JUSTIN
Jurnal Sistem dan Teknologi Informasi
p-ISSN : 2460-3562 / e-ISSN : 2620-8989
DOI: 10.26418/justin.v11i3.61111
Vol. 11, No. 3, Juli 2023

Klasifikasi Citra *Rontgen* Covid-19 dengan menggunakan *Deep Learning*

Evrita Lusiana Utari[a1], Prastowo Kristiyanto[b2], Agus Qomaruddin Munir[a,b3]
[a]*Teknik Elektro , Teknologi Informasi, Universitas Respati Yogyakarta*
Jl. Lasda Adisucipto Km 6,3 Depok Sleman Yogyakarta
[1]evrita_lusiana@respati.ac.id
[3]agusqmrn@respati.ac.id
[b]*Universitas Respati Yogyakarta*
Jl. Laksda Adisucipto Km 6,3 Depok Sleman Yogyakarta
[2]prastowo@yahoo.com

Abstrak

Citra adalah representasi dari suatu obyek yang ditulis ulang pada suatu medium dengan nilai tertentu (intensitas) yang memiliki koordinat x dan y. Citra *Rontgent* merupakan salah satu jenis citra medis yang dapat digunakan untuk mendeteksi dan mempelajari suatu penyakit. Namun citra *rontgen* terkadang terlihat kabur sehingga sedikit sulit untuk mengintepretasi citra. Terlebih lagi adanya redaman sinar-X yang berbeda antara kelenjar pada jaringan yang normal dengan yang terpapar penyakit. Dengan mengimplementasikan *deep learning* dengan metode klasifikasi citra dapat memilah gambar berdasarkan ekstrasi fitur dan bobot pada jaringan syaraf tiruan. Ketika *GPU* yang dimiliki adalah *AMD*, salah satu cara agar dapat menjalankan *Deep Learning* menggunakan *AMD* adalah mnggunakan *PlaidML*.

Tahapan yang dilakukan pada pelatihan dan pengujian adalah melakukan *pre-procesessing*, ektraksi fitur menggunakan lapisan JST *VGG16* tanpa lapisan pengklasifikasi (konvolusi dan pooling) yang menghasilkan *bottleneck.npy*, kemudian membuat lapisan pengklasifikasi sendiri untuk melatih klasifikasi kelas covid dan normal menggunakan data *bottleneck.npy*. Tingkat akurasi yang diperoleh pada tahap pelatihan beserta validasi pada pelatihan, dan pengujian berturut-turut adalah 99%, 97%, dan 94%. Selanjutnya ketika dievaluasi dengan *F1 Score* mendapatkan hasil 0,939.

Kata kunci: Deep Learning, VGG16, bottleneck, PlaidML, Deep Learning menggunakan AMD

CLASSIFICATION COVID-19 RONTGEN IMAGE WITH DEEP LEARNING

Abstract

Image is representation of an object that rewritten on a medium with a certain value (intensity) which has x and y coordinates. X-ray image is one type of medical image that can be used to detect and study a disease. However, the X-ray image sometimes looks blurry so it is a little difficult to interpret the image. Moreover, there is a different X-ray attenuation between glands in normal and diseased tissues. By implementing deep learning with image classification methods, it is possible to sort images based on feature extraction and weights on an artificial neural network. When the GPU owned is AMD, one way to be able to run Deep Learning using AMD is to use PlaidML.

The stages carried out in training and testing are pre-processing, feature extraction using the VGG16 ANN layer without a classifier layer (convolution and pooling) which results in bottleneck.npy, then creating its own classifier layer to train covid and normal class classification using bottleneck.npy data. The level of accuracy obtained at the training stage along with validation in training, and testing are 99%, 97%, and 94%, respectively. Furthermore, when evaluated with the F1 Score, the result was 0.939.

Keywords: Deep learning, VGG16, bottleneck, PlaidML, Deep learning using AMD

1. PENDAHULUAN

Covid-19 (*Coronavirus Disease* 2019) adalah salah satu penyakit mematikan yang sejak 2019 menjadi sangat disorot oleh media karena mudah menular ke manusia. Walaupun virus ini masuk ke Indonesia pada bulan Maret 2020, hingga sekarang tahun 2021 dampak dari virus ini masih terasa walaupun tidak separah dahulu sewaktu tahun 2020. Salah satu dampaknya yaitu berupa kebijakan dilarang mudik pada tanggal 6 hingga 17 Mei 2021 (Menko PMK, 2021).

Covid-19 disebabkan oleh *Severe Acute Respiratory Syndrome Coronavirus* 2 (SARS-CoV-2). Covid-19 memiliki gejala seperti flu biasa seperti batuk, pilek, nyeri pada tenggorokan, serta demam. Selain memiliki gejala seperti flu biasa, Covid-19 menyebar melalui *droplet* ketika seseorang berbicara, bersin, atau batuk. Sehingga penyakit ini sangat mudah menular kepada orang lain (Kemkes, 2020). Virus ini bermula di Wuhan Provinsi Haubei China pada Desember 2019. Virus ini menyebar ke seluruh dunia. Covid-19 dinyatakan sebagai darurat kesehatan internasional oleh *WHO* sejak 30 Januari 2020 (WHO, 2020). Tidak terkecuali Indonesia, negeri ini juga terdampak covid-19. Laporan kasus covid-19 pertama di Indonesia terdapat dua kasus sekaligus pada 2 maret 2020 (Putranto, 2020).

Pada tanggal 02 Maret 2021, kasus covid-19 di Indonesia mencapai 1.347.026 yang telah terkonfimasi positif terkena covid-19. Namun ada 1.160.863 yang telah sembuh serta 36.518 yang telah meninggal (covid19.go.id, 2021). Untuk mendeteksi covid-19 terdapat beberapa pendekatan, seperti *rapid test*, *PCR*, Genose dan *Swab*. Selain keeempat metode tersebut, untuk mendeteksi covid-19 bisa juga dengan cara pengambilan citra medis, baik itu *rontgen*, *CT-Scan*, dan *MRI*. Selain untuk mengidentifikasi penyakit termasuk covid-19, citra medis juga dapat dijadikan untuk mempelajari suatu penyakit pada umumnya, covid-19 secara khusus (Lin et al., 2020).

Salah satu dari citra medis adalah citra *rontgen*. Citra *rontgen* lebih mudah didapatkan karena hampir semua rumah sakit pasti memilikinya. Selain itu biaya operasionalnya yang lebih terjangkau jika dibandingkan dengan *CT-Scan* dan *MRI*. Namun citra *rontgen* terkadang terlihat kabur sehingga sedikit sulit untuk mengintepretasi citra. Karena sedikit sulit untuk diintepretasi maka hasil pembacaannyapun terkadang berbeda. Terlebih lagi ditambahnya ada redaman sinar-X yang berbeda antara kelenjar pada jaringan yang normal dengan yang terpapar penyakit (Pereira et al., 2020).

Untuk mengatasi permasalahan ini dapat diselesaikan dengan pengolahan citra, terlebih lagi jika ditenagai dengan *deep learning*. Kemampuan *deep learning* dalam dalam mengekstrak fitur klinis dan hasil laboratorium dapat mendeteksi covid-19 lebih dini. Dengan mengimplementasikan *deep learning* dengan metode klasifikasi gambar dapat memilah gambar berdasarkan ekstrasi fitur dan bobot pada jaringan syaraf tiruan. Namun kemampuan klasifikasi gambar tidak hanya untuk memilah citra dengan mengelompokan saja. Teknik juga bisa untuk memprediksi suatu citra ke dalam kelompok yang sudah dilatih. (Yudistira dkk, 2020).

Begitu pula dengan implementasi dari *keras* terdapat banyak sekali penelitian sebelumnya. Salah satunya dilakukan oleh Shafira (2018) dalam penelitiannya tentang Implementasi *Convolutional Neural Network* untuk Klasfikasi Citra Tomat Menggunakan *Keras*. Pada penelitian ini menyatakan bahwa dalam penelitian ini terdapat langkah-langkah yang harus dilakukan sebelum penelitian dilakukan. Langkah-langkah tersebut yakni, pengumpulan data, preproses citra, perancangan pengolahan citra menggunakan *keras* pada aplikasi R-Studio. Peubah yang digunakan pada penelitian ini terdapat beberapa peubah. Peubah-peubah tersebut yaitu, ukuran filter, jumlah *neuron*, jumlah data. Terdapat juga peubah perbandingan antara data untuk *training* dan juga *testing*. Dan dari perubahan peubah-peubah tersebut juga memiliki dampak terhadap akurasi ketepatan memprediksi klasifikasi citra tomat (Shafira, 2018).

Beberapa penelitian lain tentang pengolahan citra yang memanfaatkan *OpenCL* salah satunya dilakukan oleh Amalia (2016) pada penelitiannya yang berjudul Implementasi Metode *Alpha Trimmed Mean Filter* Menggunakan *OpenCL* untuk Penghapusan *Noise* pada Citra Berwarna menyatakan bahwa pemerogaman parallel *CPU* dan *GPU* lebih efektif jika dibandingkan dengan pemrogaman serial atau hanya mengandalkan *CPU* saja. Pada penilitian ini memaparkan perbandingan *Noise Density* antara kanal merah, biru, dan hijau. Peubah yang digunakan berupa pengambilan gambar pada luar ruangan, dalam ruangan serta persentase densitas *noise* (Amalia, 2016).

Pada penilitian yang berjudul *Stucture of Deep Learning Inference Engines for Embedded Systems* yang dilakukan oleh Seung-Mok Yoo dkk pada ICTC tahun 2019. Pada penelitian ini menjelaskan tentang pemanfaatan plaidml pada Sistem Tertanam (*Embedded System*). Dengan begitu akselerasi hardware pada Sistem Tertanam dapat dilakukan. Akselerasi disini agar perangkat sistem tertanam dapat memproses data lebih cepat. Namun pengendalian dilakukan pada server laboratorium (Yoo dkk, 2019)

Namun sayangnya ketika melatih model *deep learning* dibutuhkan daya komputasi yang tinggi. Jika mengandalkan *CPU* saja dirasa masih sangat kurang karena kemampuan komputasi *CPU* kalah dengan kemapuan komputasi paralel dari *GPU*. Sehingga *GPU* dapat menjadi solusi dalam kebutuhan komputasi yang tinggi. Namun permasalahan tidak hanya pada hal yang sudah disebutkan saja. Ketika ingin menggunakan *GPU* sebagai alat komputasi dalam *deep learning*, kebanyakan aplikasi atau *platform* maupun *framework* diharuskan menggunakan *GPU* dengan merk dari Nvidia. Namun dapat diatasi dengan *plaidml* yang dikembangkan oleh vertex.ai.

Pengolahan citra dengan menggunakan *deep learning* ada banyak implementasinya diantaranya ada *Convolutional Neural Network*, *Residual Neural Network*, *Dense Neural Network*, dll. Pada penelitian ini akan mengimplementasikan *deep learning* dengan metode *bottleneck* dan algoritma *CNN* atau *Convolutional Neural Network* sebagai ekstraktor fitur. Metode *bottleneck*

merupakan metode yang singkat dikarenakan ekstraksi fitur hanya sekali, namun memberikan hasil yang memuaskan. Berdasarkan laman tutorial pada blog keras dengan judul *Building powerful image classification models using very little data*, dengan metode ini dapat mencapai tingkat akurasi lebih dari 90%. (Chollet, 2016). *Framework* yang mendukung metode *bottleneck* ada banyak salah satunya *keras*. *Keras* menjadi pilihan karena *open source* sehingga siapapun dapat memodifikasi dan memberikan kontribusi. Selain itu juga *keras* adalah salah satu *framework* yang didukung oleh plaidml. Dengan begitu dapat menjalankan *keras* dengan *GPU* (*OpenCL*). Sehingga ketika melakukan komputasi dengan *deep learning* yang menggunakan *framework keras* dapat dilakukan lebih cepat jika dibandingkan dengan *CPU*.

Rumusan masalah:

1. Mengklasifikasikan citra berbasis *keras* dengan plaidml sebagai *backend* dengan metode *bottleneck*.
2. Melatih dan menguji klasifikasi citra berbasis *keras* dengan plaidml sebagai *backend* dengan metode *bottleneck*.

Tujuan dari penelitian :

1. Untuk mengetahui bahan-bahan yang dibutuhkan untuk mempersiapkan perancangan klasifikasi gambar berbasis *keras* dengan plaidml sebagai *banckend*.
2. Untuk mengatahui langkah-langkah dalam perancangan *deep learning*, lebih khususnya klasifikasi gambar berbasis *keras* dengan plaidml sebagai *banckend*.

2 METODOLOGI

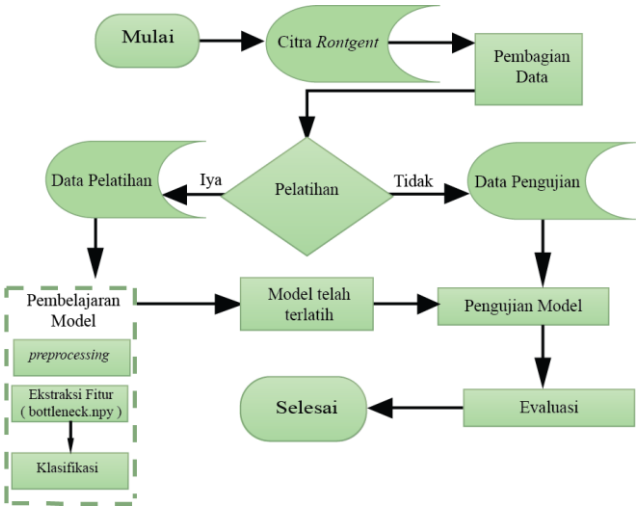
Metode yang dilakukan dengan mempersiapkan data set citra yang diambil dari kumpulan data yang akan menjadi bahan uji coba pada penelitian. Dataset yang penulis gunakan adalah dataset *Covid-19 Radiography Database*. Dataset *Covid-19 Radiography Database* diunduh dari situs Kaggle. Kaggle adalah sebuah situs yang menyediadakan dataset dan juga kompetisi *data scientist* dan merupakan anak perusahaan dari google. Dataset ini diperbarui oleh Tawsifur Rahman pada 06 Maret 2021. Dataset ini berukuran 745 MB yang berisi 3616 data *rontgen* covid-19, 6012 *lung opacity*, sekitar 10200 data *rontgen* dengan keadaan normal dan 1345 data *rontgen* dengan kategori viral pneumonia. Semua dataset berekstensi PNG dan memiliki resolusi 299x299 piksel. Namun yang digunakan untuk pelatihan 4000 pada dua *classes* dan validasi 1000 pada dua *classes*. *Classes* yang dimaksud ialah Covid dan Normal. Jadi total citra yang menjadi bahan pelatihan jumlahnya ada 5000 citra. Karena jika menggunakan semua data, laptop yang digunakan untuk uji coba tidak kuat.

Berdasarkan gambar 1 citra *rontgent* dari data set dibagi menjadi dua bagian yaitu data pelatihan dan data pengujian. Dengan perbandingan 4000:1000. 4000:1000 ini terbagai menjadi dua kelas yaitu COVID dan Normal. Sehingga masing-masing *kelas* pada data pelatihan ada 1000 citra. Sedangkan pada data pengujian ada 500 citra. Setelah terbagi data akan dimasukkan ke dalam sistem untuk dipelajari oleh sistem.

Pada pembelajaran sistem terbagi beberapa proses dengan urutan *preprocessing*, ekstraksi fitur, dan klasifikasi. Pada tahap *preprocessing* hanya menyakala ulang (*rescale*) nilai-nilai intensitas dari matrik (citra) yang semula memiliki rentang nilai -255 hingga 255 menjadi -1 hingga 1. Sebagai contoh, seperti matrik dibawah.

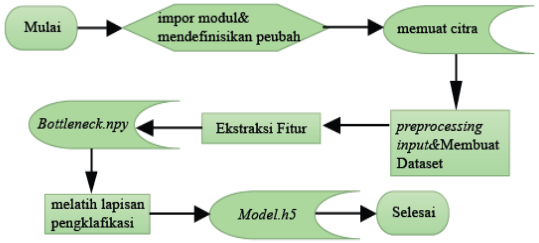
Tahap berikutnya ekstraksi fitur, karena metode yang penulis gunakan menggunakan metode fitur *bottleneck* maka ekstraksi fitur dan proses klasifikasi dilakukan secara terpisah. Pada tahap ekstraksi fitur dilakukan ekstraksi menggunakan JST *VGG*16 (Jaringan Syaraf Tiruan *Visual Geometry Group* yang memiliki 16 lapisan berbobot) tanpa lapisan pengklasifikasi sehingga menghasilkan ekstraksi fitur dengan ukuran matrik 4000, 7, 7, 512 yang kemudian disimpan dengan ekstensi npy. Kemudian tahap klasifikasi ekstraksi fitur yang telah didapat dijadikan masukan pada lapisan pengklasifikasi sehingga model *deep learning* dapat membedakan kelas yang diberikan.

Setelah model terlatih model akan diuji dengan sebuah program. Kemudian pada program tersebut dilakukan beberapa tahapan persiapan untuk menguji model terlatih tersebut. Persiapannya berupa mengimpor model.h5 yang sudah terlatih, memasukkan data pelatihan, dan pembuatan label. Model.h5 yang diimpor berisi arsitektur JST beserta bobot pada masing-masing lapisan yang telah dilatih. Kemudian model diuji dengan data pelatihan tersebut. Setelah itu persiapan berikutnya yaitu untuk menampilkan hasil pengujian dengan membuat *dataframe* yang kemudian diubah menjadi file *comma separated value* (.csv). Pada *dataframe* tersebut dilalakukan penggantian label dari 0 dan 1 menjadi COVID dan Normal. Setelah itu data ditampilkan menggunakan *crosstab*. Ketika terjadi ketidakseuain atau kesalahan prediksi dapat dilacak dengan cara membuka Microsoft excel dengan mengubah file csv menjadi file excel biasa dan dapat dilihat dimana data yang keliru.



Gambar 1 Diagram Alir Perancangan Sistem

Perancangan aplikasi pelatihan model
Adapun untuk diagram alir untuk pelatihan model dapat dilihat pada gambar 2

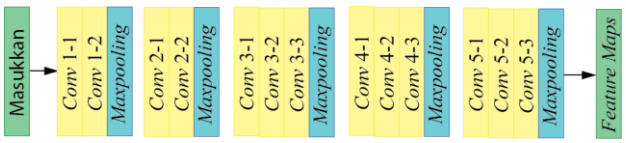


Gambar 2 Diagram Alir Pelatihan

Impor modul dan definisi *variable* peubah

Pada python modul-modul tidak secara otomatis termuat. Untuk memuat modul tersebut harus diimpor terlebih dahulu. Modul-modul yang diperlukan pada Perancangan pelatihan adalah plaidml.keras, os, numpy, *ImageDataGenerator* dari keras.preprocessing, *Sequential* dan *model* dari keras.models, *Flatten*, *Dropout*, *Dense*, *Activation* dari keras.layers, *application* dan *optimizers* dari keras, matplotlib.pylab, *ModelCheckpoint* dari keras.callbacks. Selain mengimpor modul-modul tersebut juga harus mendefinisian *function* plaidml.keras.install_backend() dan os.environ["KERAS_BACKEND"] = "plaidml.keras.backend" supaya dapat memuat *keras* dengan *backend PlaidML*. Untuk selanjutnya mendefinisikan *variable* global yang dibutuhkan supaya memudahkan dalam pengiistilahan dan penyeragaman jika pemanggilan lebih dari sekali. *Variable* yang didefinisikan adalah ukuran citra, direktori untuk menyimpan model, direktori data pelatihan dan validasi, jumlah data pelatihan dan validasi, jumlah *epoch*, ukuran *batch size*, jumlah data yang akan dijadikan label dan *callback* yang digunakan.

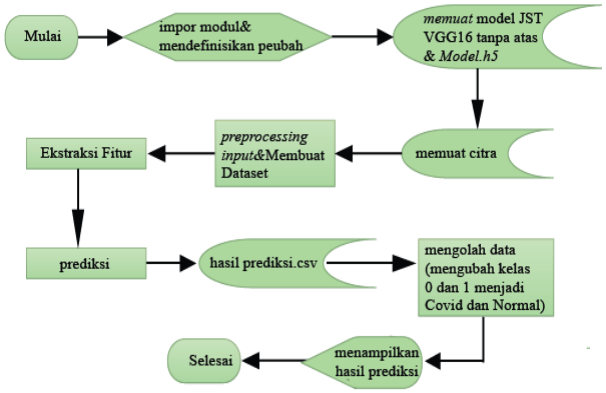
Ekstraksi Fitur
Pada tahap ini dibuat sebuah *function* ekstraksi fitur agar dalam pelatihan tidak perlu melakukan ekstraksi fitur secara berulang. Sehingga dapat menghemat waktu karena proses ekstraksi fitur lumayan lama. Pada proses ini dilakukan beberapa persiapan yaitu penyekalaan ulang, mengimpor model VGG16, pembuatan dataset. Setelah itu dilakukan ekstraksi fitur menggunakan JST VGG16 yang kemudian disimpan dengan ekstentensi .npy.



Gambar 3. Struktur Ekstraksi Fitur

Pelatihan Model
Pada tahap ini juga dibuat *function* pelatihan. Setelah proses ekstraksi selesai data ekstraksi dimuat kembali pada *function* ini. Ketika menggunakan *function*, *variable* hanya bisa diakses didalam *function* (*private variable*). Itulah mengapa data harus dimuat ulang dan pendefinisian ulang *variable* pada ekstraksi fitur. Setelah termuat, membuat *variable* untuk label dengan parameter pada *variable* yang sudah didefinisikan secara global. Kemudian membuat lapisan *neural network* bagian atas untuk mengklasifikasi *classes*. Lapisan *neural network* yang digunakan adalah *Flatten*, *Fully Connected* (dense) dengan aktifasi *ReLU*, *Dropout*, dan *Fully Connected* (dense) dengan aktifasi *Sigmoid*. Setelah mendefinisikan JST, harus dikompilasi terlebih dahulu menggunakan model.compile(). Kemudian menghadirkan model.fit_generator() untuk melatih JST pengklasifikasi dan merekam jejak pelatihan menggunakan *function* history.keys() dan matplotlib.pylab.

Perancangan Aplikasi untuk Pengujian Model
Adapun diagram alir untuk pengujian model dapat dilihat pada gambar 3.



Gambar 4. Diagram Alir Pengujian

Impor modul dan mendefinisikan *Variable* peubah

Begitu pula pada pengujian model, juga dilakukan impor modul yang diperlukan. Modul-modul tersebut adalah *plaidml.keras*, os, *ImageDataGenerator*, dari *keras.preprocessing.image*, *Model* dan *load_model* dari *keras.models*, matplotlib.pylab, numpy, cv2 (*opencv*), pandas, *seaborn*, dan terakhir *applications* dari *keras*.

Persiapan Pengujian dan Pengujian Data

Pada persiapan pengujian dilakukan pendefinisian *variable* yang berisi perintah tertentu seperti memuat VGG16, memuat model yang sudah dilatih, pembuatan label, nama file, penyekalaan ulang (normalisasi), dan pembuatan *dataset* menggunakan *function* flow_from_directory().
Untuk pengujian data, tetap harus melalui ekstraktor fitur yang menggunakan *variable* yang berisi VGG16 yaitu model2 dan ditambahkan *function predict_generator()*. Untuk *function predict_generator*() diisi dengan *variable dataset*. Untuk pengujian *classes* menggunakan *variable* yang berisi model yang telah dilatih yaitu *variable* model. *Variable* model ditambahkan *function predict_classes()*. Pada *function predict_classes*() diisi dengan *variable* ekstraktor fitur. Selanjutnya *array* yang berisikan prediksi

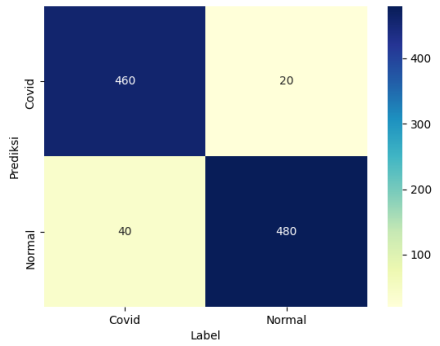
kelas dijadikan matrik satu dimensi dengan *function flatten*().

Pengolahan Data menggunakan *DataFrame*

Setelah proses prediksi selesai langkah berikutnya yaitu menyimpan data mentah dari prediksi menjadi file *comma separated value* (.csv). Pada data mentah ini prediksi dan label masih menggunakan angka yaitu 0 untuk COVID dan 1 untuk Normal. Data diolah supaya prediksi dan label menjadi COVID dan Normal dengan memanfaatkan *DataFrame* dari modul pandas.

Visualisai Hasil Prediksi

Setelah label dibenahi menjadi COVID dan Normal, langkah berikutnya yaitu menampilkan hasil prediksi dengan menggunakan *function crosstab* dari modul pandas agar hasil prediksi menjadi bentuk tabel. Namun hasil dari *function crosstab* sedikit membingungkan dan kurang menarik. Maka dari itu hasil prediksi yang sudah diubah menjadi tabel dikonversi dengan menggunakan *function heatmap* dari modul *seaborn* dengan begitu menghitung tingkat akurasinya akan mudah. Sehingga hasilnya seperti pada gambar 5.



Gambar 5. Hasil dan Prediksi

Perancangan Evaluasi

Setelah dataset pengujian diuji menggunakan model terlatih dan mendapatkan hasil. Selanjutnya dilakukan evaluasi dari pengujian tersebut. Evaluasi pada penelitian ini menggunakan akurasi dan *F1-Score.* Akurasi didapatkan dari persentase keberhasilan model memprediksi data dengan tepat. Sedangkan *F1-Score* didefinisikan sebagai tingkat keberhasilan dari model yang telah dibangun. Keberhasilan ini merajuk pada nilai presisi dan *recall* yang baik. Untuk mendapatkan nilai akurasi, *F1-Score*, *recall*, dan juga presisi dapat menggunakan *confusion matrix* (pandas.crosstab).

TABEL 1.PERANCANGAN EVALUASI

Nilai	**Kondisi**	**Nilai Aktual**	
		Positif	**Negatif**
Nilai Prediksi	Positif	*True Positive* (TP)	*False Positive* (FP)
	Negatif	*False Negatif* (FN)	*True Negatif* (TN)

Akurasi

Akurasi adalah perbandingan jumlah data yang diprediksi benar (TP+TN) dengan seluruh data. Akurasi mencerminkan keakuratan dari model yang telah dilatih. Nilai Akurasi didapatkan dari persaman berikut:

$$Akurasi = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

Presisi dan *Recall*

Presisi adalah perbandingan antara nilai *true positive* dengan jumlah dari *true positive* dan *false positive*. Presisi mencerminkan tingkat keakuratan data yang diinginkan dengan hasil prediksi dari model yag telah terlatih. Presisi dihitung dengan rumus:

TP/(TP+FP) (2)

Sedangkan *recall* merupakan perbandingan antara *True Positive* dengan jumlah dari *True Positive* dan *False Negatif*. Recall mencerminkan penemuan kembali suatu informasi. Jika ditulis dengan persamaan maka:

TP/(TP+FN) (3)

F1 *Score*

F1 *Score* didefinisikan sebagai perhitungan kombinasi dari nilai presisi dan *recall*. Hasil perhitungan tersebut disebut dengan nilai pengukuran. *F1 Score* diperoleh dari persamaan berikut:

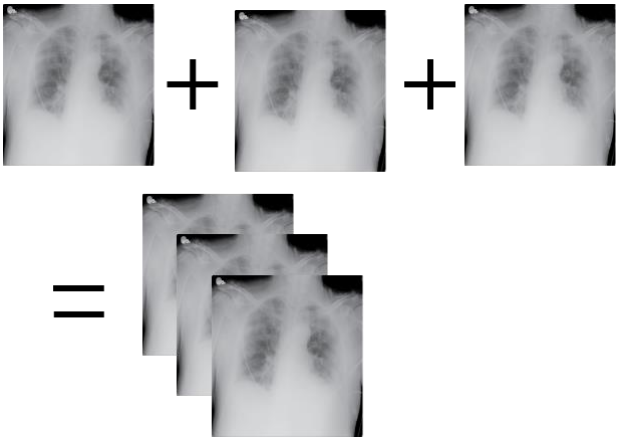
$$F1\ Score = 2 \times \frac{\text{Presisi} \times Recall}{\text{Presisi} + Recall} \quad (4)$$

3. HASIL DAN PEMBAHASAN

Pada penelitian ini data yang telah terbagi kedalam data pelatihan akan diproses dari tahap *pre-processing*, pembuatan *dataset*, pengolahan citra pada *deep learning*. Data yang terbagi kedalam data pengujian juga akan diproses. Semua tahapan dibuat dengan bahasa python dengan *framework keras* dan *PlaidML* sebagai *backend*. Aplikasi yang digunakan adalah notepad++ dan anaconda. Selain itu, pada penelitian ini juga digunakan Microsoft excel untuh menampilkan hasil pelatihan secara rinci.

Tahap Pembuatan Dataset dan *Preprocessing*

Dataset adalah kumpulan data yang menjadi masukan pada sistem *deep learning*. Masukkan Model *deep learning VGG*16 membutuhkan tiga kanal, sedangkan citra skala keabuan hanya memiliki satu kanal. Maka dari itu Model sacara bawaan menjadikan citra skala keabuan menjadi tiga kanal dengan cara menduplikasi kanal pertama. Sehingga ketiga kanal memiliki nilai intensitas yang sama. Hal ini dapat diilustrasikan pada gambar 6



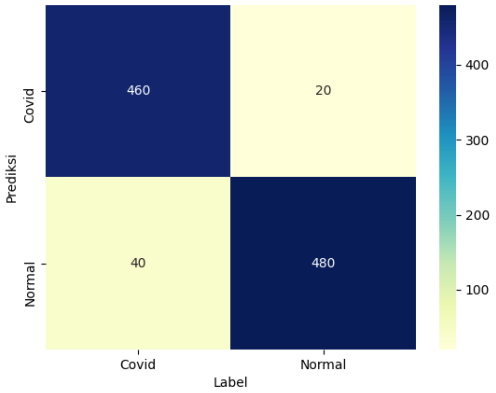
Gambar 6. Ilustrasi Tiga Kanal Skala Keabuan

Hasil penguiian pada penelitian ini berupa tampilan *confussion matrix* dari modul *crosstab* yang disempurnakan menggunakan *heatmap* dari modul *seaborn* dan berkas *microsoft excel*. *Crosstab* dapat menampilkan keseluruhan data label dan predikasi kedalam suatu tabel. Sehingga dapat memudahkan untuk menghitung tingkat akurasi pengujian dan tingkat kesalahan pengujian. Sedangkan berkas *Microsoft excel* digunakan untuk melacak berkas yang salah predikasi.
Hasil dari *crosstab* tersebut berupa tabel 2.

TABEL 2. CROSSTAB PENGUJIAN

Prediksi Label	Covid	Normal
Covid	460	40
Normal	20	480

Untuk memperindah tampilan *crosstab* dan mempermudah pembacaan tersebut, maka *crosstab* diubah menjadi *heatmap terlihat pada gambar 7.*



Gambar 7. *Heat Map* Pengujian

Dikarenakan ada beberapa data yang salah prediksi dengan jumlah 60 data, maka untuk melacak berkas yang salah prediksi tersebut digunakan *Microsoft excel*. Terlihat pada tabel 3.

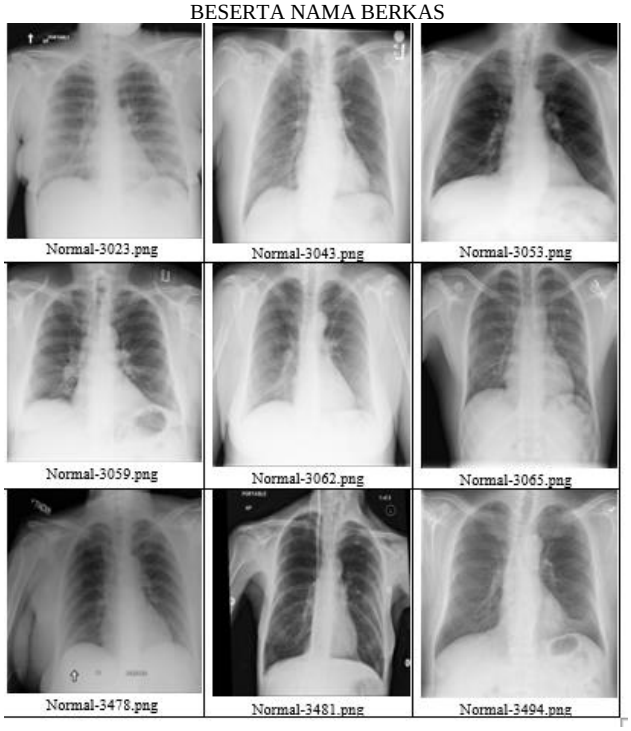
TABEL 3. DATA NORMAL SALAH PREDIKSI

Nama Berkas	Label	Prediksi
Normal-3023.png	Normal	Covid
Normal-3043.png	Normal	Covid
Normal-3053.png	Normal	Covid
Normal-3059.png	Normal	Covid
Normal-3062.png	Normal	Covid
Normal-3065.png	Normal	Covid
Normal-3074.png	Normal	Covid
Normal-3075.png	Normal	Covid
Normal-3079.png	Normal	Covid
Normal-3092.png	Normal	Covid
Normal-3120.png	Normal	Covid
Normal-3133.png	Normal	Covid
Normal-3142.png	Normal	Covid
Normal-3176.png	Normal	Covid
Normal-3183.png	Normal	Covid
Normal-3247.png	Normal	Covid
Normal-3258.png	Normal	Covid
Normal-3478.png	Normal	Covid
Normal-3481.png	Normal	Covid
Normal-3494.png	Normal	Covid

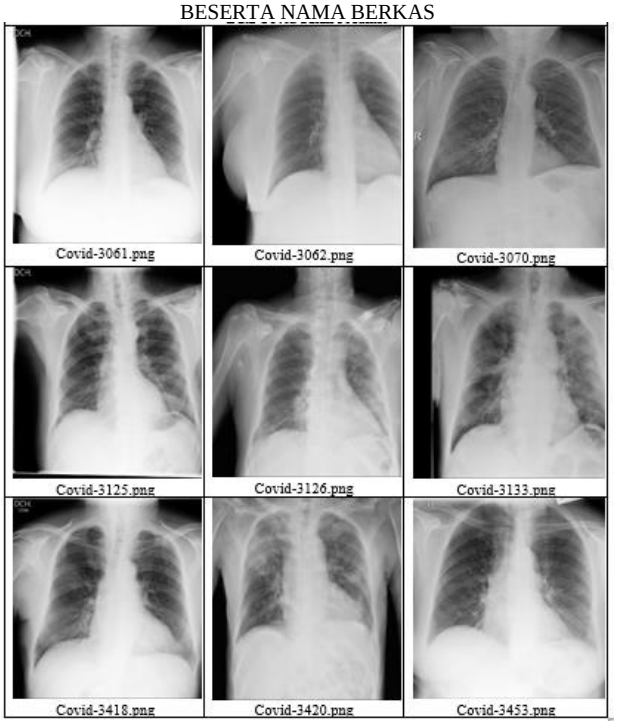
TABEL 4. DATA COVID SALAH PREDIKSI

Nama Berkas	Label	Prediksi
COVID-3061.png	Covid	Normal
COVID-3062.png	Covid	Normal
COVID-3070.png	Covid	Normal
COVID-3125.png	Covid	Normal
COVID-3126.png	Covid	Normal
COVID-3133.png	Covid	Normal
COVID-3157.png	Covid	Normal
COVID-3161.png	Covid	Normal
COVID-3166.png	Covid	Normal
COVID-3174.png	Covid	Normal
COVID-3192.png	Covid	Normal
COVID-3197.png	Covid	Normal
COVID-3205.png	Covid	Normal
COVID-3215.png	Covid	Normal
COVID-3216.png	Covid	Normal
COVID-3220.png	Covid	Normal
COVID-3242.png	Covid	Normal
COVID-3245.png	Covid	Normal
COVID-3259.png	Covid	Normal

TABEL 5 SAMPEL DATA CITRA NORMAL SALAH PREDIKSI BESERTA NAMA BERKAS



TABEL 6. SAMPEL DATA CITRA COVID SALAH PREDIKSI BESERTA NAMA BERKAS



Evaluasi Pengujian

Setelah mengetahui hasil pengujian, selanjutnya menghitug akurasi dan *F1 Score* untuk melihat tingkat akurasi dan keberhasilan dari model yang telah dilatih. Untuk menghitung tingkat akurasi dapat menggunakan rumus sebagai berikut:

$$Akurasi = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Akurasi = \frac{460 + 480}{460 + 40 + 20 + 480}$$

$$Akurasi = \frac{940}{1000}$$

$$Akurasi = 94\%$$

Berikutnya mengitung presisi meggunakan persamaan berikut:

Presisi = TP/(TP+FP)
Presisi = 460/(460+40)
Presisi = 460/500
Presisi = 0,92

Dan untuk menghitung *recall* dengan rumus berikut:

Recall = TP/(TP+FN)
Recall = 460/(460+20)
Recall = 460/480
Recall = 0,958

Selanjutnya menghitung *F1 Score* menggunakan rumus berikut:

$$F1\ Score = 2 \times \frac{Presisi \times Recall}{Presisi + Recall}$$

$$F1\ Score = 2 \times \frac{0{,}92 \times 0{,}958}{0{,}92 + 0{,}958}$$

$$F1\ Score = 2 \times \frac{0{,}88136}{1{,}878}$$

$$F1\ Score = \frac{1{,}76272}{1{,}878}$$

$$F1\ Score = 0{,}939$$

Dengan begitu nilai akurasi yang didapatkan oleh model yang telah terlatih adalah 94%, sedangkan untuk *F1 Score* yang diperoleh 0,939. Selain dua nilai tersebut tingkat akurasi yang diperoleh untuk validasi data sewaktu pelatihan adalah 97%, sedangkan untuk tingkat akurasi pengujian data adalah 94%. Sehingga model yang telah dilatih dapat dikatakan sangat baik. Karena untuk validasi data sewaktu pelatihan dengan tingkat akurasi pengujian tidak terpaut jauh.

4 KESIMPULAN

Kesimpulan yang didapatkan dari penelitian untuk merancang klasifikasi citra berbasis *keras* dengan plaidml sebagai *banckend* dengan metode *bottleneck* adalah aplikasi anaconda beserta modul yang diperlukan, *dataset*, model JST yang dijadikan basis pelatihan dan pengujian, perangkat keras yang didukung oleh *plaidml*.

Tahapan yang dilakukan pada pelatihan dan pengujian adalah melakukan *pre-procesessing*, ektraksi fitur menggunakan lapisan JST *VGG16* tanpa lapisan pengklasifikasi (konvolusi dan pooling) yang menghasilkan *bottleneck.npy*, kemudian membuat lapisan pengklasifikasi sendiri untuk melatih klasifikasi kelas covid dan normal menggunakan data *bottleneck.npy*. Tingkat akurasi yang diperoleh pada tahap pelatihan beserta validasi pada pelatihan, dan pengujian berturut-turut adalah 99%, 97%, dan 94%. Ketika diuji dengan *F1 Score* hasil yang diberikan adalah 0,939. Nilai akurasi pada validasi data dengan skor akurasi pengujian tidak terpaut jauh. Sehingga dapat diambil kesimpulan bahwa model yang telah terlatih ini sudah cukup baik.

DAFTAR PUSTAKA

[1] Nuraini, Ratna. "Kasus Covid-19 Pertama, Masyarakat Jangan Panik". https://indonesia.go.id/narasi/indonesia-dalam-angka/ekonomi/kasus-covid-19-pertama-masyarakat-jangan-panik diakses pada 20 Februari 2021

[2] Tim Komunikasi Komite Penanganan Corona Virus Disease 2019 (COVID-19) dan Pemulihan Ekonomi Nasional. 2021. "Kesembuhan COVID-19 Terus Meningkat Menjadi 1.160.863 Orang". https://covid19.go.id/p/berita/kesembuhan-covid-19-terus-meningkat-menjadi-1160863-orang Diakses pada 21 Maret 2021

[3] Mutiara, Puput. 2021. "Pemerintah Larang Mudik Lebaran 2021". https://www.kemenkopmk.go.id/pemerintah-larang-mudik-lebaran-2021. diakses 30 Mar, 2021

[4] Dartamasia, 2020. "Deteksi Penggunaan Masker menggunakan Xception Tansfer Learning". Jurnal INSTEK (Informatika Sains dan Teknologi) Volume 5 Nomor. 2, Oktober 2020 P –ISSN: 2541-1179, E-ISSN : 2581-1711

[5] Acharya, Tinku dan Ray, Ajoy K. "*Image Processing: Principle and Applications*". New Jersey: John Wiley & Sons, Inc., Hoboken, Canada: simultaneously, ISBN-13 978-0-471-71998-4 (cloth: alk. paper) ISBN-10 0-471-71998-6 (cloth: alk. paper) 2015.

[6] Petrou, Maria dan Bosdogianni, Panagiota.. "*Image Processing: The Fundamentals*". Singapore: JOHN WILEY & SONS, LTD, ISBN 0-471-99883-4 1999.

[7] Pereira, Rodolfo M. et al. "*COVID-19 identification in chest X-ray images on flat and hierarchical classification scenarios*". Computer Methods and Programs in Biomedicine 194 (2020) 105532. https://doi.org/10.1016/j.cmpb.2020.105532

[8] Moolayil, Jojo John. "*Learn Keras for Deep Neural Networks*". Canada: Apress. ISBN-13 (pbk): 978-1-4842-4239-1 ISBN-13 (electronic): 978-1-4842-4240-7 https://doi.org/10.1007/978-1-4842-4240-7, 2016

[9] Chollet, Francois. "*Deep Learning with Python*". Shelter Island: Manning. ISBN 9781617294433, 2017.

[10] Chollet, Francois. "*Building powerful image classification models using very little data*". 2016. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html diakses 25 Juni 2021

[11] Kementerian Kesehatan., "Pertanyaan dan Jawaban Terkait Covid-19". https://www.kemkes.go.id/article/view/20031600011/pertanyaan-dan-jawaban-terkait-covid-19.html diakses pada 01 Apil 2021

[12] *WHO*. 2020. "*Archived: WHO Timeline - COVID-19".* https://www.who.int/news/item/27-04-2020-who-timeline---covid-19 diakses pada 30 April 2021.

[13] Wulandri, Putri,. "Klasifikasi Tingkat Keganasan Kanker Serviks Menggunakan Metode *Deep Residual Network* (*ResNet*)". Skripsi. UIN Sunan Ampel Surabaya. 2019.

[14] Amalia, Kurnia Rizky. "Implementasi Metode *Alpha Trimmed Mean Filter* Menggunakan *OpenCL* untuk Penghapusan *Noise* pada Citra Berwarna". Skripsi. UIN Malik Ibrahim Malang. 2016.

[15] S. -m. Yoo et al., "*Structure of Deep Learning Inference Engines for Embedded Systems*," International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2019, pp. 920-922, doi: 10.1109/ICTC46691.2019.8939843.

[16] Rahmadewi, Reni, dkk. "Klasifikasi Penyakit Paru Berdasarkan Citra *Rontgen* dengan Metode Segmentasi *Sobel*". Jurnal Teknik Elektro Vol. 5 No. 1 Maret 2016. ISSN: 2302 – 2949. DOI: 10.20449/jnte.v5i1.174

[17] Yudistira, Novanto, dkk. "Deteksi Covid-19 Citra Sinar-X Dada Menggunakan *Deep Learning* yang Efisien". Jurnal Teknologi Informasi dan Ilmu Komputer (JTIK) Vol. 7 No. 6 hlm. 1289-1296. e-ISSN: 2528-6579. 2020 DOI: 10.25126/jtiik.202073651

[18] Clara, Serafim, dkk. "Implementasi Seleksi Fitur Pada Algoritma Klasifikasi Machine Learning Untuk Prediksi Penghasilan Pada *Adult Income Dataset*". Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA) Hal. 741-747. e-ISBN 978-623-93343-3-8, 2021.

[19] Dewi, Ni Luh Gede M.U., "Analisis Perbandingan Filter Median, Filter Maksimum-Minimum dan Filter Rerata menggunakan Citra *Rontgen*t Paru". Skripsi. Universitas Respati Yogyakarta.

[20] Gulli, Antonio dan Sujit pal. "Deep Learning with Keras". Birmingham: Published by Packt Publishing Ltd. ISBN 978-1-78712-842-2, 2017.

[21] Rafael C. Gonzales dan Richard E. Woods. "*Digital Image Processing*". New Jersey: Prentice-Hall dan Tom Robbins ISBN 0-201-18075-8, 2002.

[22] Wani, M. Arif, et al. "*Advances in Deep Learning*". Studies in big data vol. 57. Singapura: MetaPress dan Springerlink. ISBN 978-981-13-6794-6, 2020

[23] Shafira, Tiara. "Implementasi Convolutional Neural Networks untuk Klasifikasi Citra Tomat Menggunakan Keras". Tugas Akhir. UII, 2018

[24] Anonim. "*About Keras*". https://keras.io/about/ diakses tanggal 16 Februari 2021

[25] Anonim, "A platform for making deep learning work everywhere". https://plaidml.github.io/plaidml/ diakses pada 20 Februari 2021

[26] Anonim, "Memahami Perbedaan PCR, Rapid Test Antigen, dan Rapid Test Antibodi dalam Pemeriksaan COVID-19". https://www.alodokter.com/memahami-perbedaan-pcr-rapid-test-antigen-dan-rapid-test-antibodi-dalam-pemeriksaan-covid-19 diakses pada 25 Februari 2021

[27] Anonim, "*Keras Application*". https://keras.io/api/applications/ diakses pada 29 Maret 2021.

[28] F. Chollet. 2017."*Xception: Deep Learning with Depthwise Separable Convolutions*," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195.