ISSN 2355-3286

# Emotion Recognition using Convolutional Neural Network on Virtual Meeting Image

Julando Omar[1], Nabila Husna Shabrina[2], Akmal Nusa Bhakti[3], Axel Patria[4]
[1, 2, 3, 4] Teknik Komputer, Universitas Multimedia Nusantara, Tangerang, Indonesia
[1]julando.omar@student.umn.ac.id, [2]nabila.husna@umn.ac.id, [3]akmal.nusa@student.umn.ac.id,
[4]axel.patria@student.umn.ac.id

Diterima 10 Juni 2021
Disetujui 24 Juni 2021

***Abstract*— This study is conducted to propose an Emotion recognition system on virtual meeting image using Convolutional Neural Network. Our system consists of 2 phases, the training phase, to train the Convolutional Neural network model, and the classification phase, to classify the images in Convolutional Neural network into seven different emotion. Haar Cascade Classifier method was employed in this work. The training was carried out using the FER-2013 dataset with 430 epoch. The proposed model gave 73,56% accuracy in classifying participant's emotion in Virtual Meeting Image.**

***Index Terms*—Convolutional Neural Network, Emotion Recognition, Face Recognition, FER-2013, Virtual Meeting**

## I. PENDAHULUAN

Dalam satu tahun terakhir, dunia edukasi mengalami perubahan drastis dalam cara kita berkomunikasi. Akibat dari pandemi yang muncul, memaksa kita untuk merubah cara kita berkomunikasi dari tatap muka menjadi secara daring. Dari percakapan sederhana antara dua orang, proses belajar mengajar, hingga sebuah meeting penting yang dilakukan oleh sebuah perusahaan, semua terpaksa dilakukan secara daring. Menurut penelitian yang dilakukan oleh ISED, pada bulan April 2020, hampir 88% responden melakukan work from home, dan melakukan meeting dengan menggunakan aplikasi virtual meeting seperti zoom meeting, google hangout dan webex. [1]

Sistem komunikasi daring sendiri tidak terlepas dari beberapa kekurangan, salah satunya adalah feedback yang diberikan saat berkomunikasi. Feedback verbal dan nonverbal yang biasanya dapat dilakukan saat berkomunikasi tatap muka mengalami hambatan dalam berkomunikasi daring. Ekspresi wajah terutama dapat menjadi komunikasi untuk mengutarakan perasaan kenyamanan, persetujuan maupun simpati kepada lawan bicara, tanpa harus adanya informasi tambahan dari anggota tubuh lainnya. [2]

Untuk mengatasi permasalahan ini, kami membuat sebuah Emotion Recognition yang menggunakan sebuah Convolutional Neural Network (CNN) pada sebuah gambar Virtual Meeting, dengan tujuan untuk membantu memahami feedback nonverbal yang diberikan melalui ekspresi dari peserta dari Virtual meeting. Cara ini dilakukan dengan melakukan dua tahapan *training* dan classification. Tahapan *training* merupakan tahapan dimana model CNN akan dipaparkan terhadap dataset FER-2013 yang berisikan berbagai ekspresi dari senang, sedih, marah, jijik, takut, kaget dan neutral. CNN dipilih karena memiliki akurasi yang lebih baik dalam melakukan emotion recognition daripada metode lain, seperti SVM. Sebagai contoh, metode SVM dengan PCA berhasil untuk melakukan emotion recognition dengan dataset CK+ dengan akurasi sebesar 81% [3], sedangkan metode CNN berhasil untuk melakukan emotion recognition pada dataset yang sama dengan akurasi sebesar 92.81% [4]

Model CNN pada penelitian ini menggunakan optimizer Adam, dan loss function berupa sparse categorical loss. Optimizer Adam dipilih karena Adam memiliki performa yang lebih baik daripada SGD ataupun SGD dengan momentum dalam melakukan klasifikasi[5], sedangkan sparse categorical loss dipilih karena label dari dataset merupakan dalam bentuk angka sehingga diperlukan sparse loss function kemudian hasil dari CNN adalah multi-class sehingga dibutuhkan categorical loss function dalam melakukan klasifikasi.

Kemudian model akan melakukan klasifikasi pada tahapan classification, dimana model akan diberikan input berupa gambar dari Virtual Meeting dan akan mengklasifikasikan berbagai emosi dari setiap wajah yang diekstrak menggunakan algoritma haar-cascade classifier peserta yang mengikuti Virtual Meeting tersebut. Haar-cascade classifier pada penelitian ini digunakan karena Haar-cascade classifier, merupakan salah satu algoritma face recognition yang paling banyak digunakan, selain itu algoritma ini juga sederhana dan efektif dalam melakukan face-recognition[6].

## II. KAJIAN LITERATUR

Studi mengenai implementasi dari CNN untuk mengetahui ekspresi muka manusia sudah banyak dilakukan. Tabel 1 merupakan beberapa studi yang telah dilakukan dalam mengimplementasi CNN yang digunakan didalam *Emotion Recognition.*

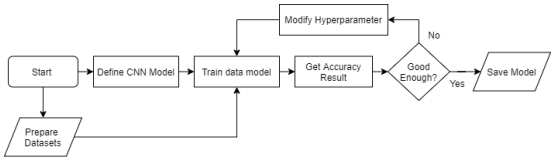
Tabel 1. Kajian Literatur

| Literatur | Akurasi | Kelebihan | Kelemahan |
|---|---|---|---|
| [7] | 70% | Model berhasil mencapai 70% akurasi | Model masih sulit membedakan ekspresi takut dengan ekspresi sedih. |
| [8] | 73.4% | Model mencapai akurasi 73% | Karena menggunakan sift, sehingga akan lebih lambat dalam melakukan prediksi emosi |
| [9] | 85% | Dapat memprediksi emosi dengan akurasi sebesar 85% | Dataset JAFFE sendiri hanya memiliki muka perempuan, sehingga akan menjadi bias saat dilakukan dengan menggunakan data real |
| [10] | 69% | Model SHCNN merupakan model yang cukup sederhana namun dapat mencapai akurasi sebesar 69% | Akurasi yang didapatkan dari Model tersebut masih cukup rendah dibandingkan dengan model yang lainnya. |
| [11] | 61%, 41%, 64% dan 65% | Model CNN merupakan model yang sudah banyak dipakai dan dapat digunakan dengan menggunakan library-library yang ada. | Akurasi yang didapatkan dari Model tersebut masih cukup rendah dibandingkan dengan model yang lainnya.. |
| [12] | 69.40% | Model yang digunakan merupakan modifikasi dari Model VCG-16 yang merupakan model CNN yang dapat digunakan dengan menggunakan library yang ada. | Akurasi yang didapatkan dari Model tersebut masih cukup rendah dibandingkan dengan model yang lainnya. |

Dari studi tersebut, maka dapat disimpulkan bahwa penggunaaan CNN untuk *emotion recognition* mempunyai akurasi rata-rata diantara 65%-70%, dengan studi [9] memiliki akurasi paling tinggi sebesar 85% dengan menggunakan dataset JAFFE dan studi [8] dengan menggunakan dataset FER-2013, yang merupakan dataset yang digunakan pada penelitian ini.

## III. METODOLOGI

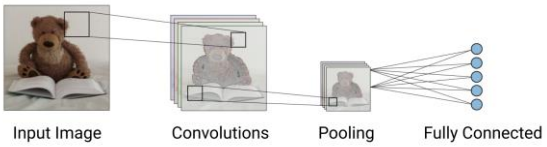
Dalam penelitian ini, metode akan dibagi menjadi 2 bagian yaitu bagian *training* dan *classification.*




Gambar 1. *flowchart training*

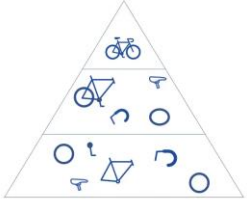
### A. *Training*

Bagian *training* merupakan bagian dimana CNN dilakukan *training*. CNN merupakan *sub-architecture* dari *Neural Network* yang umumnya terdiri atas *convolutional layer*, *pooling layer* dan *fully connected layer*. [13]




Gambar 2. CNN [14]

*Convolutional layer* merupakan *layer* yang menggunakan *filters* untuk melakukan konvolusi kedalam gambar *input.* Hasil dari *convolutional layer* adalah *feature map*. *Non-linearity* di dalam model didapatkan dari penggunaan *ReLu* pada akhir setiap *convolutional layer. Convolutional* layer lainnya juga dapat mengikuti *convolutional layer* sebelumnya, dimana jika hal tersebut dilakukan maka akan dibentuk sebuah *feature hierarchy.* [15]



Gambar 3. *Feature Hierarchy* [16]

*Layer* selanjutnya yang terdapat dalam CNN adalah *pooling layer*. *Layer* ini berfungsi untuk mengatasi dimensionalitas dari sebuah *feature map*. *Pooling layer* memiliki cara kerja yang mirip dengan *convolutional layer* namun setiap *filter* yang terdapat pada *pooling layer* tidak memiliki *weights*. *Pooling layer* berguna untuk mengurangi kompleksitas, meningkatkan efisiensi dan mengurangi *overfitting*. [15]

*Layer* terakhir yang terdapat pada CNN adalah *fully connected layer*. *Fully-connected layer* berfungsi untuk melakukan klasifikasi terhadap CNN yang ada. *Fully-connected* layer menggunakan fungsi *softmax* untuk menentukan hasil keluaran yang ada. [15] Arsitektur CNN pada metode dapat dilihat pada tabel 2.

Tabel 2. Arsitektur CNN

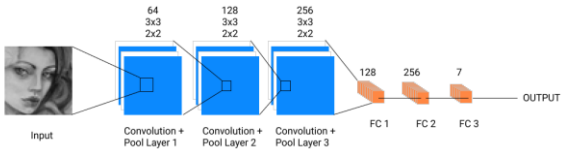
| ***Layer*** | **Ukuran** | **Jumlah *filters*** |
|---|---|---|
| *Input Layer* | 48 x 48 | - |
| *Convolution 1* | 3 x 3 | 64 |
| *Max Pool 1* | 2 x 2 | - |
| *Convolution 2* | 3 x 3 | 128 |
| *Max Pool 2* | 2 x 2 | - |
| *Convolution 3* | 3 x 3 | 256 |
| *Max Pool 3* | 2 x 2 | - |
| *Fully Connected 1* | - | 128 |
| *Fully Connected 2* | - | 256 |
| *Fully Connected 3* | - | 7 |

Agar tidak terjadi *overfitting* maka *Batch Normalization layer* dan *Dropout layer* digunakan di antara *layer-layer* yang terdapat di dalam arsitektur CNN.

*Batch Normalization* adalah teknik yang digunakan untuk standarisasi *input* dari setiap *mini-batch*. *Batch Normalization* memiliki efek untuk menstabilkan proses *learning* dan mengurangi jumlah *training epoch* untuk men-*train* sebuah *neural Network*. [17]

*Dropout* merupakan salah satu teknik *regularization* pada *neural network*. Pada teknik *Dropout*, ketika *training* dijalankan maka beberapa *layer output* akan dibiarkan atau di-*dropped out*. *Dropout* memiliki efek untuk membuat *training* menjadi sedikit *noisy* sehingga membuat *nodes* pada sebuah layer untuk memiliki responsibilitas terhadap suatu input. [18]

Model CNN pada penelitian ini menggunakan optimizer adam, dan loss function berupa *sparse categorical loss*.




Gambar 4. Visualisasi Arsitektur CNN

Dataset yang digunakan untuk *training* pada penelitian ini adalah *Open Database* FER-2013 [19]. Basis data berisi gambar dengan ukuran 48x48 piksel, yang telah dikategorikan sesuai dengan ekspresi yang ditampilkan. Basis data memiliki 7 jenis ekspresi wajah, yaitu marah, jijik, takut, gembira, sedih, terkejut, dan netral, yang terbagi menjadi dua kategori, yaitu kumpulan data untuk pelatihan dan kumpulan data untuk pengujian. Ada total 28709 gambar untuk pelatihan, dan total 3589 gambar untuk pengujian, total 32.298 gambar untuk diproses.

Sebelumnya, FER-2013 adalah kumpulan data sumber terbuka yang pertama, dibuat untuk proyek yang sedang berjalan oleh Pierre-Luc Carrier dan Aaron Courville, kemudian dibagikan secara publik untuk kompetisi Kaggle, tak lama sebelum ICML 2013.

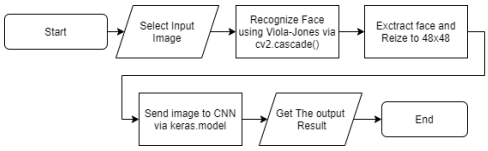
Pada bagian *Training* kemudian dibagi menjadi beberapa bagian, yang antara lain adalah:

- *Define CNN model*
  Pada tahapan ini arsitektur dari model CNN dibentuk dengan menggunakan library *Tensorflow* [20] dan *Keras* [21] dalam bahasa pemrograman *Python*.
  *Prepare Dataset*

  Untuk dapat melakukan *training* pada CNN maka dibutuhkan *dataset*. *Dataset* yang digunakan adalah dataset FER-2013 yang telah dilakukan normalisasi distribusi agar model CNN yang ada tidak terjadi *overfitting*. Normalisasi distribusi dilakukan dengan cara metode *Oversampler*. Kemudian dataset *training* kemudian dibagi menjadi 75% data *training* dan 25% data validasi.
- *Train Data Model*
  Pada tahapan ini, model yang sebelumnya telah dibentuk akan dilakukan *training*. *Training* dilakukan dengan *dataset* berupa *dataset* yang sebelumnya sudah disiapkan. Pada tahapan ini juga dilakukan konfigurasi *hyperparameter* seperti *epoch*, *batch size*, dan *step* yang digunakan dalam *training*.

- *Get Accuracy Result*
  Pada tahapan ini, dilakukan evaluasi terhadap model CNN pada metrics akurasi, jika metrics akurasi dirasa sudah mencukupi maka *training* selesai dan model akan disimpan untuk digunakan pada tahapan selanjutnya, jika akurasi dari model dirasa tidak mencukupi, maka dapat dilakukan modifikasi terhadap *hyperparameter* model agar didapatkan akurasi yang lebih baik.

*B. Classification*

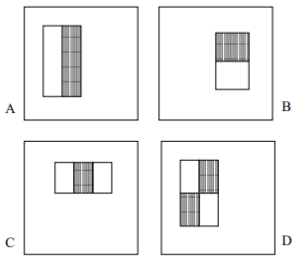



Gambar 5. *flowchart Classification*

Tahapan *classification* adalah tahapan dimana user dapat mengklasifikasikan dan mendeteksi ekspresi wajah manusia. Hal pertama yang dilakukan untuk mendeteksi ekspresi wajah manusia adalah dengan melakukan deteksi wajah manusia. Algoritma pendeteksi wajah yang kami gunakan adalah *Haar-cascade classifier*.

Dalam melakukan *face recognition* algoritma *Haar-cascade Classifier* melakukan beberapa langkah, antara lain sebagai berikut [22]:

- Memilih *Haar-like feature*
  Haar-like feature adalah kumpulan fungsi berbentuk persegi, hasil dari *haar-like features*, merupakan angka hasil proses dari fungsi tersebut. *Haar-like features* ini akan digunakan untuk mendeteksi apakah adanya *lines*, *edges* dan juga *diagonal* di dalam *image*, yang membantu untuk melakukan pemisahan *feature* dalam wajah. Dalam algoritma *viola-jones* terdapat 3 *haar-like features*, yaitu *edg*e, *lines* dan *diagonal*.




Gambar 6. *Haar-like features* [22]

Berdasarkan gambar diatas, maka gambar A digunakan untuk mendeteksi *edges* dari wajah, sedangkan gambar B dan gambar C digunakan untuk mendeteksi *lines* wajah pada *subregion* gambar, sedangkan gambar D digunakan untuk mendeteksi *line* diagonal. *Haar like feature* ini kemudian akan dibagi menjadi *sub-window* dan akan di iterasi ke seluruh gambar untuk menghasilkan *integral image*.

- Membuat *Integral image*
  Integral *image* merupakan sebuah struktur data yang digunakan untuk mempercepat dalam melakukan perhitungan jumlah nilai dari suatu pixel pada suatu gambar. Setiap pixel dari *integral image*, merupakan hasil penjumlahan dari pixel diatas pixel tersebut, pixel yang terdapat pada bagian kiri pixel tersebut dan juga nilai dari pixel itu sendiri. Untuk menghitung suatu pixel pada gambar kita dapat menggunakan 4 titik untuk perhitungan. Perhitungan keempat titik ini memudahkan dalam perhitungan *feature* yang didapat dari *haar-like features*. Dengan menggabungkan *haar like feature* dengan *integral image*, kita dapat dengan cepat melakukan perhitungan nilai dari *haar-like feature* tersebut.

- Menjalankan *Adaboost Training*
  Adaboost merupakan salah satu algoritma machine learning dimana *classifier* dari algoritma ini merupakan kumpulan-kumpulan dari *weak learner* yang menghasilkan sebuah *strong classifier*. Dalam algoritma viola-jones, *weak learner* yang ada adalah *feature-feature* yang dihasilkan oleh *haar like feature*, dimana satu fitur merupakan satu *weak classifier*. Dalam melakukan *face recognition*, *adaboost* melakukan klasifikasi kedalam semua *subregion* dari gambar dan mengecek apakah *classifier* yang ada memberikan nilai yang positif atau negatif, jika nilai positif, maka pada gambar tersebut terdapat wajah, sedangkan jika nilai negatif maka tidak terdapat wajah pada *sub-region* tersebut.

- Membuat *Cascading classifier*.
  *Cascading classifier* digunakan untuk mengklasifikasi kembali, bagian dari gambar mana yang memiliki wajah dan mana yang tidak merupakan wajah. *Cascading classifier* digunakan karena hasil dari klasifikasi adaboost masih memiliki jumlah yang cukup banyak, yaitu hampir 8000 *features* untuk gambar berukuran 24 x 24, dengan *feature* yang cukup banyak ini, akan membutuhkan waktu yang cukup lama untuk melakukan *face recognition*, oleh sebab itu *cascading classifier* dibutuhkan. *Cascading classifier* bekerja dengan cara membagi menjadi beberapa *stage*, dimana *stage* pertama merupakan *feature* yang dianggap paling bagus untuk merepresentasikan wajah, *stage* kedua merupakan *feature* yang lain untuk diklasifikasikan. Dengan menggunakan *cascading classifier* ini, ketika *subregion* gambar input yang dimasukan kedalam *classifier stage* pertama menghasilkan hasil positif, maka *subregion* gambar tersebut akan diklasifikasikan lagi kedalam *stage* kedua, namun jika *subregion* gambar tersebut memiliki nilai yang negatif, maka gambar tersebut akan dibuang dan

akan dicari *subregion* gambar tersebut yang memiliki wajah.

Dalam penelitian ini, *Haar-cascade classifier* diimplementasikan menggunakan library *openCV*. library openCV sendiri merupakan library untuk melakukan pengolahan citra yang terdapat pada bahasa pemrograman *python.* Metode OpenCV yang digunakan adalah cv2.cascade() dimana metode tersebut menerima input berupa file .xml yang berisi hasil *training* untuk mendeteksi wajah.

Tahapan pada tahap klasifikasi dibagi menjadi beberapa bagian, bagian tersebut antara lain adalah:

- *Select Input Image*
  Pada tahapan ini, *user* dapat melakukan input image yang berupa gambar wajah manusia ataupun gambar wajah pada *virtual meeting* secara manual yang akan dilakukan pengenalan emosi. *Input image* diterima menggunakan metode *OpenCV imread*.
- *Recognize Face*
  Setelah gambar diterima, maka pada gambar tersebut akan dikenali mana yang merupakan wajah manusia dan mana yang bukan. pengenalan ini menggunakan algoritma *haar-cascade classifier* yang telah dijelaskan sebelumnya.
- *Extract Face and Resize to 48x48*
  Setelah wajah terdeteksi, maka akan dilakukan ekstraksi wajah tersebut untuk diteruskan sebagai input pada CNN. Karena input pada CNN memiliki besaran berupa 48x48 pixel, maka gambar wajah yang telah terdeteksi tersebut harus dilakukan perubahan ukuran sehingga gambar menjadi ukuran 48 x 48 pixel.
- *Sent Image to Model via Keras.model*
  Setelah wajah terdeteksi dan diubah menjadi ukuran 48x48 pixel, maka gambar wajah tersebut diberikan menjadi *input* dalam model CNN menggunakan keras.predict. Keras.predict digunakan untuk melakukan klasifikasi oleh CNN untuk menghasilkan ekspresi yang sesuai dengan ekspresi yang ada.
- *Get The Output Result*
  Setelah CNN berhasil mengklasifikasikan wajah yang ada, maka ekspresi dapat ditampilkan melalui metode *OpenCV imshow*. Wajah manusia yang terdeteksi akan dikelilingi oleh persegi serta teks yang menampilkan ekspresi wajah yang diprediksi.

## IV. HASIL

Pada penelitian ini, model CNN ditraining menggunakan *library tensorflow-gpu* versi 2.3.0. Dataset dari *training* yang digunakan adalah dataset FER-2013 yang didapatkan dari *kaggle*[19]. Dataset FER-2013 ini berbentuk file csv, sehingga diperlukan *pre-processing* data menggunakan *library pandas. File csv* yang dibaca merupakan file csv dengan nama 'train.csv'. Pada data yang telah dibaca, maka akan dilakukan normalisasi terhadap distribusi setiap gambar dari ekspresi wajah dengan menggunakan fungsi RandomOverSampler dari *library imblearn*. Hasil dari normalisasi distribusi adalah setiap ekspresi memiliki gambar sebanyak 7215 gambar. Kemudian dari hasil distribusi tersebut, dataset dilakukan pembagian dimana 80% data akan digunakan sebagai data *training* dan 20% akan digunakan sebagai *data test*.

Model CNN dibuat pada aplikasi web *Google Colab* yang menggunakan *Google Compute Backend GPU* dengan *GPU* berupa A100, RAM sebesar 12GB dan *disk storage* sebesar 45GB. *Google Colab* merupakan aplikasi web yang dapat menjalankan fail *jupyter notebook* yang dapat menjalankan bahasa pemrograman python.

Model CNN kemudian dibuat menggunakan library *keras* dan *tensorflow* sesuai dengan arsitektur yang diajukan dalam penelitian ini. Hasil dari model CNN pada *tensorflow* dan *keras* antara lain adalah sebagai berikut:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 48, 48, 64)        640
_________________________________________________________________
batch_normalization (BatchNo (None, 48, 48, 64)        256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 24, 24, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 128)       73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_3 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 256)       295168
_________________________________________________________________
batch_normalization_4 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 128)               1179776
_________________________________________________________________
batch_normalization_5 (Batch (None, 128)               512
_________________________________________________________________
activation (Activation)      (None, 128)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               33024
_________________________________________________________________
batch_normalization_6 (Batch (None, 256)               1024
_________________________________________________________________
activation_1 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 1799
=================================================================
Total params: 1,588,359
Trainable params: 1,586,311
Non-trainable params: 2,048
_________________________________________________________________
```

Gambar 7. Hasil dari model.summary()

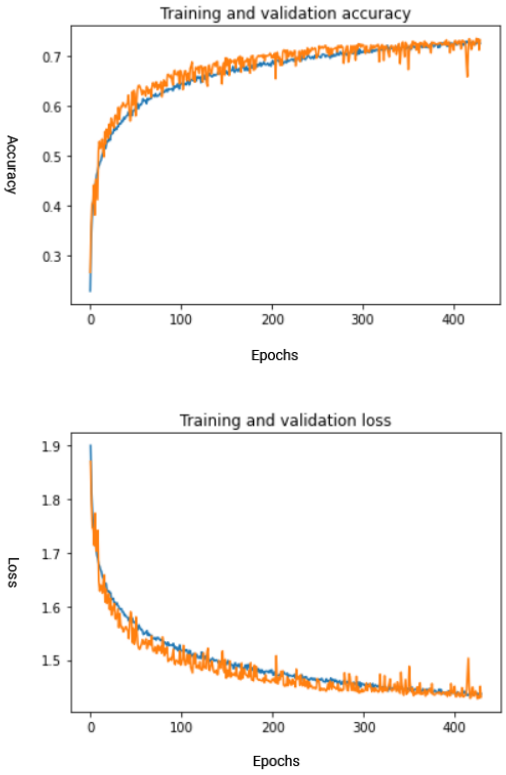
Model kemudian di-*compile* dengan *parameter* berupa *optimizer* yang menggunakan Adam, dan *loss* yang menggunakan fungsi *sparse categorical cross entropy.*

Model kemudian dilakukan training dengan menggunakan *hyperparameter* berupa *batch-size*

sebesar 64, *steps* dengan rumus jumlah data yang terdapat pada training dibagi dengan 128, *validation split* 0.25 serta *epoch* sebanyak 430. Model juga diberikan fungsi *callback* agar model yang tersimpan adalah model yang memiliki *validation loss* paling sedikit.

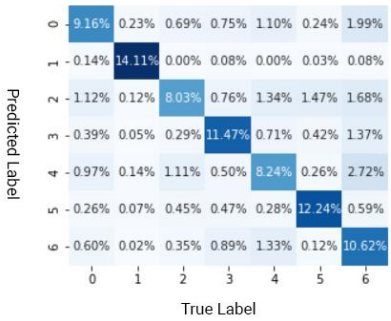
Waktu yang dibutuhkan untuk melakukan *training* pada model CNN adalah selama 29 menit, dimana waktu yang dibutuhkan per *epoch* adalah selama 4 detik.

Hasil CNN yang di-*training* sebanyak 430 *epochs* memiliki akurasi sebesar 72.55% pada *training accuracy* dan akurasi sebesar 73.24% pada *validation accuracy*. Model yang disimpan melalui fungsi *callback* memiliki akurasi sebesar 73,15% pada *training accuracy* dan akurasi sebesar 73,56% pada *validation accuracy*. Hasil grafik kurva dari *accuracy* dan *loss* dari *training* dan *validation* loss dapat dilihat pada gambar 8.



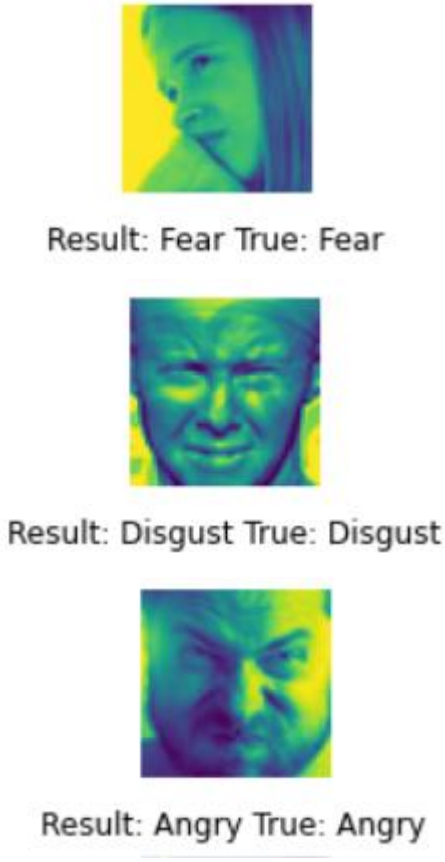

Gambar 8. Grafik Kurva *Training* dan *validation accuracy* dan *loss*, validasi memiliki kurva berwarna jingga dan *training* memiliki kurva berwarna biru

Hasil dari model yang telah di training ketika dijalankan dengan dataset test mencapai akurasi sebesar 73.86% dengan *loss sebesar 1.42*. Hasil *confusion matrix* dari model CNN dengan dataset test dapat dilihat pada gambar 9.




Gambar 9. Hasil *confusion matrix*. Sumbu X menunjukkan hasil asli dari test dan sumbu Y merupakan hasil prediksi model

Pada *confusion matrix*, *label* 0 - 6 mendandakan setiap ekspresi yang ada dimana label 0 merupakan ekspresi senang, label 1 merupakan ekspresi jijik, label 2 merupakan ekspresi takut, label 3 merupakan ekspresi senang, label 4 merupakan ekspresi sedih, label 5 merupakan ekspresi kaget dan label 6 merupakan ekspresi netral. Contoh hasil model CNN pada dataset test dapat dilihat pada gambar 10.




Gambar 10. Visualisasi Hasil dari test

Kemudian, pada bagian *classification*, peneliti menggunakan library *OpenCV* untuk membaca file gambar, melakukan *preprocessing*, berupa *cropping*, *merubah gambar menjadi grayscale* serta menaruh teks pada gambar.

Bagian *classification* dijalankan pada *jupyter notebook* yang menjalankan *tensorflow* versi 2.4.1 serta berjalan pada perangkat keras dengan spesifikasi sebagai berikut:

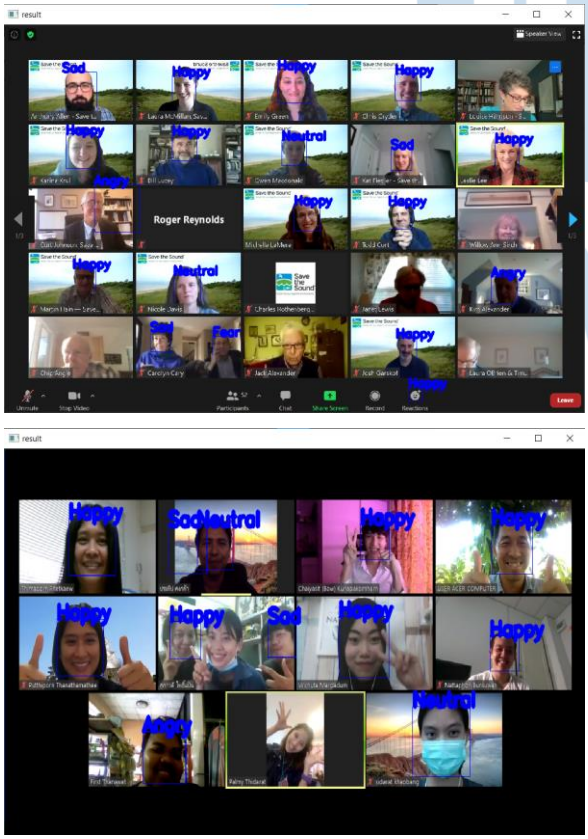
CPU : Intel i7-8750H

GPU: Nvidia GTX1050 4GB

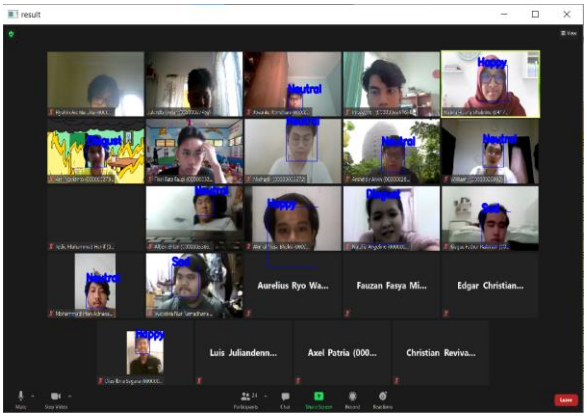
RAM: 8 GB

kemudian untuk melakukan *face recognition* digunakan fungsi cv2.cascadeClassifier dan cv2.detectMultiScale. *Cascade classifier* yang digunakan didapatkan dari *Github OpenCV* dengan nama 'haarcascade_frontalface_default.xml'. Model CNN dimasukan dengan cara menggunakan fungsi keras.model.load_model dengan parameter model yang telah di *training* sebelumnya. kemudian untuk melakukan pengenalan ekspresi maka digunakan fungsi model.predict yang menerima hasil gambar 48x48 wajah dari gambar yang tadi telah dimasukan.

Hasil dari pengenalan ekspresi muka manusia dapat dilihat pada gambar dengan menggunakan data screenshot dari *virtual meeting environment* yang berupa *zoom meeting* yang diambil dari internet serta hasil yang *zoom meeting* yang didapatkan dari dokumentasi pribadi peneliti.




Gambar 11. Pengenalan ekspresi wajah manusia pada *virtual meeting environment* yang diambil dari internet [23][24]




Gambar 12. Pengenalan ekspresi wajah manusia pada *zoom meeting*

## V. ANALISIS

Berdasarkan dari hasil yang didapatkan, maka dapat disimpulkan bahwa model berhasil untuk melakukan klasifikasi ekspresi wajah manusia. Hal ini didukung dengan akurasi sebesar 73%. Grafik kurva dari *training* dan *validation* juga menunjukan bahwa model CNN yang terdapat pada penelitian tidak terjadi *overfitting.*

*Epoch* sebesar 430 dipilih karena jika epoch melebihi 450 maka akan terjadi *overfitting* dan kemudian jika *epoch* lebih kecil dari 430, maka akurasi yang didapatkan akan menjadi tidak akurat. *Batch size* sebesar 64 digunakan sebagai *batch size* yang cukup untuk *training* dengan GPU dan agar tidak terjadi *convergence* yang lebih cepat sehingga menghindari terjadinya *overfitting. Epoch* 430 juga dipilih karena dengan menambah *epoch*, maka *training time* yang dibutuhkan juga akan menjadi lebih lama.

Pada *confusion matrix* model dapat dilihat bahwa 3 nilai *error* terbesar terdapat value *neutral* dengan *sad*, *neutral* dengan *happy* dan *neutral* dengan *fear*. Hal ini dapat terjadi karena dataset FER2013 memiliki gambar yang cukup mirip antara ekspresi *sad*, *neutral* dan *happy*. Sehingga ketika melakukan klasifikasi maka akan terjadi kesalahan dalam pengenalan ekspresi tersebut.

Pada percobaan dengan gambar *screenshot*, dapat dilihat bahwa terdapat beberapa wajah yang tidak terdeteksi. Hal tersebut dapat terjadi karena algoritma *haar-cascade* tidak dapat mendeteksi ketika pencahayaan kurang, serta ketika wajah orang tidak berupa wajah *front-frontal.*

Pada *virtual meeting environment* juga terdapat kelemahan, seperti tipe kamera yang berbeda-beda setiap orang sehingga resolusi dan pencahayaan yang dihasilkan juga cukup buruk untuk algoritma *haar-cascade* untuk melakukan *face-recognition*. Kecepatan Internet juga berpengaruh terhadap resolusi gambar dari wajah manusia yang mengikuti *virtual meeting*

sehingga menyebabkan dapat terjadinya kesalahan pada saat pengenalan ekspresi dilakukan.

## VI. SIMPULAN

Dapat disimpulkan bahwa penelitian berhasil dilakukan. Model CNN yang di training mencapai akurasi sebesar 73.56%. Selain itu, sistem juga berhasil untuk melakukan pengenalan berbagai ekspresi wajah partisipan terhadap *input* gambar yang berupa gambar *screenshot* dari *virtual meeting zoom.*

Salah satu kelebihan dari penelitian ini adalah sistem kami dapat melakukan pengenalan ekspresi wajah partisipan dalam *virtual meeting* pada hampir semua wajah partisipan yang menyalakan kamera pada saat *screenshot* diambil, Berbeda dengan studi-studi sebelumnya, yang hanya dapat melakukan pengenalan ekspresi wajah hanya pada satu wajah. Selain itu, pada model yang kami *training*, juga dapat membedakan setiap ekspresi wajah dengan baik, dimana *error* yang terjadi adalah paling besar pada *confusion matrix* adalah sebesar 2.72% yang terjadi pada label netral dengan sad. Model CNN yang kami gunakan juga menggunakan 1.5 juta parameter, sehingga waktu *training* yang dibutuhkan menjadi lebih singkat yaitu 29 menit, daripada waktu *training* yang dibutuhkan oleh studi-studi lainnya yang menggunakan parameter yang melebihi dari 1.5 juta parameter.

Salah satu limitasi yang dari penelitian ini adalah model tidak dapat selalu memprediksi setiap peserta yang terdapat di dalam *Virtual Meeting*, hal tersebut dapat terjadi karena wajah dari peserta tidak memiliki pencahayaan yang cukup. Selain itu, kelemahan berikutnya adalah sistem yang peneliti rancang belum dapat melakukan klasifikasi secara *real-time.*

Diharapkan melalui pendeteksian wajah ini, pendeteksi wajah ini dapat digunakan untuk mengetahui kepuasan seseorang dalam berkomunikasi, membantu menentukan karakteristik seseorang, dan mendapatkan masukan nonverbal dari lawan komunikasi yang kita ajak bicara.

Hasil dari penelitian ini diharapkan dapat direalisasikan agar dapat digunakan umpan balik dalam sistem pembelajaran atau pembelajaran online untuk mengetahui tingkat kepuasan dan konsentrasi mahasiswa atau mahasiswa dalam perkuliahan atau ruang kelas online, sebagai sistem umpan balik ketika melakukan tes pasar sistematis dan untuk membantu menemukan masalah psikologis seseorang. Selain itu, peneliti juga berharap, agar sistem di kemudian hari dapat melakukan klasifikasi secara *realtime* pada *virtual meeting enviroment.*

## DAFTAR PUSTAKA


[1] U. F. R. Rahmawaty and E. M. Lokollo, "WORK FROM HOME," ISED, 2020.

[2] J. A. Devito, in Interpersonal communication book, global edition, Pearson Education Limited, 2015, p. 139.

[3] J. J. Pao, "Emotion Detection through Facial Feature Recognition," p. 6, 2018

[4] D. Y. Liliana, "Emotion recognition from facial expression using deep convolutional neural network," *Journal of Physics: Conference Series*, vol. 1193, p. 012004, 2019.

[5] D. Soydaner, "A Comparison of Optimization Algorithms for Deep Learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 13, p. 2052013, 2020.

[6] L. Cuimei, Q. Zhiliang, J. Nan, and W. Jianhua, "Human face detection algorithm via Haar cascade classifier combined with three additional classifiers," 2017 *13th IEEE International Conference on Electronic Measurement & Instruments* (ICEMI), 2017.

[7] I. Lasri, A. R. Solh, and M. E. Belkacemi, "Facial Emotion Recognition of Students using Convolutional Neural Network," 2019 *Third International Conference on Intelligent Computing in Data Sciences* (ICDS), 2019.

[8] T. Connie, M. Al-Shabi, W. P. Cheah, and M. Goh, "Facial Expression Recognition Using a Hybrid CNN–SIFT Aggregator," *Lecture Notes in Computer Science*, pp. 139–149, 2017.

[9] D. Yang, A. Alsadoon, P. W. C. Prasad, A. K. Singh,and A. Elchouemi, "An Emotion Recognition Model Based on Facial Recognition in Virtual Learning Environment," *Procedia Computer Science*, vol. 125, no. 2009, pp. 2–10,2018, doi:10.1016/j.procs.2017.12.003.

[10] S. Miao, H. Xu, Z. Han, and Y. Zhu, "Recognizing Facial Expressions Using a Shallow Convolutional Neural Network," *IEEE Access*, vol. 7, pp. 78000–78011, 2019.

[11] Y. Gan, "Facial Expression Recognition Using Convolutional Neural Network," *Proceedings of the 2nd International Conference on Vision, Image and Signal Processing*, 2018.

[12] G. P. Kusuma, J. Jonathan, and A. P. Lim, "Emotion Recognition on FER-2013 Face Images Using Fine-Tuned VGG-16," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 6, pp. 315–322, 2020.

[13] "Convolutional Neural Networks cheatsheet," CS 230 - Convolutional Neural Networks Cheatsheet. [Online]. Tersedia: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks. [Diakses: 16-Mei-2021].

[14] S. Amidi and A. Amidi, architecture-cnn-en.jpeg. Stanford University.

[15] IBM Cloud Education, "What are Convolutional Neural Networks?," IBM. [Online]. Tersedia: https://www.ibm.com/cloud/learn/convolutional-neural-networks. [Diakses: 16-Mei-2021].

[16] IBM Cloud Education, Feature Hierarchy.jpg.

[17] J. Brownlee, "A Gentle Introduction to Batch Normalization for Deep Neural Networks," Machine Learning Mastery, 03-Dec-2019. [Online]. Tersedia: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/. [Diakses: 18-Mei-2021].

[18] J. Brownlee, "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks," Machine Learning Mastery, 06-Aug-2019. [Online]. Tersedia: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. [Diakses: 18-Mei-2021].

[19] "Challenges in Representation Learning: Facial Expression Recognition Challenge," Kaggle. [Online]. Tersedia: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data?select=fer2013.tar.gz. [Diakses: 10-Mei-2021].

[20] TensorFlow. [Online]. Tersedia: https://www.tensorflow.org/. [Diakses: 18-May-2021].

[21] K. Team, "Simple. Flexible. Powerful.," Keras. [Online]. Tersedia: https://keras.io/. [Diakses: 18-May-2021].

[22] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. CVPR 2001.

[23] Save The Sound, image.jpg. Save The Sound.

[24] P. Kongkla, Screenshot-18.png. WALAILAK UNIVERSITY.G. R. Barai, S. Krishnan dan B. Venkantesh, "Smart Metering and Functionalities of Smart Meters in Smart Grid - A Review," *EPEC 2015*, pp. 138-145, 2016.