used Squeeze-and-Excitation blocks [10] which is a channel attention mechanism that assign weights to each feature channel and model interdependencies between them. For the classification head of the network, we added Dropout [11] before the final linear layer to prevent the model from overfitting.

4 Training

For training, we used PyTorch Lightning [12] that provides tools such as saving model checkpoints, logging to Tensor-Board and multi-GPU training. For the optimization part, AdamW [13] which provides global weight decay regularization was used. We experimented on a set of learning rates from 0.0001 to 0.001 and found that 0.0005 gives the best result, with a batch size of 64. For the Dropout normalization, the ratio of

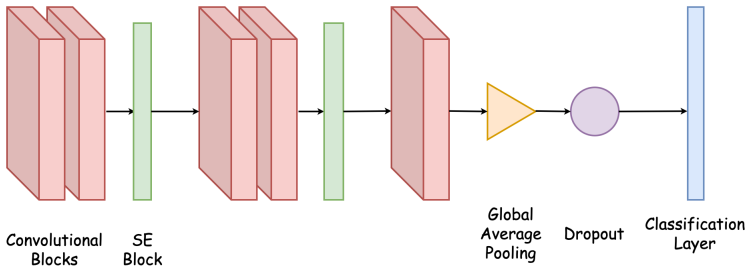
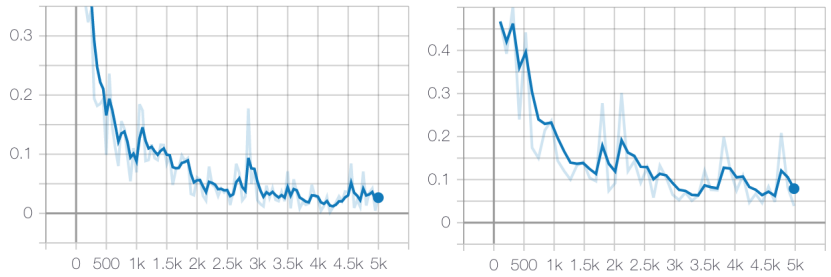


Figure 2: *The network architecture employed in our model. A combination of convolutional and squeeze-and-excitation blocks were used, followed by global average pooling, dropout and the classification layer.*

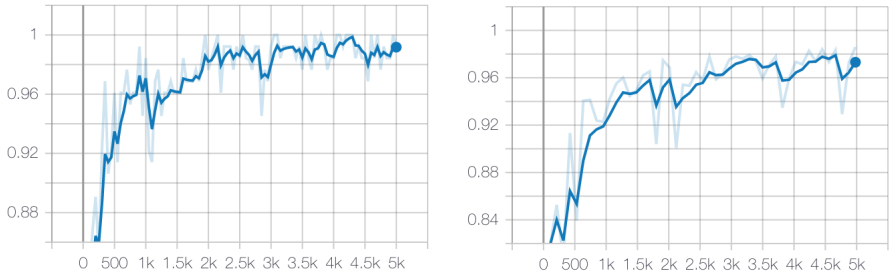
zeros was set to 0.2.

For the loss function, since our task is multi-classification, cross entropy loss was used. Since our data set is balanced over all classes, we did not use a weighted loss function. For choosing the best model and stopping the training early, we monitored the validation accuracy. Overall, the model was trained for at most 60 epochs.



(a) Train loss (b) Validation loss

Figure 3: *Loss curves for train and validation sets*



(a) Train accuracy (b) Validation accuracy

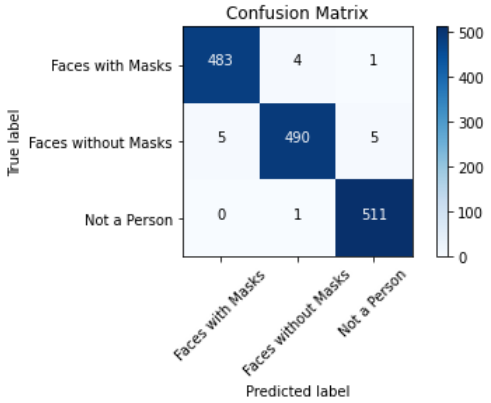
Figure 4: *Classification accuracy curve for train and validation sets*

5 Evaluation

In this section, we evaluate the model and measure it's performance based on the predicted labels and target labels. This can be obtained by using the Pytorch DataLoader for the test data and collecting the predictions for the test set, then comparing the predictions to the target labels of the test set [14]. The performance metrics used are precision, f1-score, recall and confusion matrix, provided by the scikit-learn package [15]. Our model achieved an average of 99% accuracy on the test images. The f1-score of 99 % shows that the performance of our model is decent across different classes and also there isn't a big gap between the precision and recall scores, since f1-score is the harmonic mean between precision and recall. It's also worth noting that the validation and test accuracies are fairly similar. This shows that the model was not overfit to the training data.

	precision	recall	f1-score	support
Faces with Masks	0.99	0.99	0.99	488
Faces without Masks	0.99	0.98	0.98	500
Not a Person	0.99	1.00	0.99	512
accuracy			0.99	1500
macro avg	0.99	0.99	0.99	1500
weighted avg	0.99	0.99	0.99	1500

(a) Evaluation report



(b) Confusion matrix of the test set

The Figure 5b shows that most of the error were between faces-with-masks and not-a-person classes. If we don't consider the second row and column, the number of misclassifications between these two classes will be more obvious. Perhaps one explanation is that when a person's face is covered, it will be a little bit harder for the model to correctly distinguish it from an object.

6 K-Fold Cross-Validation

The k-fold Cross-Validation (KCV) technique is one of the most used approaches by practitioners for model selection and error estimation of classifiers [16]. It is a popular method because it is simple to understand and it is generally results in a less biased or less optimistic estimate of the model skill than other methods. As shown in 5, the KCV splits a dataset into k folds. At each iteration, one fold is used as the test set and the rest of dataset is used to train the model.

Figure 5: *10-fold cross-validation [17]*

In our experiments, we used k= 10. We split the dataset into 10 subsets and trained the model 10 times on different sets. Random shuffling and a fixed random seed was used before splitting. Table 3 shows the cross-validation results.

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	Fold10	Mean	SD
Accuracy	98.9	98.6	99.2	98.5	99.1	99.0	98.4	99.2	99.0	98.5	98.84	0.29

Table 3: *Validation accuracy results of the 10-fold cross-validation experiment*

The results show that the standard deviation of the validation accuracy of different folds is very small (0.3 %). As a result, it's fair to conclude that our data does not exhibit a considerable bias and has a high variance. Otherwise, at least one fold should've yielded a tangible decrease in the accuracy compared to the others. **In comparison to Part I**, the results in terms of accuracy are also fairly similar. This is also on par with the similarity of the results between different folds.

References

[1] IBM, "Face mask detection at the edge," (Accessed November 15, 2020). [Online]. Available: https://www.ibm.com/cloud/blog/face-mask-detection-at-the-edge

[2] N. Developer, "Face mask detection at the edge," (Accessed December 1, 2020). [Online]. Available: https://developer.nvidia.com/blog/implementing-a-real-time-ai-based-face-mask-detector-application-for-covid-19/

[3] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, "Maskedface-net–a dataset of correctly/incorrectly masked face images in the context of covid-19," *Smart Health*, p. 100144, 2020.

[4] "Large scale visual recognition challenge 2017 (ilsvrc2017)," http://image-net.org/challenges/LSVRC/2017/downloads, accessed November 5, 2020.

[5] H. Choi, "Cnn output optimization for more balanced classification," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 17, no. 2, pp. 98–106, 2017.

[6] R. Kumar, Z. Xiaosong, R. U. Khan, I. Ahad, and J. Kumar, "Malicious code detection based on image processing using deep learning," in *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, 2018, pp. 81–85.

[7] N. Codella, J. Cai, M. Abedini, R. Garnavi, A. Halpern, and J. R. Smith, "Deep learning, sparse coding, and svm for melanoma recognition in dermoscopy images," in *International workshop on machine learning in medical imaging*. Springer, 2015, pp. 118–126.

[8] A. F. Al-Daour, M. O. Al-Shawwa, and S. S. Abu-Naser, "Banana classification using deep learning," 2020.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[10] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018. [Online]. Available: https://doi.org/10.1109/cvpr.2018.00745

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[12] W. Falcon, "Pytorch lightning," *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[13] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2018.

[14] J. Brownlee, "Pytorch tutorial: How to develop deep learning models with python," https://machinelearningmastery.com/pytorch-tutorial-develop-deep-learning-models/, accessed November 19, 2020.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella, "The'k'in k-fold cross validation." in *ESANN*, 2012.

[17] A. Talpur, "Congestion detection in software defined networks using machine learning," Ph.D. dissertation, PhD thesis, 02 2017, 2017.