Custom Convolutional Neural Network Model Implementation for Mask Detection on Atlas Hilens Device

Yosua Alvin Adi Soetrisno
Professional Engineering Education
Diponegoro University
Semarang, Indonesia
yosua@live.undip.ac.id

Sumardi
Professional Engineering Education
Diponegoro University
Semarang, Indonesia
sumardi@elektro.undip.ac.id

Agung Nugroho
Professional Engineering Education
Diponegoro University
Semarang, Indonesia
agung2nugroho@gmail.com

Jaka Windarta
Professional Engineering Education
Diponegoro University
Semarang, Indonesia
jakawindarta@lecturer.undip.ac.id

***Abstract*—Deep learning is one of the models used in many classifications because it can obtain many characteristics of objects. Health protocols are highly emphasized in this pandemic era. One of the issues that can arise at an event is the public's failure to wear masks. This study developed a system to detect people who are not wearing masks. Transfer learning from several existing deep learning models is used to build the model. The contribution of this research is trying to make a proper and straightforward model that could be implemented in "Atlas Hilens" to get portability. The MobilNetV2 or InceptionV3 model cannot be implemented in "Atlas Hilens" because a kind of FusionBatchNormalizationv3 layer is not supported. The CNN model built on "Atlas Hilens" will be compared to the MobileNetV2 or InceptionV3 model built with Tensorflow on a PC. "Atlas Hilens" can adopt the Tensorflow model to implement, although it needs adjustments from the existing detection model. The efficiency and accuracy of these two models will be compared. This model was trained with a dataset from many data sources on Github with a real-time scenario and several students. The success of this model is that it can detect people who are wearing or not wearing masks with sure accuracy. The accuracy obtained for real-time detection on "Atlas Hilens" is 77.27%, and the accuracy obtained for image detection on PC is 87.35%, with an average accuracy of 82.31%.**

Keywords—artificial intelligence, mask detection, CNN, mobilenetv2, deep learning, tensorflow, GPU

I. INTRODUCTION

Every resident is urged by the World Health Organization (WHO) to wear a mask as part of current pandemic protocols, one of which is the ability to maintain health protocols. The use of community masks will become less common as new normal conditions emerge. COVID-19 can be transmitted through people unfamiliar with wearing masks in certain situations. Closed-off areas that are frequently infected by COVID-19 are densely populated. Wearing a face mask is one of the best ways to protect yourself from disease, aside from getting vaccinated. Artificial Intelligence (AI) technology can be used to detect the use of masks in public areas [1]. Developer kits and in-chip AI systems are both options for implementing this technology. With the help of artificial intelligence (AI) kits, security cameras can monitor whether people are wearing masks correctly [2].

These technologies are being used in various industries, including image processing. In addition, AI and deep learning can be used to solve problems during a pandemic. COVID-19 transmission is still possible, even though the pandemic has been declared almost endemic. AI can predict the spread and growth of COVID-19 based on specific parameters. Imaging techniques like roentgenography and magnetic resonance imaging (MRI) datasets are frequently used in image processing to look for abnormalities and diseases (MRI). AI and deep learning can monitor health protocols based on this simple idea. Wearing a face mask is the most visible way to monitor health protocols. Face detection is the primary method of mask detection. If a part of the face is covered, it's safe to assume someone is wearing a mask [3].

The primary goal of this research is to develop a deep learning model for detecting the usage of the mask. "Atlas Hilens" is the name from Huawei of the AI developer kit that includes this model. The model for "Atlas Hilens" can be programmed using Huawei's cloud platform. Skills or interfaces are needed to implement the model. In Huawei's cloud platform, "ModelArt", there is a YoloV3 model that was a ready-made deep learning model for "Atlas Hilens" object detection. YoloV3 is widely applied in detecting various things and is one of the most commonly used models.

It was decided to use CNN instead of YoloV3 because this model is only available for "Atlas Hilens" devices in China. It's worth noting that, in addition to YoloV3, MobileNetV2 has also been used in previous studies. Because of FusionBatchNormalizationV3 in the convolution layer, the MobileNetV2 model cannot be used in "Atlas Hilens." A PC or laptop platform will be used to test the MobileNetV2 model compared to CNN's (Convolutional Neural Network).

Face detection is required before detecting the mask to confirm that the face is wearing the mask. A Caffe with Resnet architecture model is used to face detection. The Caffe model is employed on a PC in this research, while the model in "Atlas Hilens" employs a simpler Haarcascade to recognize faces. A CNN-trained mask detection model is used after the face detection model has successfully detected a face. External and mask datasets obtained from students were used to validate this model. The mask detection model is saved as a Tensorflow Frozen Graph and then converted to an offline model (.om) for use in "Atlas Hilens." An interface program is also being developed to read this model and apply it to the detection process. This model will be linked to a camera on the "Atlas Hilens" and will be able to be utilized in public spaces.

978-1-6654-7150-3/22/$31.00 ©2022 IEEE

II. Related Work

Various deep learning models are used in multiple investigations. This investigation demonstrates that numerous approaches and architectural models are used in the detection process. Deep learning models are employed for object detection and other image processing. The distinction is whether the deep learning architecture may be used without modification or if it needs to be adjusted [4].

There are occasions where transfer learning is required by introducing knowledge of past objects, and others require training with specific things from the start in the deep learning models that are already present. There are many methodologies in mask detection, such as using transfer learning, modifying the architecture after transfer learning, and using gradual techniques with multi-deep learning. Several practical application studies have chosen a suitable model for embedded devices [5].

The InceptionV3 model was chosen from various comparisons, including Xception, MobileNet, MobileNetV2, VGG15, and Resnet50. Five layers have been added to InceptionV3's network to replace the last 48 convolution layers. Other layers include average pooling, flattening, densifying, dropping-out, and inference with the activation function layer. Deep learning model architecture and layers are shown in Figure 1. In this case, the model architecture is altered even when using transfer learning.

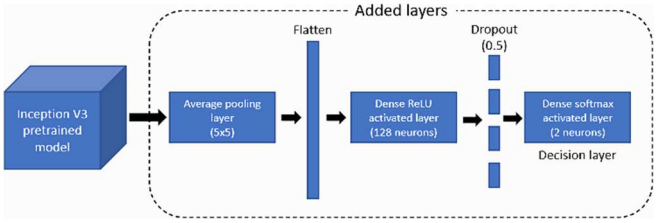

Fig. 1. Display of additional layers inserted in InceptionV3

"Face Mask Detection using Transfer Learning of InceptionV3" by G. Jignesh Chowdary [6] was conducted using the InceptionV3 model and had near 100% accuracy. Still, there was an issue in the crowd for unexplained detection. Even though MobileNetV2 and InceptionV3 have an accuracy close to 100%, it is still necessary to test these models to see if they can correctly detect masks in this research.

Amit Chavda's research [7], titled "Multi-Stage CNN Architecture for Face Mask Detection," performed a classification utilizing layered CNNs, demonstrating that one-layer CNN is insufficient and that detection stability issues exist. It gives the idea of this research to use one deep learning model for face detection and another for mask detection.

Another deep learning model, MobileNetV2, with 92% accuracy, is used in Preeti Nagrath's research [8] entitled "SSDMNV2: A real-time DNN-based face mask detection system using single-shot multi-box detector and MobileNetV2." The situation is different from this research and needs to be confirmed. Implementing a model to AI developer kit like "Atlas Hilens" is proof of concept and accuracy in the other study. It will also be tested with a dataset close to real-world conditions and one that includes student masks, using the model plant in the AI developer kit.

A ResNet 50-based deep learning model is used in Mohammed Loey's research [9], but the final layer of ResNet is replaced with traditional machine learning. These include SVM, a decision tree, and an ensemble learning algorithm. There are ways to tweak the final layer of deep learning to improve the detection of these masks, as demonstrated by this example. Figure 2 depicts the hybrid deep learning model's architecture.

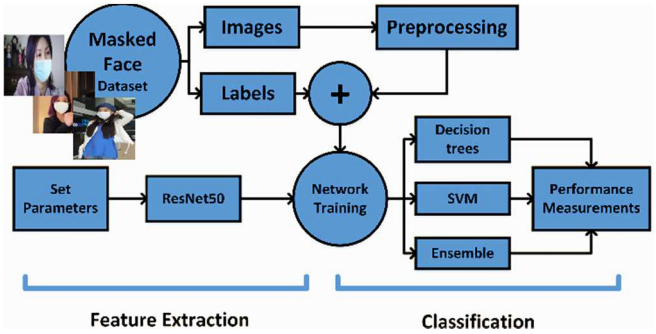

Fig. 2. Hybrid Deep Learning Model

III. Methodology

Mask detection is being developed in two stages: first, a CNN-based detection model is being developed and tested, and then an application for mask detection is being developed using the CNN model transferred from Tensorflow to "Atlas Hilens." Using models trained on Tensorflow that are saved in.h5 format in "Atlas Hilens" is impossible.

The .h5 model must be converted to a .pb model, a frozen graph that can be used to create an offline model. "Atlas Hilens" relies on a cloud-based development platform, so it cannot be programmed remotely. For experiments on a PC with a webcam and for "Atlas Hilens" connected to a webcam that is integrated with "Atlas Hilens," this mask detection system is used. The High Definition Multimedia Interface (HDMI) will directly show the detection results on a screen (HDMI). Figure 3 provides an overview of the system's overall design.

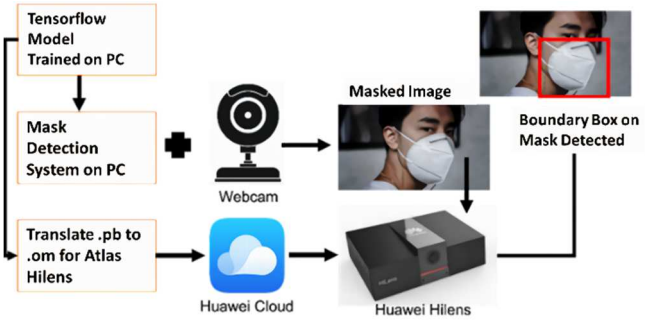

Fig. 3. Mask recognition system architecture with PC and "Atlas Hilens"

Tensorflow model saved using the h5 format. The Keras library was used to construct this Tensorflow model, hence the.h5 file extension. Deep learning convolutional layers can be built quickly and easily with Keras's sequential libraries. Keras simplify Tensorflow's library, so it only needs to be applied to layers that exhibit intuitive, deep learning properties. Because it is based on Keras, the.h5 model must be transformed into a Tensorflow frozen graph before it can be used as an offline model [10].

Nearly 60 million parameters are required to classify images using deep learning models. In addition, backward weighting during training requires calculating some of the same gradients. Each of these variables will have a value in the Tensorflow model. As a result, the.h5 model is frozen to .pb model so that it can be used repeatedly by identifying and storing all the necessary data (graphs, weights, configurations, and some parameters) in one file.

A mechanism for accessing deep learning libraries by OpenCV is required to use the offline deep learning model when creating interfaces to detect masks. Hilens's library will be called by the interface, which is slightly different. The Hilens libraries use other function calls. The "Atlas Hilens" device requires a color system and image format adjustments in addition to using the Hilens library. For example, the image captured on the camera can be changed from height, weight, channel to channel, height, and weight to adjust the image format. The face detection model used in Haarcascade uses a gray color scheme, so the YUV format from Hilens camera needed to be converted to a gray color scheme.

The use of Keras and computational graphs will be affected by the version of Tensorflow used. "Atlas Hilens" currently could only use version 1.xx or below version 2.xx for Tensorflow. Tensorflow 1.15 is used in this research because Keras sequential layers were used for training. Using FusedBatchNormalization, which is present in some built-in transfer models like MobileNetV2, is currently not possible in Tensorflow 1.15.

In training the model, hardware with high performance is needed because the training process requires quite heavy computing. An increasing number can see this computation of hidden layer architectures that detect important features and match existing features to decide whether to wear a mask or not. The more data and the complexity of the model created, the longer the training process will be. The hardware used in modeling the mask recognition system can be seen in Table 1.

TABLE I. HARDWARE REQUIREMENT

PC or Laptop Specification	
Specification	**Description**
Processor	Intel(R) Core(TM) i5 11400H
RAM	24 GBytes DDR 4
GPU	NVIDIA GeForce RTX 3050
Storage	475 GB NVMe WDC SSD
AI Developer Kit "Atlas Hilens" Specification	
Specification	**Description**
Processor	HiSilicon Hi3559A processor Dual-core ARM Cortex A73 @1.6 GHz, 32 KB I-Cache, 64 KB D-Cache/512 KB L2 cache
RAM	Processor: DDR4 4 GB, 64-bit, 2400 Mbit/s AI Accelerator: LPDDR4X, 128-bit, 8 GB, 3200 Mbit/s
GPU	Dual Da Vinci AI cores: 8 TFLOPS/FP16, 16 TOPS/INT8
Storage	Onboard 32 GB eMMC

Several steps are required to develop a mask recognition system model, beginning with collecting data in the form of photos of people wearing masks segregated from persons who do not wear masks and then proceeding with the modeling procedures as illustrated in Figure 4.

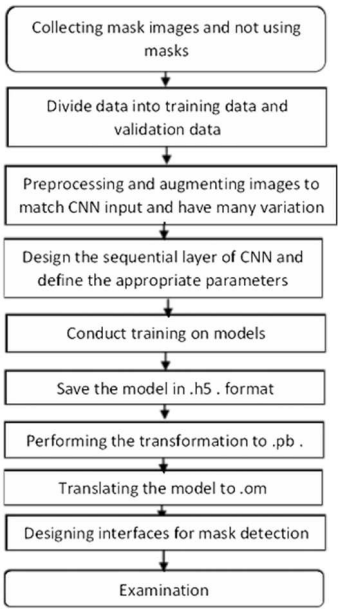

Fig. 4. Flowchart of the mask detection system

After the images have been collected, they will be divided into three categories: training data, test data, and validation data, as shown in Figure 4. Preprocessing is required after the image has been divided into sections to ensure that the image's resolution, color space, and color channel, as well as their length and width, are all compatible with the input from deep learning. A deep learning architecture can be built if the image entered follows the input format. Table 2 shows the CNN architecture that was used.

Three different convolution layer combinations may be observed in Table 3.3. This combination is done to gain properties ranging from the broadest to the most specific. The filter parameter indicates the number of kernels that must be twisted to obtain the activation map. The more filters you use, the more precise the traits you want to convolute become. The number of filters is commonly multiplied by 2n, so CNN utilizes 32, 64, and 128. The more filter variants needed, the more complicated the dataset you want to convolute.

There is a kernel size parameter in addition to the filter parameter. The kernel size specifies the length and breadth of the window used to execute the convolution. Starting with 1x1, 3x3, 5x5, and so on, the kernel size is frequently odd. The smaller the kernel size used, the smaller the area that system wants to understand as part of the image features

Because some features can only be observed when the capture area is more comprehensive, the larger the kernel size is employed to capture an immense amount. Because the size of the entered image will be converted to a dimension of 150x150, which does not require a kernel size larger than 3x3, the kernel size utilized in the model is 3x3 without change.

The value "same" is substituted for "valid" in the padding parameter. The padding parameter makes the spatial volume dimensions reasonable so that the output volume matches the input volume.

The "ReLu" function is used for activation, as mentioned in the preceding chapter. Using "ReLu" helps keep the computations required to run neural networks from becoming exponentially. Because the classification is whether or not to utilize a mask, sigmoid activation is used in the final layer.

TABLE II. CNN ARCHITECTURE AND PARAMETERS USED

Layer Type	Parameter	Output dimension	Num of Param #
conv2d (Conv2D)	filters=32, kernel_size=(3,3), padding=same, input_shape(150,150,3)), activation=relu	(None, 150, 150, 32)	896
conv2d_1 (Conv2D)	filters=32, kernel_size=(3,3), padding=same, input_shape(150,150,3)), activation=relu	(None, 150, 150, 32)	9248
max_pooling2d (MaxPooling2D)	pool_size=(2,2)	(None, 75, 75, 32)	0
dropout (Dropout)	0.5	(None, 75, 75, 32)	0
conv2d_2 (Conv2D)	filters=64, kernel_size=(3,3), padding=same, activation=relu	(None, 75, 75, 64)	18496
conv2d_3 (Conv2D)	filters=64, kernel_size=(3,3), padding=same, activation=relu	(None, 75, 75, 64)	36928
max_pooling2d_1 (MaxPooling2D)	pool_size=(2,2)	(None, 37, 37, 64)	0
dropout_1 (Dropout)	0.5	(None, 37, 37, 64)	0
conv2d_4 (Conv2D)	filters=128, kernel_size=(3,3), padding=same, activation=relu	(None, 37, 37, 128)	73856
max_pooling2d_2 (MaxPooling2D)	pool_size=(2,2)	(None, 18, 18, 128)	0
dropout_2 (Dropout)	0.5	(None, 18, 18, 128)	0
flatten (Flatten)		(None, 41472)	0
dense (Dense)	256,activation=relu	(None, 256)	256
dropout_3 (Dropout)	0.5	(None, 256)	0
dense_1 (Dense)	50, activation=relu	(None, 50)	12850
dropout_4 (Dropout)	0.5	(None, 50)	0
dense_2 (Dense)	1, activation=sigmoid	(None, 1)	51

Additional to convolution, a max-pooling layer is also present. It's a pooling operation that calculates the maximum value for a small chunk of the feature map and uses it to create a downsampled (pooling) feature map. It's like sifting through many images to find the most relevant characteristics. The input from 150x150 becomes 75x75 in the CNN shown in Table 3.3 due to the maximum pooling of 2x2. A dropout parameter of 0.5 is employed in this research, which means that the probability of the neuron not being used in specific backward propagation phases is 0.5.

The final layer consists of mechanisms that are flat and dense. The following is an example of density. Each of the five steps will be applied to a network with three inputs and 16 outputs, which is what we're looking for in this example: a Dense network with three inputs and 16 outputs. The output from the existing layer will be a sequence of vectors [D(x[0,:]), D(x[1,:]),..., D(x[4,:])] of the form [D(x[0,:]), D(x[1:]) (5, 16). To get a 15-dimensional vector, flatten it first and then perform a dense layer.

IV. EXPERIMENTS AND RESULTS

Figure 5 is an example of the detection results. In addition to using image detection, real-time detection is also carried out. Real-time detection produces 77% accuracy due to the camera's resolution factor and the quality of the image obtained by the camera. Figure 6 is an example of the detection results from the image.

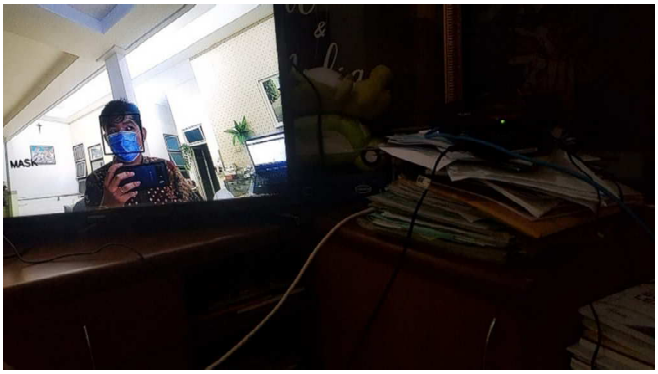

Fig. 5. Mask detection results in real-time

Fig. 6. Mask detection results from the image

Testing is the most critical step in developing a perfect model. As part of the testing process, analysis is also required to ensure that the model is ready to be used in the real world. The following are test results based on the PC model and the model used in "Atlas Hilens," which is applied in real-time. Accuracy, precision, sensitivity, specificity, and F1 score are metrics used to evaluate performance. As a metric, accuracy is the percentage (positive and negative) of correct predictions.

Accuracy is the performance indicator showing the percentage of correctly predicted and not correctly predicted overall data. Precision is the ratio of correct predictions to the overall positive expected outcome. Precision answers the question, "What percentage of people are correctly predicted to wear a mask of the total number of people who are predicted to wear a mask." Sensitivity answers the question, "What percentage of people are predicted to wear masks compared to all people who wear masks." The specificity answers the question, "What percentage of people are correctly predicted not to wear a mask compared to the total number of people who do not wear a mask" [11].

To calculate the F1 score, we weigh average precision and recall (specificity). If you're trying to detect whether or not someone is wearing a mask, you'll want to look at sensitivity as your guide. An additional comparison will be made by comparing the F1 score due to the asymmetry in the number of false positives and negatives. Table 3 will show the performance metrics of the various models tested.

TABLE III. COMPARISON OF THE PERFORMANCE OF EACH MODEL IMPLEMENTED IN "ATLAS HILENS" AND ON PC

Parameter	CNN on "Atlas Hilens"	CNN on PC	MobileNetV2 on PC
TP	11	1660	1912
FP	4	340	270
TN	6	1834	1405
FN	1	166	635
Accuracy = (TP + TN) / (TP+FP+FN+TN)	0,77	0,87	0,78
Precission = (TP) / (TP+FP)	0,73	0,83	0,87
Recall = (TP) / (TP + FN)	0,91	0,90	0,75
Specificity = (TN)/ (TN + FP)	0,6	0,84	0,83
F1 Score = 2 * (Recall * Precission) / (Recall + Precission)	0,81	0,86	0,8

As shown in Table 3, there were three separate experiments. The first experiment was with Hilens devices and real-time video. This experiment's model is based on a CNN. Only 22 of the image's data points are used in this experiment to make a detection. Figure 4.15 shows an example of a correctly detected image. The accuracy results, which is 77%, are still adequate, but the sensitivity results are excellent. Although the F1 score is lower than the CNN model that runs directly on a PC, the MobileNetV2 model is still better than the MobileNetV2 model.

The CNN model was used in the second experiment, but this time on a PC. Experiments on a PC with the camera in real-time indicated that the model performed better than when it ran on "Atlas Hilens." Table 3 does not show this, although the detection results are either mask-based or not much better. Misconduct is due to a discrepancy in camera sensors between the PC and "Atlas Hilens."

Even though the resolution of the PC camera and "Atlas Hilens" is the same as 720p or 1.3 MP, the quality is not always the same. The Frame Per Second (FPS) discrepancy in "Atlas Hilens" is due to the GPU speed difference between the NVIDIA GeForce RTX 3050 on the PC and the Da Vinci Core in "Atlas Hilens." Figure 7 shows an example of a video captured by the application from a PC camera.

Table 4 compares the current study's accuracy to that of various earlier research. The comparison uses accuracy because, in earlier research, there was no F1 score. Although the percentage accuracy of the implementation is below earlier, it could show that accuracy could be changed if the dataset is different.

TABLE IV. COMPARISON OF THE PERFORMANCE OF EACH MODEL IMPLEMENTED IN "ATLAS HILENS" AND ON PC

Model	Parameter	Percentage	Testing description
G. Jignesh Chowdary using InceptionV3 [6]	Accuracy	100%	Not good. There are some testing data misclassified
Amit Chavda [7] using Multi-Stage CNN entitled NasNetMobile	Accuracy	99.45%	There are no models to try in the Hilens library
Preeti Nagrath [8] uses the SSDMNV2 model that, like MobileNetV2	Accuracy	93%	It cannot be applied in Hilens because there is a FusionBatchNormalizationV3 layer
Atlas Hilens + CNN (Proposed Model)	Accuracy	77% (real time) and 87% (on PC)	Using regular CNN, but accuracy is different from PC due to camera sensor problems and also GPU quality

"Atlas Hilens" can only use the basic architecture, such as convolution functions, max pooling, dropout, flattening, dense, and several supported functions. FusedBatchNormV3 is an operator on MobileNetv2 released in 2009, primarily for NVIDIA's CUDA optimization. This operator is not supported on AI Ascend processors in mixed compute mode.

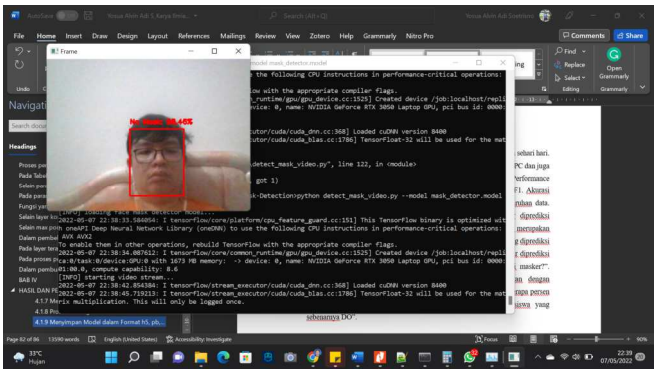

Fig. 7. Mask detection on PC using MobileNetV2

Table 3 shows an exciting conclusion, namely that the sensitivity of MobileNetV2 is relatively low compared to CNN. This demonstrates that MobileNetV2 can recognize the use of masks but not of detecting those who do not. Although MobileNetV2's architecture is more complicated than CNN's, this does not always imply that it performs better. The results of real-time detection on a PC using a CNN model are shown in Figure 8. In addition to the models depicted in Table 3, the InceptionV3 model was tested, but it did not perform well enough to be compared.

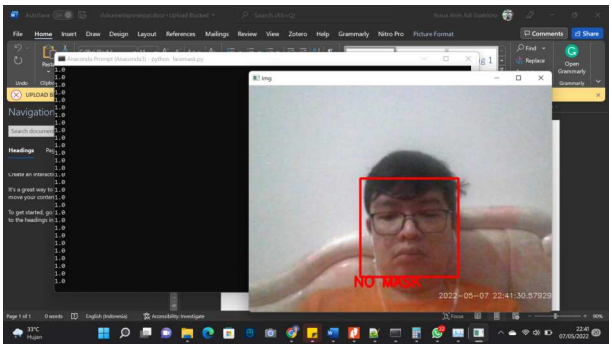

Fig. 8. Mask detection on PC using CNN

V. CONCLUSION

The CNN model on a PC has an accuracy of 87.35 percent, while the CNN model used in "Atlas Hilens" has an accuracy of 77.27 percent with an average of 82.31 percent. The arrangement of the convolution layer and classification layer in the deep learning architecture will affect the model's quality. Before mask detection, CNN is utilized for successful face detection. When applied to "Atlas Hilens" with real-time detection, accuracy can vary depending on image datasets. It indicates flaws in detecting black masks and a background nearly the same color as the skin or mask tested with student datasets.

Because this deep learning model can be used on mobile devices, it can be used more widely and is not limited to PCs. The forms of the CNN layer that can be applied and the properties of the layer are understandable. It can be an example of a project where simple AI applications are used using a direct case study on mask identification.

ACKNOWLEDGMENT

This work is supported and fully funded by Strategic Research Annual Budget Grand from the Professional Engineering Education, part of the Faculty of Engineering Diponegoro University.

REFERENCES

[1] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning: a review of classification and combining techniques," Artif. Intell. Rev., vol. 26, no. 3, pp. 159–190, Nov. 2006, doi: 10.1007/s10462-007-9052-3.

[2] F. De Felice and A. Polimeni, "Coronavirus Disease (COVID-19): A Machine Learning Bibliometric Analysis," In Vivo, vol. 34, no. 3 suppl, pp. 1613–1617, 2020, doi: 10.21873/invivo.11951.

[3] H. B. Syeda et al., "Role of Machine Learning Techniques to Tackle the COVID-19 Crisis: Systematic Review," JMIR Med. Inform., vol. 9, no. 1, p. e23811, Jan. 2021, doi: 10.2196/23811.

[4] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," Neurocomputing, vol. 187, pp. 27–48, Apr. 2016, doi: 10.1016/j.neucom.2015.09.116.

[5] A. Santoso and G. Ariyanto, "IMPLEMENTASI DEEP LEARNING BERBASIS KERAS UNTUK PENGENALAN WAJAH," Emit. J. Tek. Elektro, vol. 18, no. 01, pp. 15–21, Jun. 2018, doi: 10.23917/emitor.v18i01.6235.

[6] G. J. Chowdary, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Face Mask Detection using Transfer Learning of InceptionV3," ArXiv200908369 Cs Eess, vol. 12581, pp. 81–90, 2020, doi: 10.1007/978-3-030-66665-1_6.

[7] A. Chavda, J. Dsouza, S. Badgujar, and A. Damani, "Multi-Stage CNN Architecture for Face Mask Detection," in 2021 6th International Conference for Convergence in Technology (I2CT), Maharashtra, India, Apr. 2021, pp. 1–8. doi: 10.1109/I2CT51068.2021.9418207.

[8] P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria, and J. Hemanth, "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2," Sustain. Cities Soc., vol. 66, p. 102692, Mar. 2021, doi: 10.1016/j.scs.2020.102692.

[9] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic," Measurement, vol. 167, p. 108288, Jan. 2021, doi: 10.1016/j.measurement.2020.108288.

[10] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep Learning With TensorFlow: A Review," J. Educ. Behav. Stat., vol. 45, no. 2, pp. 227–248, Apr. 2020, doi: 10.3102/1076998619872761.

[11] R. Arthana, "Mengenal Accuracy, Precision, Recall dan Specificity serta yang diprioritaskan dalam Machine Learning." [Online]. Available: https://rey1024.medium.com/mengenal-accuracy-precission-recall-dan-specificity-serta-yang-diprioritaskan-b79ff4d77de8.