Face Mask Detection in Real-time for the Era of COVID-10 Pandemic

Jinan University

VAN SONGHIENG

2020059118

***Abstract*: The Corona Virus Disease (COVID-19) pandemic is still a challenging health catastrophe that has an impact on people's lives all over the world as of 2022. But if we're talking about how to avoid contracting this illness, wearing a mask is the first measure that most people think is successful. Because of this, mask detection is crucial to daily living in the pandemic age. The MobilenetV2 and VGG Mask detection technique is presented in this work. To instruct and detect whether a face is covered in a mask or not from sets of photos, live camera footage, movies, etc., two databases of face masks have been used. This experiment shows that 1200 samples can be validated with 99% accuracy.**

Keyword: COVID-19, MobileNetv2, VGG, Tensorflow, Keras, Machine learning.

I. INTRODUCTION

Since the declaration of the COVID-19 virus as a pandemic by WHO, Face mask stated that efforts have been made by various parties to reduce the spread of the virus. Wearing a face mask in the public place to impede COVID-19 transmission. Furthermore, since more rules being implemented, the people are forced by law to wear a face mask in the public place and wherever they interact with other people.

The process of monitoring a large population that has a different habit. The authorities need a solution to be able to validly control the implementation of the law, which begins with the availability of the data quickly and accurately. One of the solutions is to use a regionally automated face mask recognition to differentiate between people who wear masks and those who do not.

II. METHOD

The face mask recognition in this study is developed with a machine learning algorithm through the image classification method: MobileNetv2 and VGG. MobileNetV2 is a method based on Convolutional Neural Network (CNN) that developed by Google with improved performance and enhancement to be more efficient. This study conducted its experiments on two original datasets. The first dataset was taken from the Kaggle dataset and the Real-World Masked Face dataset (RMFD); used for the training, validation, and testing phase so the model can be implemented to the dataset.

The model can be produced by following some steps which are

(1) Data collecting.

(2) Pre-processing.

(3) Split the data.

(4) Building the model.

(5) Testing the model.

(6) implement the model.

III. PROCESSING

1. Data Collection

Before creating the model, machine learning model need to have the dataset first. There are several methods for gathering data from this. One option is to utilize an open-source program that is available online, another is to screenshot images from Google one after another, and a third is to build the data on our own by taking pictures of ourselves or asking others to contribute photos to the dataset. The URL I use to create my data set is as follow https://generated.photos/faces/.

2. Pre-processing

Due to the efficacy of training, picture scaling is a significant pre-processing step in computer vision models. The lower the picture size, the better the model will run. In this study, enlarging an image result in 224 × 224 pixels.

The next step is to convert all of the photos in the dataset into an array. The picture is transformed into an array for use by the loop function. Following that, the picture will be utilized to pre-process input using MobileNetV2.

The final step in this phase is to do hot encoding on labels because many machines learning algorithms cannot operate immediately on data labeling. They demand that all input and output variables, including this procedure, be numeric. The tagged data will be converted into a numerical label so that the algorithm can comprehend and interpret it.

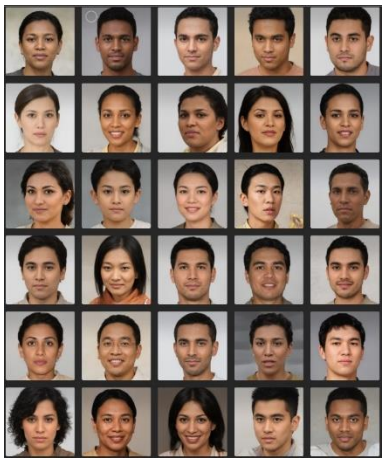

***Figure 1*: Sample of the group of Data**

3. Split the Data

After we have completed the data collection, we may divide the dataset into two groups, one for wear masks and one for non-wear masks. We should also note that even one data collection

cannot be combined with another. We must also eliminate the duplicate images.

Below is the Sample of the images:

Figure 2: **Person without mask**

Figure 3**: Person with mask**

Below is all the amount of dataset that we have collected for the training model: **We have 1602 for mask data set and 1746 for without mask dataset.**

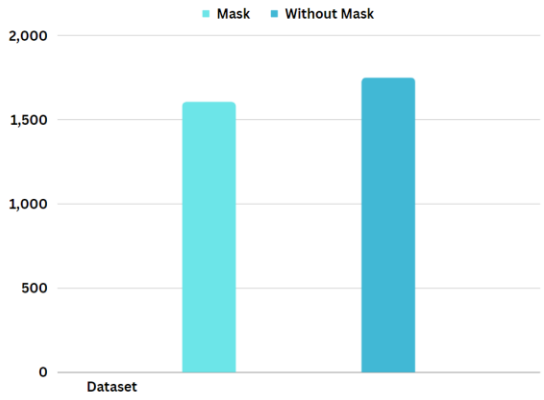

Figure 4**: Dataset Diagram**

4. Building the Model for MobileNetV2

The next phase is building the model. There are six steps in building the model which are shown in the below figure.

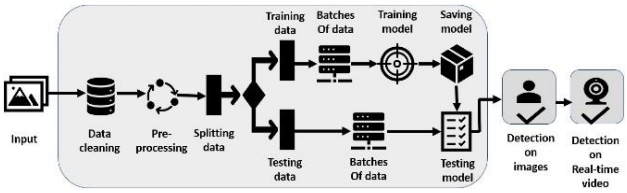

Figure 5: **MobilevNet2 Algorithms**

Algorithm 2: Deployment of Face Mask Detector in VGG

Pre-processing and Training on Dataset:

1. Load images with pixel values.
2. Resize the photos and convert them to a 1D array.
3. Load the filenames as well as their labels.
4. perform data augmentation and then split the data into training and testing batch.
5. Install the MobilenetV2 model from Keras. Compile and train it on training batches.
6. Store the model.

Deployment of Face Mask Detector

1. Load the previously saved classifier from disk. Also, load the face detector from OpenCV.
2. If the option is image classification, then load images.
3. if the choice is classification in real-time.
4. Load a real-time feed from OpenCV.
5. Read the feed frame by frame.
6. Apply a face detection model to detect faces in a frame read in real-time.
7. If faces are found, crop them to the bounding box using the face detection model.
8. The face classifier model predicts.
9. Display the output in real time and feed.
10. Else, show the normal feed.
11. End the Stream by pressing q.

5. Testing the Model for MobileNetV2

To make sure the model can predict well, there are steps in testing the model. The first step is making predictions on the testing set. The result for 20 iterations in checking the loss and accuracy when training the model is shown in Table 1.

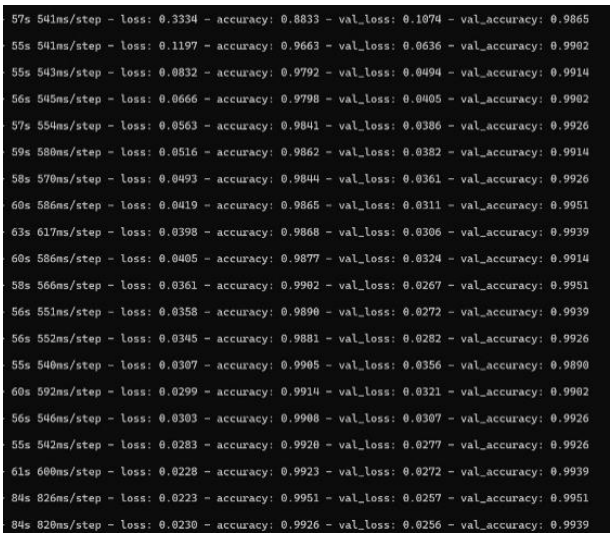

***Figure 6*: Result from checking Model lose and accuracy**

Table I. Iteration of checking the loss and accuracy

Epoch	Loss	Accuracy	Val_loss	Val_acc
1/20	0.333	0.883	0.107	0.986
2/20	0.119	0.966	0.630	0.990
3/20	0.083	0.979	0.049	0.991
4/20	0.066	0.979	0.040	0.990
5/20	0.056	0.984	0.038	0.992
6/20	0.059	0.938	0.172	0.991
7/20	0.059	0.941	0.269	0.992
8/20	0.129	0.954	0.276	0.995
9/20	0.151	0.945	0.322	0.993
10/20	0.136	0.949	0.260	0.995
11/20	0.118	0.958	0.214	0.993
12/20	0.120	0.959	0.354	0.992
13/20	0.106	0.963	0.179	0.989
14/20	0.118	0.956	0.381	0.990
15/20	0.128	0.954	0.310	0.992
16/20	0.108	0.962	0.273	0.992
17/20	0.107	0.957	0.210	0.993
18/20	0.108	0.957	0.257	0.995
19/20	0.106	0.959	0.217	0.993
20/20	0.091	0.968	0.250	0.993`

From the above figure and Table, we can see that the accuracy is increasing start on second epoch, and loss is decreasing after it. So then, next step is making the model evaluation as shown the below Figure.

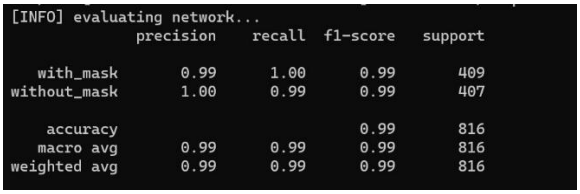

***Figure 7*: Result from the real time Model Evaluation**

Table II. Model Evaluation

	Precision	Recall	F1-Score	Support
With mask	0.99	1.00	0.99	409
Without mask	1.00	0.99	0.99	407
Accuracy			0.99	816
Macro avg	0.99	0.99	0.99	816
Weighted avg	0.99	0.99	0.99	816

Training Loss and Accuracy

***Figure 8:* Training Lose and Accuracy on MobilenetV2**

6. Building Model for VGG

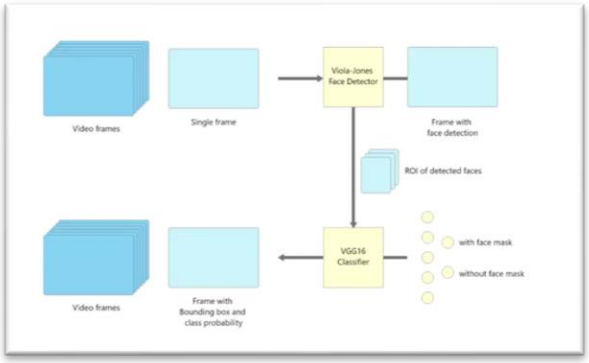

Figure 9: **VGG Algorithms**

Algorithm 2: Deployment of Face Mask Detector in VGG

1. Converting BGR image to RGB as cv2.imread() function reads image as BGR.
2. Resizing the image to 160 x 160 x 3.
3. Converting the image into the NumPy matrix.
4. Rescaling all pixels from 0 to 1
5. Resizing the image matrix to #images, 160, 160, 3.
6. Converting target labels into one-hot vector.
7. Splitting into train set, validation set, and test set.

Figure 10**: Structure of the VGG-16**

7. Testing model for VGG

Below are the results which we have receive after complete the training model.

Table III. Iteration of checking the loss and accuracy

Epoch	Loss	Accuracy	Val_loss	Val_acc
1/11	0.0512	0.8831	0.0503	0.9745
2/11	0.0677	0.9729	0.0707	0.9759
3/11	0.0424	0.9846	0.0751	0.9699
4/11	0.0415	0.9831	0.0875	0.9608
5/11	0.0337	0.9883	0.0456	0.9789
6/11	0.0322	0.9906	0.0537	0.9699
7/11	0.0294	0.9895	0.0592	0.9789
8/11	0.0125	0.9962	0.0716	0.9699
9/11	0.0133	0.9959	0.0549	0.9729
10/11	0.0209	0.9932	0.0928	0.9789
11/11	0.0105	0.9959	0.0922	0.9669

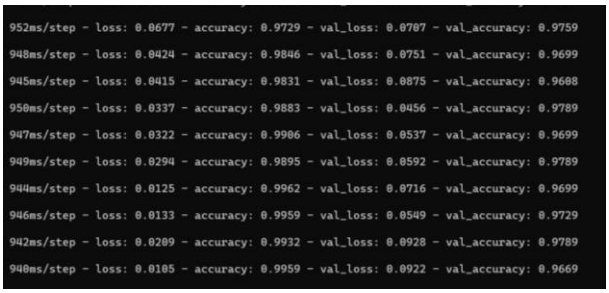

Figure 11**: Result from checking lose and accuracy**

When the accuracy line is stable, it signifies that there is no need for more iterations to improve the model's accuracy. The following step is to evaluate the model.

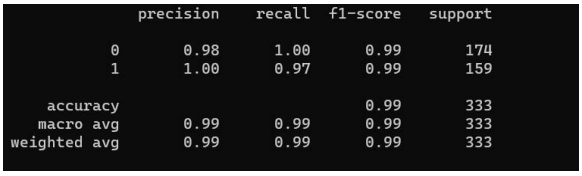

Figure 12**: Result from the real time Model Evaluation**

Table IV. Model Evaluation

	Precision	Recall	F1-Score	Support
With mask	0.98	1.00	0.99	174
Without mask	1.00	0.97	0.99	159
Accuracy			0.99	333
Macro avg	0.99	0.99	0.99	333
Weighted avg	0.99	0.99	0.99	333

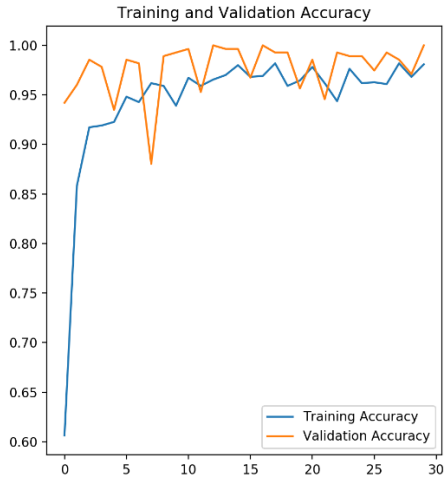

Figure 13: **Training Accuracy on VGG**

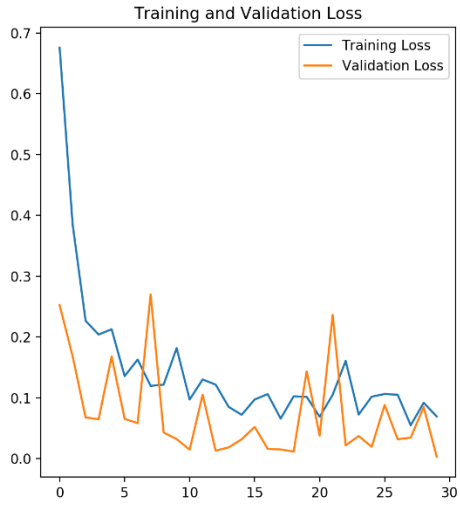

Figure 14: **Training Loss on VGG**

Evaluation Metrics

To evaluate the performance of the proposed model, many performance measures are required. "Accuracy, precision, F1 score, and recall" are the performance measurement parameters, and they may be found in equations (1-4) below. "True Positive" is "TP", "True Negative" is "TN", "False Positive" is "FP", and "False Negative" is "FN".

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

$$"Recall = \frac{TP}{TP+FN}" \quad (2)$$

$$"Accuracy = \frac{(TP+TN)}{[(TP+FP)+(TN+FN)]}" \quad (3)$$

$$Fl\ score = 2\ \frac{Precision*Recall}{(Precision+Recall)} \quad (4)$$

Where Tp = Truepositive,
Tn = Truenegetive
Fp = Falsepositive
Fn = Falsenegative

IV. COMPARISION

MobileNetV2

MobileNet's second version was released in April 2017. MobilenetV2 and MobilenetV1 both use depth and pointwise convolution. The inverted residual layer and the linear bottleneck layer are new CNN layers in MobileNetV2. MobileNetV2 has two new features: 1) linear bottlenecks and 2) shortcut linkages between bottlenecks in the bottleneck zone, there is input and output between models, but the inner layer or layer encapsulates the model's capacity to transform lower-level concepts (i.e., pixels) to higher-level descriptors (i.e., image categories). Finally, like with residual connections in traditional CNNs, eliminating bottlenecks allows for faster training and higher accuracy.

VGG

The VGG16 convolutional neural network model was suggested by K. Simonyan and A. Zisserman of the University of Oxford. VGG16 is a Convolutional Neural Network (CNN) architecture that was used to win the 2014 ILSVR (ImageNet) competition. It is widely recognized as one of the best vision model designs ever created. The most notable aspect of VGG16 is that, rather than a large number of hyper-parameters, they focused on 33 convolution layers with stride one and always used the same padding and max-pooling layers 22 filter with stride 2. This sequence of convolution and max-pooling layers is maintained throughout the design. Finally, for output, it has two FC (Fully Connected layers) and a soft max. The number 16 in VGG16 refers to its sixteen weighted layers. It is a massive network with an estimated 138 million parameters.

Training Result Analysis

This size adjusts the size of the pre-trained model from ImageNet used in the VGG16 and MobileNetV2 models. The pre-trained model will classify the identities of people who wear masks. Before testing using other image data, we tuned the batch size and epoch parameters to see the accuracy values for training and validation. The best results fromthe batch size values were used to experiment with variations in the number of epochs, namely 10, 20, 30, 40,and 50. In this study, we used Adam as optimizer and binary_cross entropy as a loss. **Table V** shows the experimental resultson the variation in the value of the number of epochs in the training data process.

Testing Result Analysis

The accuracy results matching data testing with VGG16 and MobileNetV2

Pre-trained Model	Accuracy
VGG16	93.47%
MobileNetV2	96.62%

Table V. Experiments in the training data stage using the number of epochs variations

Pre-trained Model	10 Epoch		20 Epoch		30 Epoch		40 Epoch		50 Epoch	
	Train Acc.	Val Acc.	Train Acc.	Val Acc.	Train Acc.	Val Acc.	Train Acc.	Val Acc.	Train Acc.	Val Acc.
VGG16	0.1273	0.2088	0.5717	0.4688	0.8328	0.8905	0.9369	0.9119	0.9657	0.9244
MobileNetV2	0.9723	0.9688	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

I. Result

The video's use of the model. The face detection method operates after the video has been reviewed from frame to frame. The method moves on to the next step if a face is found. Face detection frames will perform reprocessing, which includes enlarging the picture size, converting to an array, and pre-processing input using MobileNetV2.

For MobileNetV2

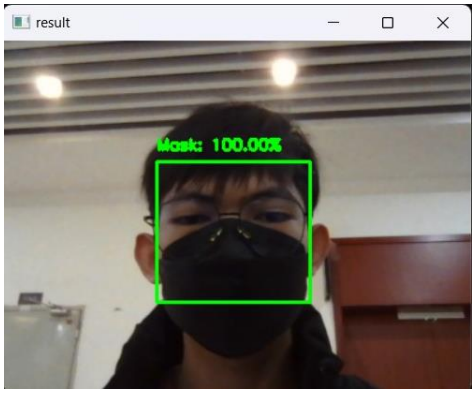

Figure 15: **with Mask using MobileNetV2**

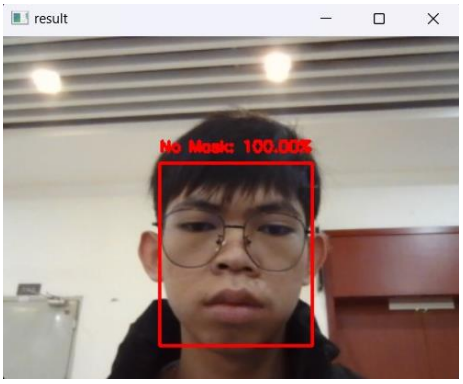

Figure 16: **Without Mask with green code**

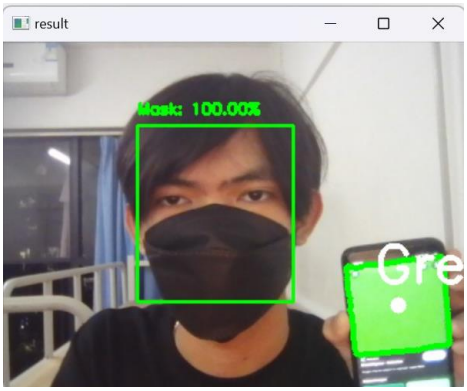

Figure 17: **with Mask with green code**

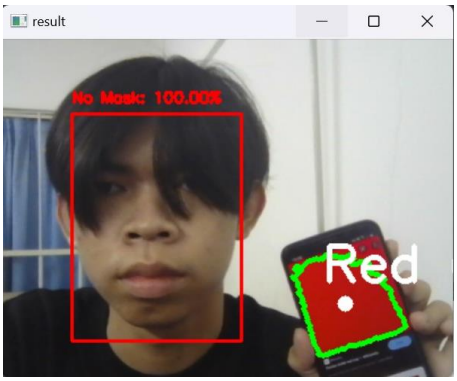

Figure 18: **Without Mask using MobileNetV2**

FOR VGG

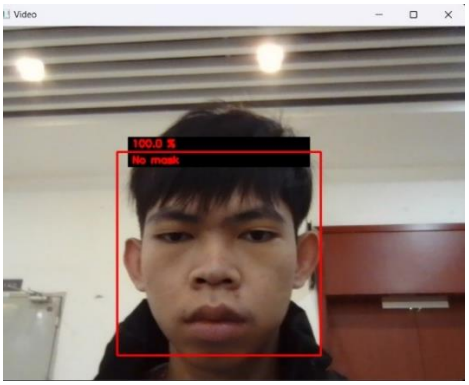

Figure 19: **Without Mask using VGG**

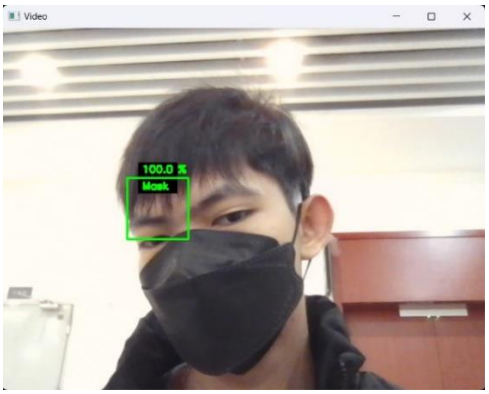

Figure 20: **with Mask using VGG**

II. CONCLUSION

Because of the unexpected advent of the COVID-19 epidemic, numerous facial recognition technologies are now being used on people wearing masks. The detection of face masks is a difficult job for face detectors. This is due to the fact that faces wearing masks have varying accommodations, degrees of occlusion, and mask kinds. There are several Face Mask detection models. These are classified into two types of two algorithms which are MobilenetV2 and VGG.

III. FUTURE WORK

As you can see, one of the pictures on the result shows one of me holding the phone with the camera detecting the colors red and green. In the MobilevNet2 project, I've also included some lines of code that detect the colors red, yellow, and green. As we can see, in China, there is a QR code to identify the health of tourists, thus the points of this color detection are responsible for the QR code detection. To summarize, this entire project is extremely important for safety purposes during the Covid-19 era, and it also helps a lot for guards' jobs to inspect the travelers one by one and change to use this software.

REFERENCE

Qin, B., & Li, D. (2020). Identifying Facemask-wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID-19.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: The next generation of on-device computer vision networks. URL Https://Ai. Googleblog. Com/2018/04/Mobilenetv2-next-Generation-of-on. Html.

Jiang, M., Fan, X., Yan, H.: RetinaMask: a face mask detector.
arXiv:2005.03950 [cs]. (2020)

M. Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR), 2018

A. Ahmad, "Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning," J. Teknol. Indones., no. October, 2017. https://www.academia.edu/download/54674088/Perbedaan_Deep_learn.pdf [22] R. Primartha, "Belajar Machine Learning; Teori dan Praktik," 2018.

http://repo.unikadelasalle.ac.id/index.php?p=show_detail&id=12893&keywords=

F. D. Adhinata, D. P. Rakhmadani, M. Wibowo, and A. Jayadi, “A Deep Learning Using DenseNet201 to Detect Masked or Non-masked Face,” vol. 9, no. 1, pp. 115–121, 2021. https://doi.org/10.30595/juita.v9i1.9624

K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1–14, 2015. https://arxiv.org/abs/1409.1556v6

K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, “Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection,” Constr. Build. Mater., vol. 157, no. September, pp. 322–330, 2017. https://doi.org/10.1016/j.conbuildmat.2017.09.110

W. Nash, T. Drummond, and N. Birbilis, “A review of deep learning in the study of materials degradation,” npj Mater. Degrad., vol. 2, no. 1, pp. 1–12, 2018. https://doi.org/10.1038/s41529-018-0058-x

D. Yu, Q. Xu, H. Guo, C. Zhao, Y. Lin, and D. Li, “An efficient and lightweight convolutional neural network for remote sensing image scene classification,” Sensors (Switzerland), vol. 20, no. 7, 2020. https://doi.org/10.3390/s20071999

M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 4510–4520, 2018. https://doi.org/10.1109/CVPR.2018.00474