Face Mask Detection dengan CNN dan Pre-trained Caffee Models

Nicko Henry Hartopujo, Dean Ananda Ramadhan, Reinaldy Sukamto, *Binus University*

Abstraksi — Pandemi COVID-19 menyebabkan adanya insentif bagi masyarakat untuk menggunakan masker untuk meminimalisir penularan. Pemerintah Indonesia juga telah menegaskan pemakaian masker pada bisnis, maupun fasilitas umum. Dengan memanfaatkan banyaknya data yang tersedia, permintaan tersebut dapat di otomatisasi oleh teknologi yang berkembang belakangan ini. *Face recognition* merupakan salah satu *task* dari *computer vision* dalam mendeteksi wajah manusia. Daripada itu, kami mengajukan sebuah metode untuk mendeteksi apabila suatu wajah memakai masker, memakai masker dengan salah, ataupun tidak memakai masker sama sekali. *Mask detection* ini didasarkan pada sebuah model yang dilatih menggunakan *Convolutional Neural Network (CNN)* untuk bisa melakukan klasifikasi pada wajah-wajah tersebut.

Kata Kunci — *Computer Vision, Neural Network, Face Recognition, Mask Detection, Convolutional Neural Network*.

- *N.H.H. Author is with the Computer Science Department, Bina Nusantara University. E-mail: nicko.hartopujo@binus.ac.id.*
- *D.A.R. Author is with the Computer Science Department, Bina Nusantara University. E-mail: dean.ramadhan@binus.ac.id.*
- *R.S. Author is with the Computer Science Department, Bina Nusantara University. E-mail: reinaldy.sukamto@binus.ac.id.*

1 Pendahuluan

1.1 Latar Belakang

Maraknya kasus COVID-19 di Indonesia diakibatkan oleh salah satu sifat dan karakteristik dari virus ini yang mudah menular sehingga menyebabkan kasus COVID-19 yang meningkat secara signifikan. Hal tersebut dapat terjadi dikarenakan penyebaran utama daripada virus tersebut dapat melalui udara (*airborne)* yang dihirup oleh manusia. Kekhawatiran penyebaran COVID-19 juga disebabkan oleh pengidap yang bepergian yang tanpa sadar menularkan virus tersebut kepada orang disekitarnya.

Salah satu hal yang dapat meminimalisir penyebaran COVID-19 adalah dengan menggunakan masker untuk menutupi sumber masuknya virus COVID-19, yaitu bagian hidung dan mulut. Masker dapat berfungsi sebagai pencegah keluar masuknya virus COVID-19 dari orang yang sedang terpapar kepada orang belum/tidak terpapar virus ini.

1.2 Rumusan Masalah

Mengangkat tentang permasalahan penyebaran COVID-19 yang sedang merebak luas di Indonesia, dalam menghambat laju penyebaran COVID-19 Indonesia, masyarakat diwajibkan untuk saling bekerja sama dalam menahan laju penyebaran dengan saling menjaga jarak dan menggunakan masker. Oleh karena itu, kami mengembangkan sebuah model yang dapat mendeteksi apakah seseorang menggunakan masker atau tidak dan bahkan akan terdeteksi juga bilang orang tersebut memakai masker namun dengan cara yang kurang tepat. Dengan model ini yang bisa diterapkan di setiap lokasi CCTV ruangan publik, maka pihak yang berwenang bisa lebih mengontrol orang-orang yang taat dalam menjaga protokol kesehatan COVID-19 dan yang tidak.

1.3 Ruang Lingkup

Ruang lingkup untuk jurnal ini adalah sebagai berikut:

1. Definisi *Convolutional Neural Network*
2. Karakteristik *datasets*
3. *Preprocessing* data
4. Model yang digunakan & struktur model
5. Proses *training* model
6. Proses pengaplikasian model

2 Pembahasan

2.1 Definisi Convolutional Neural Network

Face mask detection merupakan studi yang dipelajari dengan menggunakan model machine learning algoritma dengan menggunakan metode klasifikasi. *Convolutional Neural Network* (CNN) yang menjadi kunci utama dalam pengklasifikasian kategori dalam melakukan *images recognition* dan *images classification [1].*

CNN image Classification mengambil kumpulan data foto yang diberikan dan kemudian memprosesnya dan mengklasifikasikannya menjadi beberapa kategori (*with mask*, *without mask*). Komputer melihat nilai dari gambar tersebut sebagai *array* dari pixel berdasarkan ukuran resolusi gambarnya. Berdasarkan dari resolusi gambar, akan terlihat nilai t x l x d (t = tinggi, l = lebar, d = dimensi). Seperti contoh sebuah gambar berukuran 6 x 6 x 6 berbentuk metriks dari saluran *RGB* (*Red, Green,* dan

Blue) dan gambar berukuran 4 x 4 x 1 berbentuk metriks yang merupakan gambar *grayscale [2].*

2.2 Sumber & Karakteristik Dataset

Dataset yang digunakan untuk membuat *model* yang dapat melakukan *face mask detection*, bersumber dari *Kaggle* dengan nama *Face Mask Detection [3]*. *Face Mask Detection Dataset* merupakan *image dataset* dengan dimensi 128 x 128 *pixel*. *Face Mask Detection Dataset* memiliki tiga kelas berbeda dengan masing-masing kelas berisi 2994 gambar. Kelas tersebut diantaranya gambar wajah dengan menggunakan masker (*with_mask*), gambar wajah yang menggunakan masker yang belum benar (*mask_weared_incorrect*), dan wajah yang tidak memakai masker (*without_mask*).

2.3 Preprocessing Data

Dengan menggunakan *validation split* sebesar 20%, kita bisa memecah *dataset* menjadi 80% kumpulan data untuk dilatih dan 20% kumpulan data untuk divalidasi. Proses *splitting dataset* dilakukan dengan menggunakan *image_dataset_from_directory function* dari *tensorflow library*.

Batch size merupakan *hyperparameter* yang menentukan jumlah sampel untuk dikerjakan sebelum memperbarui parameter model internal. *Batch size* yang digunakan adalah 32 dan pada setiap akhir batch, prediksi dibandingkan dengan *ground truth* dan dinilai dengan menggunakan *validation set*.

Adam Optimizer merupakan algoritma pengoptimalan yang dapat digunakan sebagai ganti dari prosedur *stochastic gradient descent* klasik untuk memperbarui *weight network* secara iteratif berdasarkan data training [4].

Dalam mengevaluasi performa algoritma dari *Machine Learning*, kita menggunakan acuan metriks akurasi dalam setiap proses *looping* sebuah *training* model. *tf.keras.metrics.Accuracy* akan mengkalkulasikan sebuah nilai dimana nilai tersebut akan menggambarkan seberapa miripnya hasil prediksi *training* model dengan nilai aslinya.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Accuracy} = \frac{TP}{TP + FP + TN + FN}$$

TP = *True Positive*
FP = *False Positive*
TN = *True Negative*
FN = *False Negative*

Precision adalah sebuah perbandingan antara nilai *True Positive* dengan banyaknya data yang diprediksi sebagai positif.

Recall adalah perbandingan antara *True Positive* dengan banyaknya data yang sebenarnya positif.

Akurasi adalah sebuah perbandingan antara nilai *True Positive* dengan banyaknya data yang telah diprediksi.

2.4 Model yang Digunakan & Struktur Model

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 128, 128, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization (BatchNo (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 63, 63, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 61, 61, 64)        18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 61, 61, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 128)       73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 28, 28, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3211392
_________________________________________________________________
batch_normalization_3 (Batch (None, 128)               512
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
=================================================================
Total params: 3,306,435
Trainable params: 3,305,731
Non-trainable params: 704
_________________________________________________________________
```

Struktur tersebut adalah struktur dari *Convolutional Neural Network Model* yang digunakan dalam melakukan proses *training*.

- *Rescaling Layer* berguna untuk mengubah *input layer* menjadi memiliki dimensi sebesar 128 x 128 x 3 *neuron*. [5]
- *Conv2d Layer* berfungsi untuk melakukan proses *convolutional* terhadap *input image* [6]. Model ini menggunakan tiga *conv2d layer* dimana masing-masing memiliki jumlah neuron sebanyak 32, 64 dan 128 dengan besar dimensi 3 x 3. Setiap layer *conv2d* yang ada pada model ini menggunakan fungsi aktivasi *relu*.
- *Batch_normalization Layer* berfungsi untuk melakukan transformasi terhadap nilai *mean* yang mendekati 0 dan nilai simpangan baku yang mendekati 1. [7]
- *MaxPooling2d* berfungsi untuk melakukan proses *convolutional* dengan *windows* tertentu dan mencari nilai maksimum terhadap *input window* untuk setiap *input channel* [8]. Model ini menggunakan tiga *maxpooling layer* dengan *pool size* atau *window size* sebesar 2x2.
- *Dropout Layer* berfungsi untuk membantu mencegah terjadinya *model overfitting* dengan secara acak mengubah beberapa *input units* menjadi 0 dengan frekuensi yang ditentukan pada parameter [9]. Model ini menggunakan tiga *dropout layer* dengan frekuensi 30%.
- *Flatten Layer* berfungsi untuk mengubah input dari *two dimensional array* menjadi *one dimensional array*.
- *Dense Layer* berfungsi untuk mendefinisikan *hidden*

layer serta berfungsi sebagai *fully connected layer* yang berada pada sebelum *output layer*. *Layer* ini memiliki jumlah *neuron* sebanyak 128 dan fungsi aktivasi *relu*.
- *Dense Layer* terakhir berfungsi sebagai output layer dengan jumlah *neuron* sebanyak 3 dan memiliki fungsi aktivasi *softmax*.

Selain menggunakan *convolutional neural network model* yang digunakan untuk proses training, digunakan juga *third-party DNNs* berupa *pre trained Caffee models* yang berfungsi untuk melakukan *face recognition* [10].

2.5 Proses Training Model

Dengan menggunakan arsitektur model yang sudah dibuat, proses pelatihan akan dilakukan dengan menggunakan fungsi *.fit*() dari objek model yang sudah dibuat. menggunakan parameter data *train*, *epochs* sebanyak 15, data validasi, dan *callbacks* menggunakan *Early_Stop* dengan patience = 3, dimana *callbacks function* berfungsi untuk menghentikan proses *training* ketika sebuah hasil akurasi sudah mencapai nilai konvergen minimal sebanyak tiga kali berturut-turut.

```
model.compile(loss=keras.losses.CategoricalCrosse
ntropy(from_logits=True),
optimizer='adam',
metrics=['accuracy',
tf.keras.metrics.Precision(),
tf.keras.metrics.Recall()])
```

Dengan menggunakan kode diatas, model akan menerapkan nilai-nilai yang diberikan pada parameternya ke dalam dirinya seperti menggunakan *loss CategoricalCrossentropy* dan *adam optimizer.*

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Figure 1. Categorical Entropy Formula [11]

Categorical Entropy adalah sebuah *loss function* yang digunakan untuk *multi-class classification*. Ini hanya bisa digunakan untuk mencari sebuah data yang berada pada salah satu *class* dari berbagai banyak *class* yang ada.

```
model.fit(train,epochs=15, validation_data=test,
callbacks=Early_Stop)
```

Dengan menggunakan kode diatas, model akan memulai *training process* dengan mengikuti nilai parameter yang diberikan seperti menggunakan *train dataset*, *epochs* sebanyak lima belas kali, menggunakan data validasi untuk melihat *score* setiap *epoch*, dan melakukan *callback function* berupa *early stopping*.

2.6 Proses Pengaplikasian Model

Model yang didapatkan dari proses *training*, akan dimanfaatkan untuk melakukan *classification* dengan *real-time camera* yang akan diaplikasikan dengan bantuan *OpenCV library*. Dalam melakukan *face mask classification*, *task* utama terbagi menjadi dua bagian yaitu melakukan *face recognition* dan juga *face mask classification* itu sendiri.

2.6.1 Face Recognition

Dengan melakukan video streaming dalam *OpenCV*, setiap frame akan dilakukan pendeteksian wajah. Wajah-wajah tersebut dideteksi dengan menggunakan bantuan *readNet function* dari *opencv library* dan juga model *DNN's Caffe*. Setiap wajah yang dideteksi, *bounding box* akan digambarkan dan frame tersebut akan di-*crop*.

2.6.2 Face Mask Classification

Setelah berhasil dilakukannya *face recognition*, akan dilakukan *image resizing* menjadi 128x128 pixel agar gambar tersebut memiliki dimensi yang sama dengan *input dimension* yang ada pada model. Jika dalam sebuah gambar tersebut terdapat minimal 1 atau lebih wajah yang terdeteksi, maka model akan melakukan prediksi apakah seseorang yang ada pada video tersebut memakai masker, memakai masker namun salah, atau tidak memakai masker. Untuk melakukan *face mask classification* ini, *detect_and_predict_mask function* digunakan untuk me-*return* atau mengembalikan nilai berupa hasil prediksi dari model dan juga lokasi dari wajah yang terdeteksi pada gambar tersebut.

3 HASIL

3.1 Hasil Model

Berikut merupakan hasil dari *training* pada model

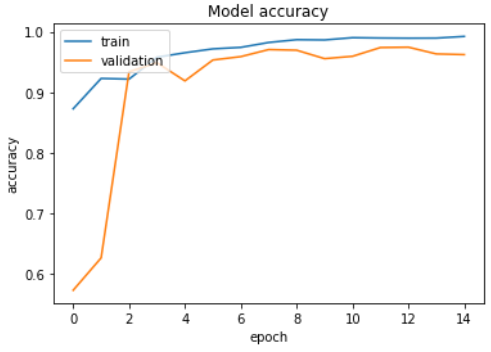

Figure 2. Accuracy Learning Curve

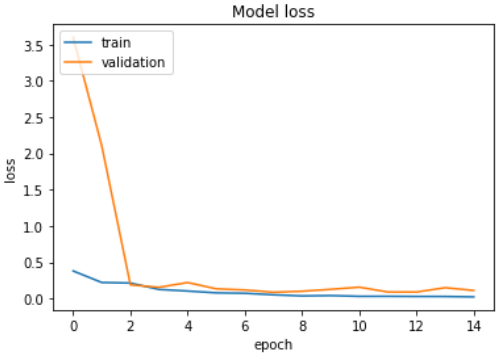

Figure 3. Loss Learning Curve

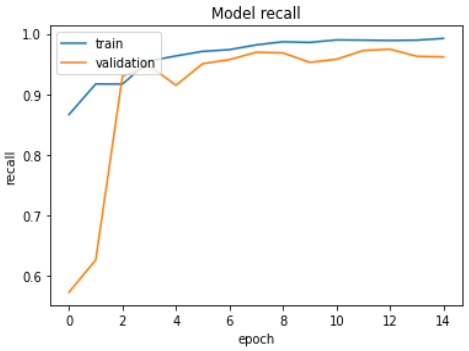

Figure 4. Recall Learning Curve

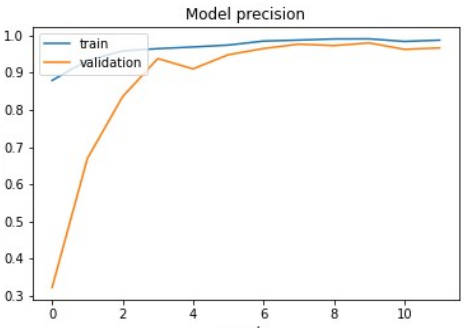

Figure 5. Precision Learning Curve

Dengan nilai *training* sebagai berikut:

- *Accuracy Score*: 0.9922
- *Loss Score*: 0.0240
- *Recall Score*: 0.9922
- *Precision Score*: 0.9925

Dan nilai *validation* seperti berikut:

- *Validation Accuracy*: 0.9621
- *Validation Loss Score*: 0.1122
- *Validation Recall Score*: 0.9615
- *Validation Precision Score*: 0.9626

3.2 Hasil Implementasi

Figure 6. Final Result

Berikut adalah hasil prediksi dari model yang sudah berhasil dibuat berdasarkan *study* paper ini. Model tersebut berhasil memprediksi setiap kondisi kemungkinan yang terjadi pada wajah seseorang.

4 Konklusi

Kesimpulannya, *study* ini berhasil membuat sebuah model yang sudah berhasil mendeteksi sebuah wajah manusia secara langsung dengan menggunakan webcam dari komputer dan berhasil memprediksi apakah wajah tersebut sedang menggunakan masker secara benar, salah, atau tidak menggunakan masker sama sekali. Dengan menggunakan sebuah kelebihan dari model *machine learning*, model ini bisa membantu pihak yang berwenang dalam menghambat penyebaran COVID-19.

Referensi

[1] Sanjaya, S. A., & Rakhmawan, S. A. (2020, October). Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic. In 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI) (pp. 1-5). IEEE.

[2] Lubis, R. MACHINE LEARNING (CONVOLUTIONAL NEURAL NETWORKS) FOR FACE MASK DETECTION IN IMAGE AND VIDEO.

[3] Kumar, V. (2021, May 19). Face mask detection. Kaggle. Retrieved January 8, 2022, from https://www.kaggle.com/vijaykumar1799/face-mask-detection

[4] *Keras: Introduction to the adam optimization algorithm*. Keras: Introduction to the Adam Optimization Algorithm - OnnoWiki. (n.d.). Retrieved January 8, 2022, from https://lms.onnocenter.or.id/wiki/index.php/Keras:_Introduction_to_the_Adam_Optimization_Algorithm#:~:text=8%20Pranala%20Menarik-,Apakah%20Adam%20optimization%20algorithm%3F,secara%20iteratif%20berdasarkan%20data%20training

[5] Tf.keras.layers.rescaling :   Tensorflow core v2.7.0. TensorFlow. (n.d.). Retrieved January 8, 2022, from https://www.tensorflow.org/api_docs/python/tf/keras/layers/Rescaling

[6] Team, K. (n.d.). *Keras Documentation: Conv2d layer*. Keras. Retrieved January 8, 2022, from https://keras.io/api/layers/convolution_layers/convolution2d/

[7] Team, K. (n.d.). *Keras Documentation: Batchnormalization layer*. Keras. Retrieved January 8, 2022, from https://keras.io/api/layers/normalization_layers/batch_normalization/

[8] Team, K. (n.d.). *Keras Documentation: MaxPooling2D layer*. Keras. Retrieved January 8, 2022, from https://keras.io/api/layers/pooling_layers/max_pooling2d/

[9] Team, K. (n.d.). *Keras documentation: Dropout layer*. Keras. Retrieved January 8, 2022, from https://keras.io/api/layers/regularization_layers/dropout/#:~:text=The%20Dropout%20layer%20randomly%20sets,time%2C%20which%20helps%20prevent%20overfitting.&text=Note%20that%20the%20Dropout%20layer,When%20using%20model.

[10] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia (pp. 675-678).