JIFOTECH (JOURNAL OF INFORMATION TECHNOLOGY)
Vol. 3, No. 1, Maret 2023

Klasifikasi File Gambar Hasil X-Ray Paru -Paru Dengan Arsitektur Convolution Neural Network (CNN)

Indra Bakti [1] ,Mohamad Firdaus[2]
[1] Teknologi Informasi, ITB Ahmad Dahlan Jakarta
[2] Teknik Industri, Universitas Indraprasta PGRI Jakarta
[1]indra.chufran@gmail.com , [2] mfirdausmumu@gmail.com

Abstract - In the last decade, the use of artificial intelligence (AI) is increasing, Convolution Neural Network (CNN) architecture as one of them has helped the world to process Big Data (big data), by conducting a training process from a large number of image datasets, and then being able to learn its characteristics and then select according to its classification. The use of the python programming language as a means to classify an image file with CNN architecture is very good and has very high accuracy. Keras is a module that has been prepared by python programming language developers to support self-learning artificial intelligence that also supports CNN architecture. In CNN programming with python language, it is mentioned that the more parameters are tested, the better the artificial intelligence gives output. Therefore, the objective this test is to prove whether the more parameters given to the system can provide better output with the method used is to provide input dataset training. The results of the study can be from the number of parameters learned 1,059,106 with an average time of 135 ms, then the machine can classify the image correctly by 87.88%. if the machine learns with more parameters of 3,824,486 with an average time of 780 ms, then the machine can guess close to 100%. The results of the study can be seen that the more the machine learns with its artificial intelligence capabilities, the more proficient the machine will be.

Keywords: python, Convolution Neral Network, X-Ray, lungs

Intisari-Dalam dekade terakhir ini pemanfaatan kecerdasan buatan (Artificial Intelligent) Semakin banyak, Arsitektur Convolution Neural Network (CNN) sebagai salah satunya telah banyak membantu dunia untuk mengolah Big Data (data besar), dengan melakukan proses pelatihan dari sejumlah besar dataset image, dan kemudian dapat mempelajari cirinya dan kemudian memilihkan sesuai dengan klasifikasinya. Penggunaan bahasa pemrograman python sebagai sarana untuk mengklasisifikasikan sebuah file gambar dengan arsitektur CNN sangat baik dan memiliki akurasi sangat tinggi. Keras adalah suatu modul yang sudah disiapkan oleh pengembang bahasa pemrograman python untuk mendukung kecerdasan buatan yang dapat belajar sendiri yang juga mendukung arsitektur CNN. Dalam pemrograman CNN dengan Bahasa python disebutkan semakin banyak parameter yang uji maka semakin baik kecerdasan buatannya memberikan output. Oleh karena itu tujuan pengujian ini adalah ingin membuktikan apakah semakin banyak parameter yang diberikan ke sistem dapat memberikan keluaran yang semakin baik dengan metode yang digunakan adalah memberikan training dataset input. Hasil penelitian dapat dari banyaknya parameter yang dipelajari 1.059.106 dengan waktu rata 135 ms, maka mesin dapat mengklasfikasikan gambar dengan tepat sebesar 87,88%. bila mesin belajar dengan parameter yang lebih banyak lagi yaitu sebesar 3.824.486 dengan waktu rata-rata 780 ms, maka mesin dapat menebak mendekati 100%. Hasil penelitian dapat dilihat bahwa semakin banyak mesin belajar dengan kemampuan kecerdasan buatannya maka mesin tersebut akan semakin mahir

Kata Kunci : python, Convolution Neral Network, X-Ray, paru-paru

I.PENDAHULUAN

Kecerdasan buatan adalah kecerdasan yang ditambahkan ke dalam sistem yang dapat diatur dalam konteks ilmiah, atau yang juga dapat disebut sebagai kecerdasan buatan atau hanya kecerdasan buatan, yang didefinisikan sebagai kecerdasan unit ilmiah.. Kecerdasan buatan ini didasari pada prinsip yang menyatakan bahwa kecerdasan manusia dapat terdefinisi sedemikian rupa sehingga mesin juga dapat meniru dan menjalankan tugas dari yang paling sederhana hingga yang lebih kompleks. Tujuan dari kecerdasan buatan adalah melakukan pembelajaran, penalaran, dan persepsi.

Bagi para Peneliti yang baru dan baru mengetahui tentang kecerdasan buatan atau AI (*Artificial Intelligence*) dan *deep learning*, pemrograman menggunakan *interface library Keras* dapat menjadi suatu alat yang sangat *powerfull* untuk digunakan. *Library* Keras ini adalah salah satu modul dari bahasa pemrograman *Python*. Oleh karena itu dengan metode *training dataset input*, apakah sistem dapat menjadi lebih cerdas. Dalam penelitian sebelumnya bahkan sistem ini dapat mengenali jenis-jenis tanaman [1], memprediksi uang kripto[2] atau memprediksi mata uang Rupiah [3]. Penelitian ini sebenarnya adalah pengembangan dari penelitian selanjutnya dari penelitian sebelumnya dengan judul "Arsitektur Convolutional Neural Network InceptionResNet-V2 Untuk Pengelompokan Pneumonia Chest X-Ray [4] yang buat oleh Indra Bakti dan Mohamad Firdaus. Dalam penelitian tersebut perlu juga dibuktikan apakah dengan menambah paramater yang ada didalam *Machine Learning* dapat menunjukkan bahwa mesin dapat memperoleh pengetahuan yang semakin meningkat. Dengan metode *training dataset input* dalam *phyton* diharapkan dapat memberikan perbandingan untuk hasilnya

II.TINJAUAN PUSTAKA

A. Python

Python adalah bahasa pemrograman yang digunakan untuk membuat berbagai aplikasi, perintah komputer, dan melakukan suatu analisis data. Sebagai bahasa

pemrograman yang memiliki popularitas tinggi, *Python* dapat digunakan untuk membuat berbagai jenis program dan juga dapat menyelesaikan berbagai permasalahan secara komputasi. Selain itu, Sebagai bahasa pemrograman tingkat tinggi *Python* juga dinilai mudah untuk dipelajari. *Python* memiliki berbagai kemampuan sebagai berikut :

- Matematika: menyelesaikan permasalahan matematika seperti trigonometri, aljabar dan kalkulus.
- Penulisan *script* sistem: membuat suatu perintah secara otomatis dan secara langsung juga menyelesaikan pekerjaan yang membutuh waktu dengan cepat.
- Pengembangan perangkat lunak: melakukang *bug tracking* dan melakukan tes perangkat lunak.
- Pengembangan Situs Web: memastikan keamanan *website* dalam memproses dan mengirim data.
- Analisa Data : melakukan visualisasi data, kalkulasi statistik dan penganalisaan data.
- Pembelajaran mesin *(Machine learning)*: membuat algoritma untuk modul pembelajaran

B. Keras

Keras adalah suatu *application programming interface* (API) yang dibuat untuk memudahkan pengguna dalam mempelajari *deep learning*. *Library* Keras termasuk API bahasa pemrograman *Python* yang merupakan bahasa tingkat tinggi. Dikembangkan secara *open source* sejak dirilis pada awal Maret 2015 dan memungkinkan seseorang untuk bereksperimen secara cepat, dan merupakan bahasa pemrograman

Keras memberi kefleksibilitasan dalam menulis kode dan juga sangat mudah. Keras juga menyediakan beberapa fitur lain sangat berguna dan dapat memperlebar kegunaan aplikasinya. Beberapa kelebihan *Keras* dari beberapa model *deep learning* lain mencakup:

- *Keras* juga menyediakan banyak fitur yang penerapannya tergantung pada kebutuhan pengguna.
- *Keras* membuat pengguna dapat langsung menggunakan model atau memasukkan fitur-fiturnya tanpa membuat model sendiri.
- *Keras* adalah suatu API python yang dibuat mudah dipelajari dan sederhana untuk implementasinya.
- *Keras* menjadi prototipe yang mudah sehingga dapat disebarkan luaskan dalam waktu yang relatif lebih singkat.

C. kaggle.com

Kaggle.com adalah situs tempat Anda dapat bertukar pikiran, mendapatkan inspirasi, bersaing dengan ilmuwan data lainnya, mempelajari pengetahuan baru dan trik pemrograman, serta melihat berbagai contoh aplikasi ilmu data di dunia nyata. Ada banyak data yang dapat digunakan untuk apa saja dari yang sederhana hingga yang lebih kompleks dan data yang cukup penting. Data yang dikandungnya asli dan referensi, sehingga model dapat dilatih dan diuji dalam proyek yang dapat banyak membantu orang. banyak fitur lain yang sangat berguna seperti data, kode, komunitas, inspirasi, kontes, dan kursus. *Kaggle* juga memiliki beberapa manfaat lain yaitu:

2.3.1. Data

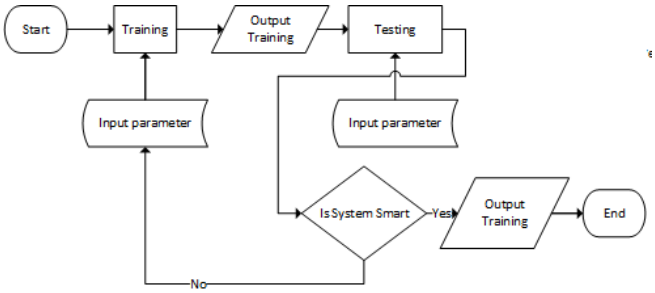
Terdapat beberapa catatan yang digunakan di Kaggle. Di dalamnya Anda dapat melihat daftar kumpulan data dengan mencari nama kumpulan data tertentu yang dikembangkan dalam model. adalah catatan dalam format file CSV, JSON, SQLite, arsip, serta *BigQuery*. Data-data ini dapat digunakan untuk bekerja dengan menggunakan berbagai pengembangan *data science*.

2.3.2. Kode

Terdapat kode yang sangat banyak di dalam situs *Kaggle*. Banyak terdapat contoh kode dari pengguna Kaggle lainnya. Ini memungkinkan pengguna untuk dengan mudah mencari pengguna lain yang berisi kode dan teks. Ini adalah cara yang bagus untuk belajar, berlatih, dan melihat bagaimana orang lain memecahkan masalah serupa. Sebagian besar pengguna *Kaggle* memprogram dengan *Python*, tetapi ada bahasa pemrograman lain seperti *SQLite*, dan *Julia*.

III.METODE PENELITIAN

Metode yang digunakan adalah memberikan *training dataset input*, yaitu memberikan input parameter sampai satu *epoch* (*epoch* adalah ketika seluruh dataset sudah melalui proses training pada Neural Netwok sampai dikembalikan ke awal untuk sekali putaran) [5] kemudian diberikan yang lebih banyak parameter kesistem kemudian melihat output grafiknya.



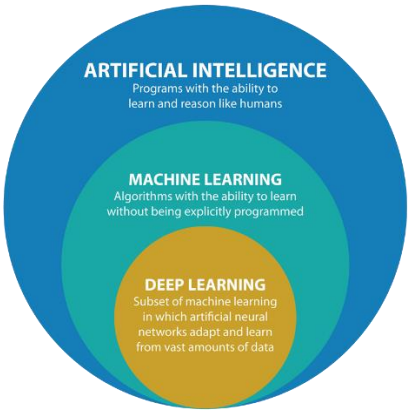
Gambar 1. Flowchart Metode Training Dataset

Era teknologi saat ini hampir semuanya melibatkan *artificial intelligence* (AI) dan *deep learning* (DL) adalah teknik yang paling banyak dipergunakan dalam pendekatan komputasi dalam bidang *Machine Learning* (ML). Banyak aplikasi yang dibentuk dengan melibatkan ML Seperti Kemampuan Long Short Term Memory Machine Learning [6] dalam jurnal yang dilakukan Hastomo,W. Jenis DL yang paling luas pemanfaatannya dalam kehidupan manusia adalah *Convolutional Neural Network* (CNN)[7].

Lapisan yang dalam dibandingkan dengan ML lainnya, menjadikan metode ini masuk dalam kategori DL. Kemampuan untuk dapat membedakan objek image, maka teknik ini lebih dikenal dengan CNN (Gambar 1). Beberapa

JIFOTECH (JOURNAL OF INFORMATION TECHNOLOGY)
Vol. 3, No. 1, Maret 2023

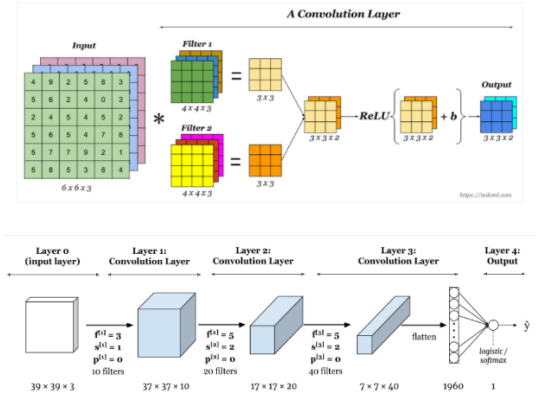
proses dan istilah dalam CNN akan dijelaskan secara singkat pada sub bagian ini.



Gambar 2. Klasifikasi CNN dalam AI

A. *Convolutional Neural Network*

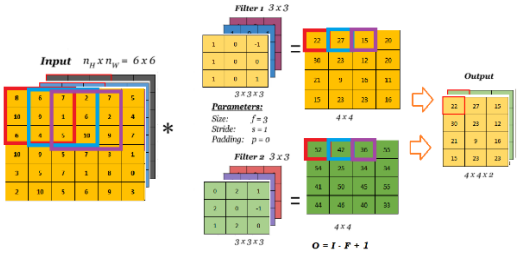
Pemikiran awal CNN adalah jaringan saraf (*neuron*) di manusia dan binatang [8]. CNN terdiri dari tumpukan *convolutional layer* dengan parameter yang berbeda-beda (*filter*, *padding*, *stride* dan lain-lain). Setiap input disusun dalam bentuk matrik tiga dimensi yaitu *height* (tinggi), *width* (lebar) dan *depth* (kedalaman), umumnya nilai *height* sama dengan nilai *width*. Untuk input matrik RGB maka nilai *channel* adalah 3, channel menyatakan nilai depth dalam matrik tersebut. Akhir dari rangkaian lapisan CNN menghasilkan model *kernel*, kemudian dilanjutkan dengan proses *fully connected*. *Fully Connected* bertujuan untuk mengklasifikasikan *image* dari suatu label tertentu. (Gambar 2)



Gambar 3. Proses konvolusi (atas) yang merupakan bagian dari lapisan CNN secara keseluruhan (bawah) [9].

B. *Proses Konvolusi*

Proses konvolusi merupakan perhitungan dot product antara matrik input dan matrik *filter*. Matrik *filter* adalah matrik segi empat (*height=width*) yang merupakan kumpulan nilai bobot (*weight*), kadang juga disebut juga sebagai matrik *kernel*. Ukuran height dan width selalu lebih kecil dari *input*, tapi nilai *depth* (*channel*) sama dengan *depth input*. Dalam proses konvolusi umumnya menggunakan lebih dari satu jenis matrik *kernel* (*filter* 1, *filter* 2, dan setrusnya). Ukuran matrik *output* untuk baris dan kolom dapat digunakan ketentuan O = I - F+1. (Gambar 4)



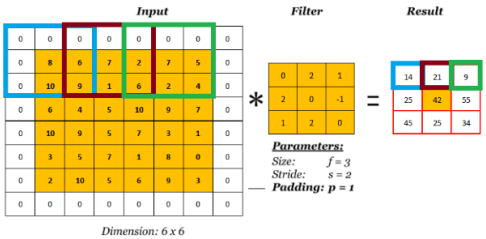
Gambar 4. Detail proses konvolusi

C. *Proses stride*

Proses *stride* sebenarnya adalah bagian dari proses konvolusi itu sendiri. Matrik *filter* akan melakukan *scanning* terhadap matrik *input*, dengan cara melakukan konvo kedua matrik tersebut sesuai ukuran matrik *filter*. Setelah proses pertama selesai dilanjutkan dengan proses konvo kedua dengan menggerakan matrik *input* (matrik *input* ke dua) yang akan di konvo dengan matrik *filter*. Pergeseran matrik input pertama dan kedua adalah sesuai dengan nilai *stride* yang ditentukan sebagai parameter dalam tiap lapisan. Proses ini diulang untuk konvo selanjutnya, sehingga dihasilkan matrik dengan ukuran yang lebih kecil. (Gambar 5)

D. *Proses padding*

Padding adalah proses penambahan nilai yang besar sama disetiap sisi *image input* (biasanya nilai 0 ata satu). Walaupun matrik *input* menjadi berukuran lebih besar dari semula, namun dalam proses konvolusi akan tetap lebih kecil dari matirk *input* aslinya. Tujuan dari *padding* adalah memberikan batasan yang lebih jelas dari suatu gambar, sehingga proses dapat menjadi lebih fokus. *Padding* kadang juga dipergunakan untuk menyesuaikan ukuran matrik *output* dengan cara menambah nilai di tiap sisi gambar sehingga sesuai dengan matrik input yang akan digunakan pada lapisan *convultional* selanjutnya. (Gambar 5)

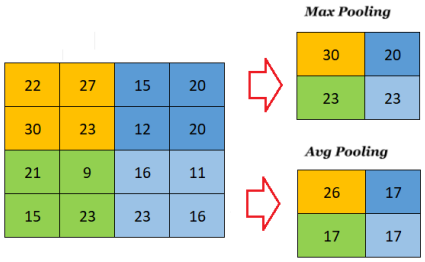


Gambar 5. Stride dan Padding

E. *Pooling*

Sebenarnya proses *pooling* identik dengan pengambilan sampel data dalam statistik, bedanya dalam *pooling* tidak ada informasi hilang. *Pooling* bertujuan untuk memperkecil ukuran matrik agar perhitungan yang dilakukan mesin

menjadi lebih cepat. Terdapat banyak jenis *pooling*, tapi yang umum dipergunakan adalah *max pooling* dan *average pooling*. *Max pooling* adalah mengambil nilai terbesar dalam ukuran matrik tertentu, sedangkan *average pooling* adalah mengambil nilai rata-rata dalam ukuran matrik tertentu.(Gambar 6)



Gambar 6. Proses pooling

F. *Activation Function*

Fungsi aktivasi diberikan setelah *proses pooling*, tujuannya agar nilai yang dihasilkan berada dalam rentang nilai tertentu (biasanya berkisar 0 sampai 1) dan juga berfungsi mengambil keputusan untuk mengambil hasil yang sesuai dan menghilangkan hasil yang tidak diinginkan (kurang baik), memetakan *input* ke *output* yang tidak *linier*, memberikan kemampuan belajar yang lebih komplek. Fungsi aktivasi juga memiliki kemampuan untuk dapat membedakan fitur-fitur yang sangat penting.

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

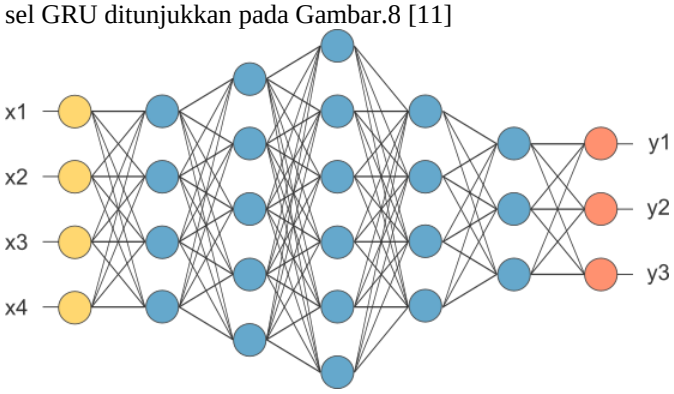
Gambar 7. Jenis fungsi aktivasi [10]

G. *Fully Connected Layer (FC)*

Proses ini berada di akhir tiap lapisan CNN, dinamakan fully connected karena semua *neuron* (*node*) terhubung dengan *neuron* di lapisan sebelum dan sesudahnya. FC berfungsi sebagai klasifikasi, proses ini secara teknis ini tidak lain adalah metode lapisan *neural network conventional* (*feed forward neural network*) [8].

H. *AlgoritmeGated Recurrent Unit (GRU)*

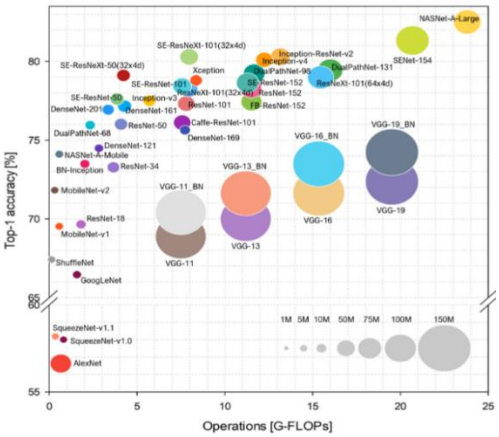
Recurrent Unit (GRU)GRUpertama kali diusulkan pada tahun 2014[24].GRU mirip dengan LSTM, tetapi lebih sederhana untuk dihitung dan diimplementasikan. Struktur sel GRU ditunjukkan pada Gambar.8 [11]



Gambar 8. Jaringan Fully Connected (FC)

I. *Arsitektur CNN*

Selain dari *ImageNet Large Scale Visual Recognition Chalenge* (ILSVRC), terdapat banyak arsitektur CNN yang sudah dibuat untuk meningkatkan akurasi melalui pengujian *input dataset* tertentu yang dijadikan sebagai standar (MNIST, *imagenet*, *coconut* dll) (Gambar 9). Namun banyak arsitektur lebih cenderung menambah lapisan menjadi sangat dalam ketimbang merancang suatu design struktur dengan metode baru. Hal ini mengakibatkan diperlukannya performansi computer yang tinggi untuk dapat menjalankan arstitektur tersebut [12].



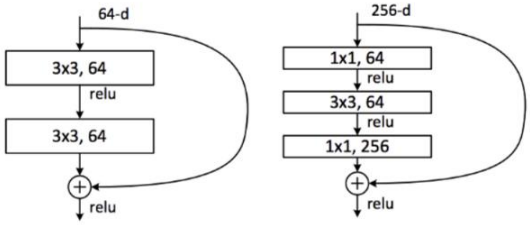
Gambar 9. Komparasi Akurasi Arsitektur CNN Berdasarkan ILSVRC [12]

J. *ResNet-152*

ResNet-152 adalah lapisan dengan kedalaman 152 lapisan. Ini merupaka suatu model dengan jaringan lapisan dalam yang cenderung akan menimbulkan *overftting* dan *vanishing gradient*. Para pengembangnya pada tahun 2015 telah mencoba mempergunakan teknik *residual blok* dan

JIFOTECH (JOURNAL OF INFORMATION TECHNOLOGY)
Vol. 3, No. 1, Maret 2023

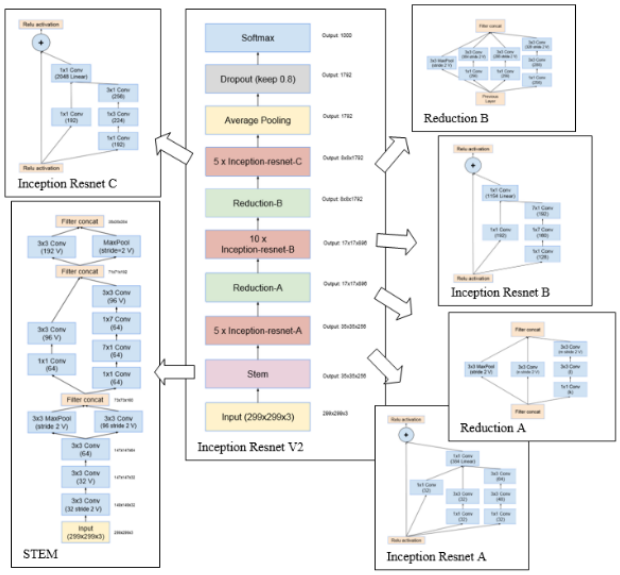
bottleneck untuk mengatasi efek *overfitting* tersebut. (Gambar 10)



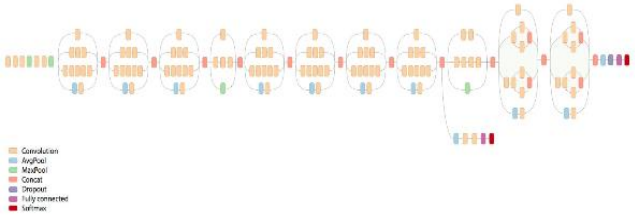
Gambar 10. Blok Residual (kiri) dan Struktur Bottleneck (kanan) [13]

K. *InceptionResNet-V2*

Sebagai upaya pengembangan CNN, Szegedy pada tahun 2014 memperkenalkan *Inception* dalam penelitiannya yang berjudul "*Going Deeper with Convolution*". *Convolution* yang merupakan upaya ektraksi gambar untuk memperoleh model dalam bentuk matrik *kernel*. Proses ini dijalankan dengan melakukan suatu penyaringan yang kemudian akan bergeser dengan nilai *stride* tertentu pada suatu gambar masukan. kemudian hasil yang didapat dari *convolution* dapat menjadi *input dan merupakan* bagian *fully connected* untuk proses klasifikasinya.



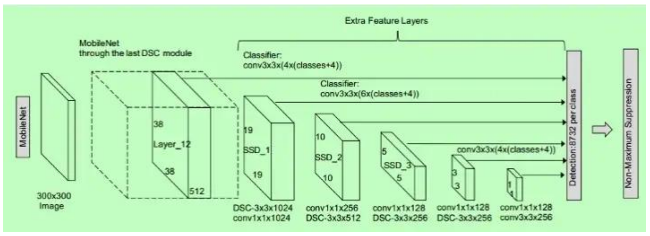
Gambar 11. Blok Modul *Inception Resnet V2*



Gambar 12. Diagram Keseluruhan *Inception Resnet V2* [13]

L. *MobileNet-V2*

Penggunaan *feature inverted residual* blok dan *bottleneck* adalah yang membedakan *mobilenet V2* ini dengan *mobilenet* aslinya. Fitur ini mampu menurunkan jumlah banyaknya perhitungan yang dilakukan mesin dibanding mobilenet asli. Dukungan *mobilenet* versi 2 untuk menerima *image input* lebih dari 32x32, memberikan kemampuan lebih baik untuk gambar berukuran besar [13]. Karena *mobilenet ini* dirancang untuk aplikasi *mobile*, maka arsitekturnya diharapkan mampu berjalan dengan kemampuan komputer yang seminim mungkin namun tetap dapat memberikan tingkat akurasi yang tinggi.

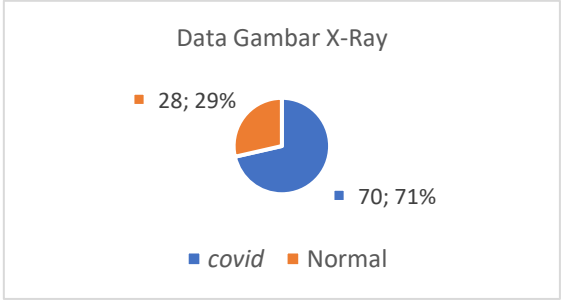


Gambar 13. Arsitektur Mobilenet [13]

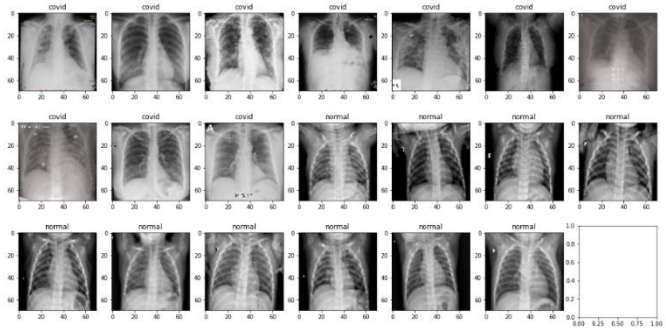
M. *Preparasi Data dan Tahap Penelitian*

Penelitian ini dimulai dengan mengunduh data dari situs [14], sebagai bahan yang akan diteliti dengan bahasa pemrograman *pyton* yang menggunakan arsitektur *Convolution Neural Network* (CNN). Dengan total gambar yaitu 70 untuk gambar *x-ray* paru-paru yang menderita *covid* dan 28 untuk gambar *x-ray* paru paru normal.

Jadi data tiap kelas terbagi 71% (70 gambar) untuk data paru-paru *covid* dan 29% (28 gambar) untuk paru-paru normal. (Gambar 14)



Gambar 14. Pie Dataset Gambar X-Ray



Gambar 15. Gambar X-Ray Paru-Paru

Dataset input adalah total 98 gambar dan dipergunakan sebagai data latih dan juga data validasi untuk arsitektur CNN ini, hasil dari latihan tadi dapat menjadikan model yang dapat dipergunakan untuk mengklasifikasi gambar.

IV.HASIL DAN PEMBAHASAN

Hasil dari proses *training* dengan *data input* diperoleh nilai *train accuracy*, *train loss*, *val accuracy* dan *val loss* dari keempat arsitektur, sebagai berikut:

Pada waktu menjalankan bahasa pemrograman python terlihat prosentase untuk *loss accuracy, validation loss* dan *validation accuracy* semakin lama semakin baik. Yang artinya mesin dapat belajar dengan baik setelah semakin banyak berlatih untuk mengklasifikasi mana gambar paru-paru yang menderita *covid* dan mana yang tidak. Sehingga pada akhir pengulangan pada perputaran latihannya mesin sudah mendapatkan akurasi 87,88% untuk tingkat keberhasilannya mengidentifikasi antara gambar paru paru *covid* dengan yang normal. Berikut adalah *coding* untuk melatih mesin di *python*

```
#TRAINING
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

sedangkan hasilnya dapat terlihat sebagai berikut :

```
Epoch 11/25
3/3 [==============================] - 0s 137ms/step - loss: 0.6006 - accuracy: 0.7846 - val_loss: 0.5439 - val_accuracy: 0.7273
Epoch 12/25
3/3 [==============================] - 0s 129ms/step - loss: 0.5873 - accuracy: 0.7077 - val_loss: 0.5208 - val_accuracy: 0.7273
Epoch 13/25
3/3 [==============================] - 0s 130ms/step - loss: 0.5489 - accuracy: 0.7385 - val_loss: 0.4965 - val_accuracy: 0.7273
Epoch 14/25
3/3 [==============================] - 0s 136ms/step - loss: 0.4949 - accuracy: 0.7538 - val_loss: 0.4128 - val_accuracy: 0.7879
Epoch 15/25
3/3 [==============================] - 0s 133ms/step - loss: 0.5756 - accuracy: 0.7385 - val_loss: 0.3518 - val_accuracy: 0.8182
Epoch 16/25
3/3 [==============================] - 0s 141ms/step - loss: 0.5233 - accuracy: 0.7077 - val_loss: 0.3284 - val_accuracy: 0.8182
Epoch 17/25
3/3 [==============================] - 0s 132ms/step - loss: 0.3868 - accuracy: 0.8000 - val_loss: 0.2915 - val_accuracy: 0.8182
Epoch 18/25
3/3 [==============================] - 0s 139ms/step - loss: 0.2809 - accuracy: 0.8308 - val_loss: 0.3291 - val_accuracy: 0.9091
Epoch 19/25
3/3 [==============================] - 0s 145ms/step - loss: 0.5373 - accuracy: 0.6923 - val_loss: 0.3567 - val_accuracy: 0.9091
Epoch 20/25
3/3 [==============================] - 0s 143ms/step - loss: 0.3412 - accuracy: 0.8923 - val_loss: 0.2218 - val_accuracy: 0.8788
Epoch 21/25
3/3 [==============================] - 0s 131ms/step - loss: 0.3676 - accuracy: 0.8000 - val_loss: 0.4488 - val_accuracy: 0.7576
Epoch 22/25
3/3 [==============================] - 0s 132ms/step - loss: 0.4916 - accuracy: 0.8154 - val_loss: 0.3905 - val_accuracy: 0.7879
Epoch 23/25
3/3 [==============================] - 0s 137ms/step - loss: 0.4288 - accuracy: 0.7846 - val_loss: 0.4094 - val_accuracy: 0.8485
Epoch 24/25
3/3 [==============================] - 0s 131ms/step - loss: 0.3913 - accuracy: 0.8308 - val_loss: 0.2378 - val_accuracy: 0.9091
Epoch 25/25
3/3 [==============================] - 0s 135ms/step - loss: 0.1812 - accuracy: 0.9077 - val_loss: 0.1718 - val_accuracy: 0.8788
Accuracy: 87.88%
```

Gambar 16. Mesin Berlatih dalam Python Dengan Parameter Pertama

Didalam pelatihan ini terdapat beberapa parameter yang dipelajari oleh mesin dan dibuat dengan *coding* dibawah ini :

```
# COMPILE
from keras.optimizers import SGD
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
```

paramater-parameter yang dipelajari adalah berjumlah 1.059.106 dan dapat terlihat didalam gambar sebagai berikut :

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_11 (Conv2D)          (None, 68, 68, 32)        896

 max_pooling2d_11 (MaxPoolin  (None, 34, 34, 32)       0
 g2D)

 conv2d_12 (Conv2D)          (None, 32, 32, 32)        9248

 max_pooling2d_12 (MaxPoolin  (None, 16, 16, 32)       0
 g2D)

 dropout_8 (Dropout)         (None, 16, 16, 32)        0

 flatten_4 (Flatten)         (None, 8192)              0

 dense_11 (Dense)            (None, 128)               1048704

 dropout_9 (Dropout)         (None, 128)               0

 dense_12 (Dense)            (None, 2)                 258

=================================================================
Total params: 1,059,106
Trainable params: 1,059,106
Non-trainable params: 0
_________________________________________________________________
```

Gambar 17. Parameter Latihan Dalam Python

Sedang didalam *python*, modul *sklearn* dapat dipergunakan untuk mengukur akurasi dan memvisualisasikan matrik *cofusion* seperti yang ditampilkan dalam Gambar 18.

Dengan *coding* sebagai berikut :

```
#CLASSIFICATION REPORT
from sklearn.metrics import classification_report
print(classification_report(y_test_n,y_prediction_n, target_names=["COVID","NORMAL"]))
```

JIFOTECH (JOURNAL OF INFORMATION TECHNOLOGY)
Vol. 3, No. 1, Maret 2023

	precision	recall	f1-score	support
COVID	0.92	0.92	0.92	24
NORMAL	0.78	0.78	0.78	9
accuracy			0.88	33
macro avg	0.85	0.85	0.85	33
weighted avg	0.88	0.88	0.88	33

Gambar 18. Report Klasifikasi Dalam Python

Dari hasil pembelajaran dan validasi dari sisi mesin dapat terlihat grafik kenaikan ke mahiran mesin untuk mempelajari dua kategori gambar antara gambar paru paru yang terkena *covid* dan yang normal. Didalam bagian ini digunakan *coding* sebagai berikut :

```
model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(16,(3,),activation='relu',input_shape=(150,150,3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dropout(0.3, seed=112),
    tf.keras.layers.Dense(500, activation='relu'),
    tf.keras.layers.Dropout(0.5, seed=112),
    tf.keras.layers.Dense(2, activation='sigmoid')])

model.compile(loss='categorical_crossentropy',
optimizer= 'Adam',metrics=['accuracy'])

history =  model.fit(
        train_generator,
        steps_per_epoch = 6,
        epochs = 25,
        validation_data = val_generator,
        validation_steps = 1,
        verbose = 1,
        )


%matplotlib inline
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']


epochs = range(len(acc))
plt.plot(epochs, acc, 'r',label ='Training Accuracy')
plt.plot(epochs, val_acc, 'b',label ='Validation Accuracy')
plt.title('Training and Validation')
plt.legend(loc = 'best')
plt.show()

plt.plot(epochs, loss, 'r',label ='Training Loss')
plt.plot(epochs,val_loss, 'b',label ='Validation Loss')
plt.title('Training and Validation')
plt.legend(loc='best')
plt.show()
```

Pada gambar. 19 terilihat bahwa semakin banyak program melakukan trainingnya semakin akurat juga hasil keluarannya.

Gambar 19. Grafik Akurasi Trainning dan Testing

Dari pembelajaran ini pula mesin semakin jarang untuk melakukan kesalahan dalam pemilihan kategori gambar Dapat terlihat di Gambar. 20 berikut ini.

Gambar 20. Grafik Loss Value Trainning dan Testing

Dari dua buah grafik diatas dapat dilihat keterkaitan antara keduanya yang menunjukan tingkat akurasi (accuration) dan kesalahan (loss). Untuk pengambilan data akurasi dan

kehilangan akurasi diatas diambil parameter kedua yang lebih banyak lagi yaitu sebesar 3.824.486.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_13 (Conv2D)          (None, 148, 148, 16)      448

 max_pooling2d_13 (MaxPoolin  (None, 74, 74, 16)       0
 g2D)

 conv2d_14 (Conv2D)          (None, 72, 72, 32)        4640

 max_pooling2d_14 (MaxPoolin  (None, 36, 36, 32)       0
 g2D)

 conv2d_15 (Conv2D)          (None, 34, 34, 64)        18496

 max_pooling2d_15 (MaxPoolin  (None, 17, 17, 64)       0
 g2D)

 flatten_5 (Flatten)         (None, 18496)             0

 dense_13 (Dense)            (None, 200)               3699400

 dropout_10 (Dropout)        (None, 200)               0

 dense_14 (Dense)            (None, 500)               100500

 dropout_11 (Dropout)        (None, 500)               0

 dense_15 (Dense)            (None, 2)                 1002

=================================================================
Total params: 3,824,486
Trainable params: 3,824,486
Non-trainable params: 0
```

Gambar 21. Parameter Loss Value Trainning dan Testing

Tetapi dengan banyaknya parameter tersebut membuat mesin lebih baik lagi dalam belajar bahkan ada yang sampai keakuratannya mencapai 100% dapat terlihat di grafiknya dan didalam gambar. 22 berikut

```
6/6 [==============================] - 5s 784ms/step - loss: 0.2765 - accuracy: 0.8793 - val_loss: 0.2672 - val_accuracy: 0.8000
Epoch 8/25
6/6 [==============================] - 5s 760ms/step - loss: 0.3135 - accuracy: 0.8621 - val_loss: 0.1332 - val_accuracy: 1.0000
Epoch 9/25
6/6 [==============================] - 5s 779ms/step - loss: 0.4784 - accuracy: 0.7759 - val_loss: 0.1391 - val_accuracy: 1.0000
Epoch 10/25
6/6 [==============================] - 5s 803ms/step - loss: 0.2950 - accuracy: 0.8833 - val_loss: 0.2643 - val_accuracy: 0.9000
Epoch 11/25
6/6 [==============================] - 5s 766ms/step - loss: 0.3235 - accuracy: 0.9167 - val_loss: 0.3159 - val_accuracy: 0.9000
Epoch 12/25
6/6 [==============================] - 5s 829ms/step - loss: 0.2848 - accuracy: 0.8500 - val_loss: 0.2021 - val_accuracy: 0.9000
Epoch 13/25
6/6 [==============================] - 4s 716ms/step - loss: 0.3289 - accuracy: 0.8621 - val_loss: 0.3967 - val_accuracy: 0.8000
Epoch 14/25
6/6 [==============================] - 5s 774ms/step - loss: 0.2201 - accuracy: 0.9310 - val_loss: 0.1470 - val_accuracy: 0.9000
Epoch 15/25
6/6 [==============================] - 4s 697ms/step - loss: 0.2890 - accuracy: 0.8621 - val_loss: 0.0376 - val_accuracy: 1.0000
Epoch 16/25
6/6 [==============================] - 5s 825ms/step - loss: 0.3240 - accuracy: 0.8793 - val_loss: 0.0472 - val_accuracy: 1.0000
Epoch 17/25
6/6 [==============================] - 4s 755ms/step - loss: 0.2710 - accuracy: 0.8793 - val_loss: 0.2174 - val_accuracy: 0.9000
Epoch 18/25
6/6 [==============================] - 5s 847ms/step - loss: 0.1727 - accuracy: 0.9483 - val_loss: 0.1557 - val_accuracy: 0.9000
Epoch 19/25
6/6 [==============================] - 4s 761ms/step - loss: 0.1115 - accuracy: 0.9655 - val_loss: 0.1564 - val_accuracy: 0.9000
Epoch 20/25
6/6 [==============================] - 5s 824ms/step - loss: 0.2341 - accuracy: 0.8966 - val_loss: 0.1420 - val_accuracy: 0.9000
Epoch 21/25
6/6 [==============================] - 5s 740ms/step - loss: 0.1347 - accuracy: 0.9310 - val_loss: 0.1063 - val_accuracy: 1.0000
Epoch 22/25
6/6 [==============================] - 4s 737ms/step - loss: 0.1416 - accuracy: 0.9138 - val_loss: 0.1585 - val_accuracy: 0.9000
Epoch 23/25
6/6 [==============================] - 5s 733ms/step - loss: 0.1297 - accuracy: 0.9138 - val_loss: 0.2016 - val_accuracy: 0.9000
Epoch 24/25
6/6 [==============================] - 4s 726ms/step - loss: 0.1501 - accuracy: 0.9310 - val_loss: 0.0247 - val_accuracy: 1.0000
Epoch 25/25
6/6 [==============================] - 5s 797ms/step - loss: 0.1191 - accuracy: 0.9167 - val_loss: 0.1101 - val_accuracy: 1.0000
```

Gambar 22. Mesin Berlatih dengan Parameter Kedua untuk

V.KESIMPULAN

Hasil dari ini didapat kesimpulan yaitu;

a. Hasil penelitian dapat dilihat bahwa semakin banyak mesin belajar dengan kemampuan kecerdasan buatannya maka mesin tersebut akan semakin mahir untuk mengerjakan pemilihannya dapat terlihat dari banyaknya parameter yang dipelajari 1.059.106, maka mesin dapat mengklasfikasikan 87,88%. Akan tetapi bila mesin belajar dengan parameter yang lebih banyak lagi yaitu sebesar 3.824.486, maka mesin dapat menebak mendekati 100%.
b. Penggunaan bahasa pemrograman *python* dengan modul *keras* dan *tensorflow* dianggap cukup efektif untuk membuat kecerdasan buatannya (*Artificial intelligent*) dapat mempelajari parameter-parameter yang ada.
c. Pengambilan data di *www.kaggle.com* juga termasuk baik karena tersedia berbagai macam data dan juga dilengkapi dengan teknologi mesin yang dapat belajar didalam *website*-nya.
d. Dikarenakan jumlah gambar *x-ray* paru-paru yang dipakai hanya berjumlah total 98 buah, dapat dianggap kurang memadai sebagai pembelajaran mesin yang ada. Diharapkan dalam penelitian kedepan datanya dapat diperbanyak kembali
e. Didalam iterasi pembelajaran mesin dengan data gambar ini dapat terlihat semakin banyak parameter untuk belajar maka semakin banyak juga waktu yang digunakan untuk parameter sebanyak 3.824.486. waktu yang diperlukan adalah rata-rata 780 ms sedangkan untuk parameter sebanyak 1.059.106, waktu yang diperlukan adalah rata-rata 135 ms.

REFERENSI

[1] N. Van Hieu and N. L. H. Hien, “Recognition of plant species using deep convolutional feature extraction,” *Int. J. Emerg. Technol.*, vol. 11, no. 3, pp. 904–910, 2020.

[2] A. S. B. Karno, W. Hastomo, & Arif, D., and E. S. Moreta, “Optimasi Portofolio Dan Prediksi Cryptocurrency Menggunakan Deep Learning Dalam Bahasa Python,” vol. 4, no. September, 2020.

[3] W. Hastomo and A. Satyo, “Long Short Term Memory Machine Learning Untuk Memprediksi Akurasi Nilai Tukar IDR Terhadap USD,” *Proseding SenTIK*, vol. 3, 2019.

[4] I. Bakti and M. Firdaus, “Arsitektur CNN InceptionResNet-V2 Untuk Pengelompokan Pneumonia Chest X-Ray,” *J. Komput. dan Teknol.*, vol. 1, no. 2, pp. 35–42, Jan. 2023, doi: 10.58290/jukomtek.v1i2.66.

[5] W. Hastomo, A. Satyo, B. Karno, and N. Kalbuana,

JIFOTECH (JOURNAL OF INFORMATION TECHNOLOGY)
Vol. 3, No. 1, Maret 2023

"Characteristic Parameters of Epoch Deep Learning to Predict Covid-19 Data in Indonesia," *J. Phys. Conf. Ser.*, 2021, doi: 10.1088/1742-6596/1933/1/012050.

[6] W. Hastomo and A. Satyo, "Kemampuan Long Short Term Memory Machine," vol. 4, no. September, pp. 229–236, 2020.

[7] L. M. R. Rere, R. Dalam, and K. Baru, "Studi Pengenalan Ekspresi Wajah Berbasis Convolutional Neural Network," vol. 3, 2019.

[8] L. Alzubaidi *et al.*, *Review of Deep Learning Concepts, CNN Architectures, Challenges, Applications, Future Directions*, vol. 08, no. 01. Springer International Publishing, 2021.

[9] B. Prijono, "Student Notes: Convolutional Neural Networks (CNN) Introduction," *Indo Machine Learning*, 2018. .

[10] Sebastian Raschka, "Activation Functions for Artificial Neural Networks," *github.io*, 2020. .

[11] A. S. B. Karno and W. Hastomo, "Optimalisasi Data Terbatas Prediksi Jangka Panjang Covid-19 Dengan Kombinasi Lstm Dan GRU," *SeNTIK*, vol. 04, no. September, pp. 181–191, 2020.

[12] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of Representative Deep Neural Network Architectures," *IEEE Access*, vol. 06, pp. 64270–64277, 2018, doi: 10.1109/ACCESS.2018.2877890.

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

[14] "COVID-19 Patients Lungs X Ray Images 10000," *kaggle*, 2020. https://www.kaggle.com/datasets/nabeelsajid917/covid-19-x-ray-10000-images.