KLASIFIKASI DEEP LEARNING UNTUK COVID-19 DENGAN CITRA CXR

Rafi Alvanzah

TABLE OF CONTENTS

01.

PENDAHULUAN

PENDAHULUAN

General

COVID-19 adalah penyakit virus yang menyebabkan pneumonia serius dan berdampak pada berbagai bagian tubuh dari ringan hingga parah tergantung pada sistem imunitas pasien.

Gejala

Terdapat beberapa gejala yang biasa ditimbulkan antara lain: demam, batuk, kehilangan rasa serta bau, sakit tenggorokan, bahkan hingga kesulitan bernafas dan nyeri dada.

Pendeteksian

Seiring dengan meningkatnya kasus COVID-19 ini meningkat pula kebutuhan mendeteksi infeksi dengan cepat dan mudah.

PENDAHULUAN

Penggunaan X-ray

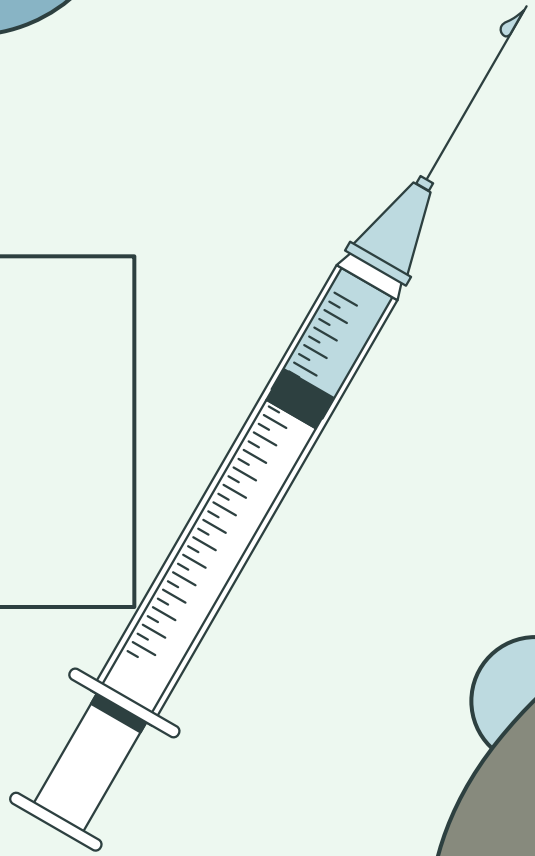
Menggunakan teknik *radiology imaging* (seperti X-ray dan CT scan) terbukti dapat membantu mendeteksi COVID-19 karena gambar X-ray dan CT scan yang dapat memberikan informasi penting tentang penyakit yang disebabkan COVID-19.

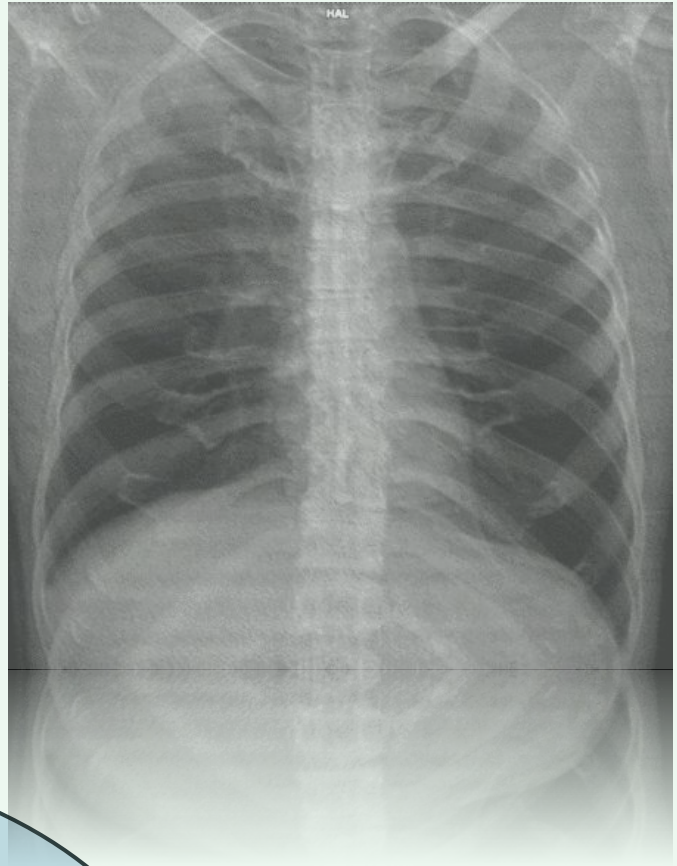
Penggunaan CNN

Teknik penambangan data (data mining) dan machine learning seperti Convolutional Neural Network (CNN) dapat diterapkan bersama dengan gambar sinar-X dan CT scan paru-paru untuk mendeteksi penyakit yang akurat dan cepat, membantu mengurangi masalah kelangkaan pengujian kit.

02.

DATASET





COVID-19 Radiography Database

https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database

COVID-19 images: 3616

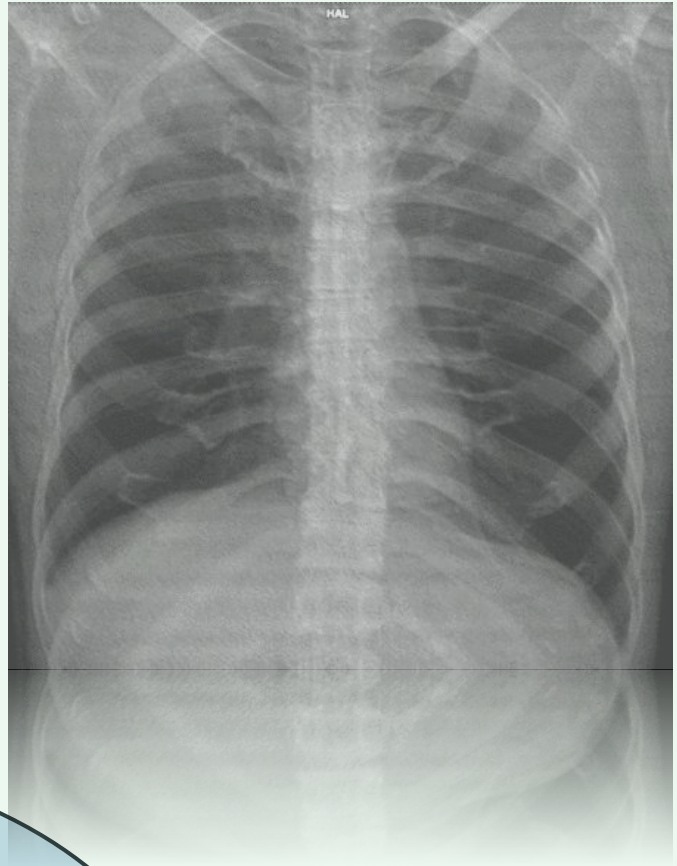
Normal images: 10192

Lung opacity images: 6012

Viral Pneumonia images: 1345

Format: Portable Network Graphics (PNG) file format

Resolution: 299*299 pixels



COVID-19 Radiography Database

https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database

Dilakukan SMOTE (Synthetic Minority Over-sampling Technique) pada data minoritas dan juga random undersampling pada data mayoritas agar tidak terjadinya imbalance dataset.

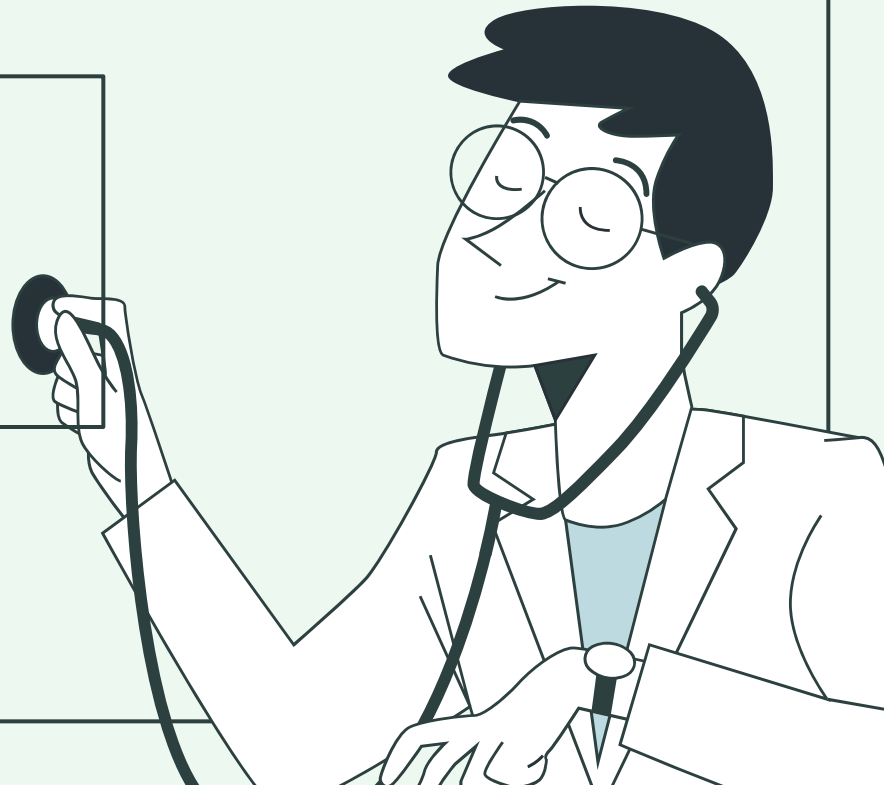
Data yang didapat:

Normal images: 5707

COVID-19 images: 5707

03.

METODE



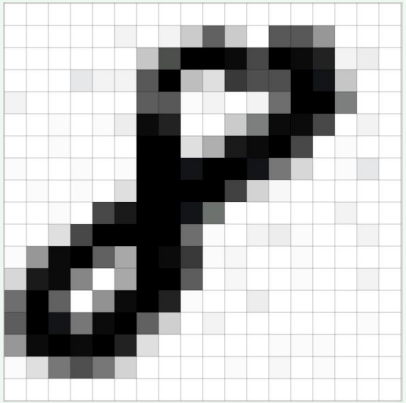
PREPROCESSING

GRAYSCALE

Grayscale adalah sebuah proses yang mengubah gambar yang terdiri dari 3-channel (RGB) menjadi 1-channel.

$$G_{Luminance} = (0.3 * R) + (0.59 * G) + (0.11 * B)$$

VEKTORISASI



NORMALISASI

Normalisasi adalah teknik *scaling* dimana nilai seluruh atribut diubah menjadi [0, 1].

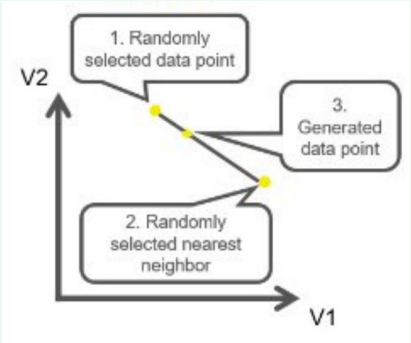
Resampling

Random Undersampling

Random Undersampling adalah teknik resampling yang dilakukan pada data mayoritas dengan cara menghapus data mayoritas secara acak.

SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) adalah teknik resampling dengan cara membentuk data sintetis baru. Teknik ini akan dilakukan pada data minoritas



Convolutional Neural Network

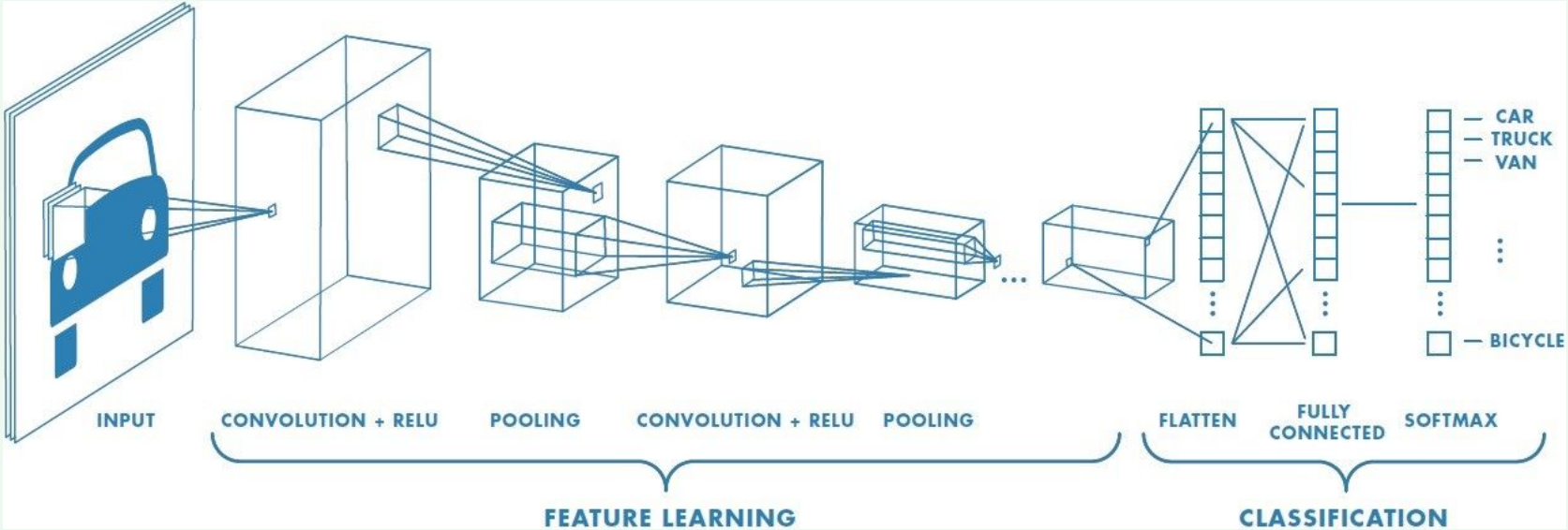
Convolutional Neural Network (CNN) merupakan versi regularisasi Multi Layer Perceptron dan tergolong kedalam deep feedforward artificial neural network. CNN terinspirasi dari proses biologis dimana pola konektivitas antar neuron menyerupai visual cortex pada binatang. Oleh karena itu, CNN banyak diterapkan pada analisis citra.

Arsitektur CNN terdiri atas :

- Input layer
- Output layer
- Hidden layer (Convolutional layer, Pooling layer, Normalization layer, Activation layer (umumnya ReLU), Fully connected layer, dan Loss layer)

Input layer pada CNN berupa tensor, yang memiliki lebar, tinggi dan kedalaman. Hal inilah yang membuat CNN dapat digunakan untuk analisis citra, karena kita tau citra digital memiliki lebar dan tinggi serta memiliki kedalaman dalam bentuk saluran warna merah, hijau dan biru (RGB).

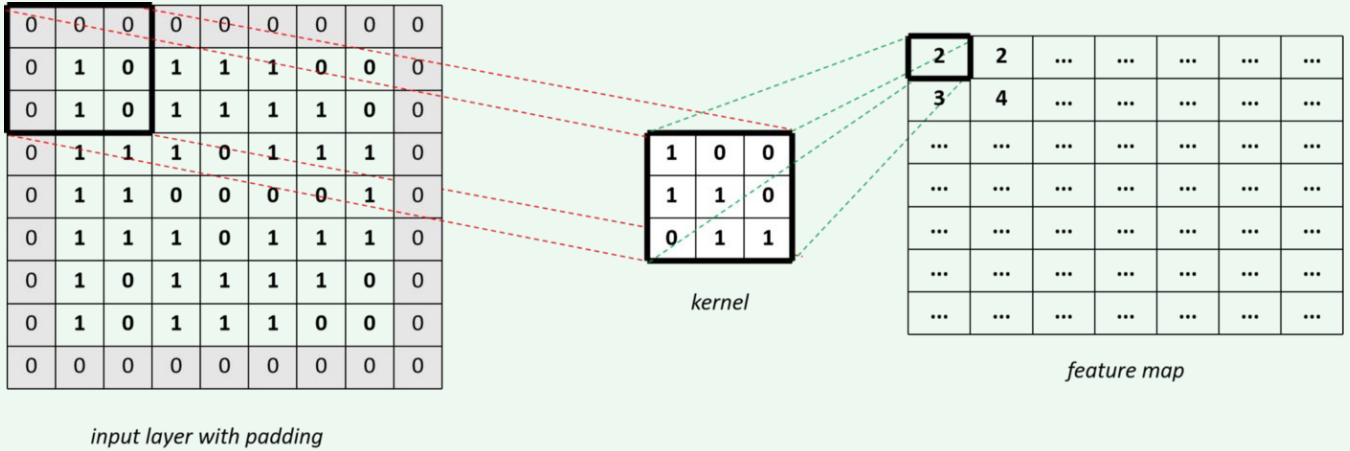
Convolutional Neural Network



Convolutional Layer

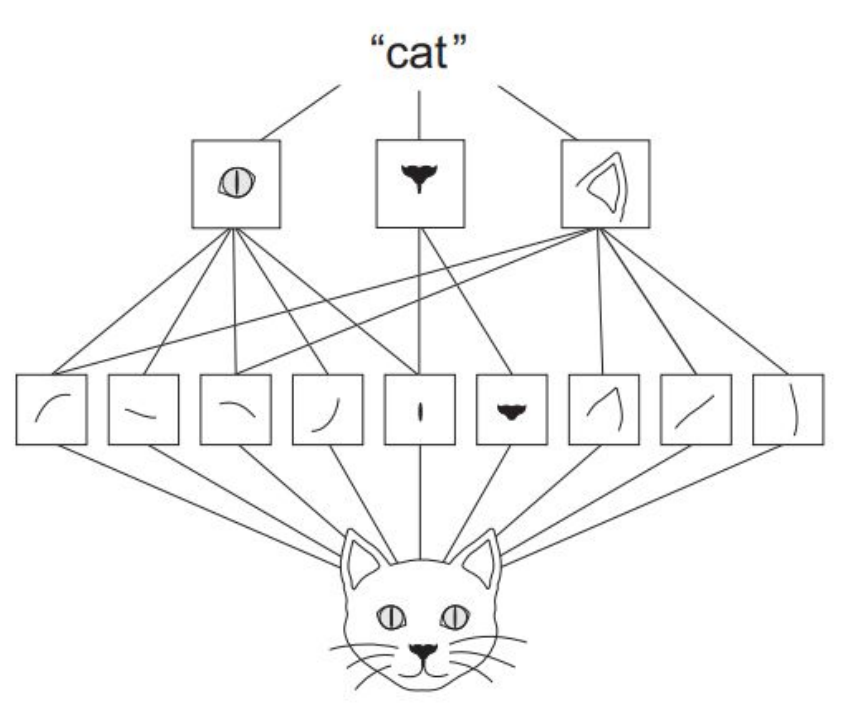
Convolutional layer menerima input berupa 3D tensor (height, width, depth) dan mengeluarkan output berupa 3D tensor (height, width, filter).

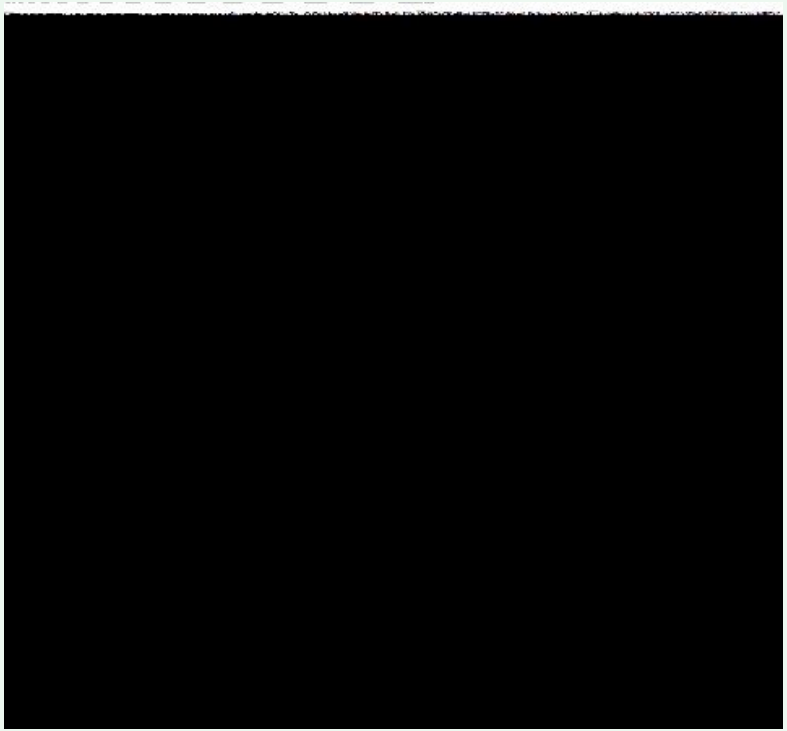
Cara kerja convolutional layer:



Kernel-Padding-Strides

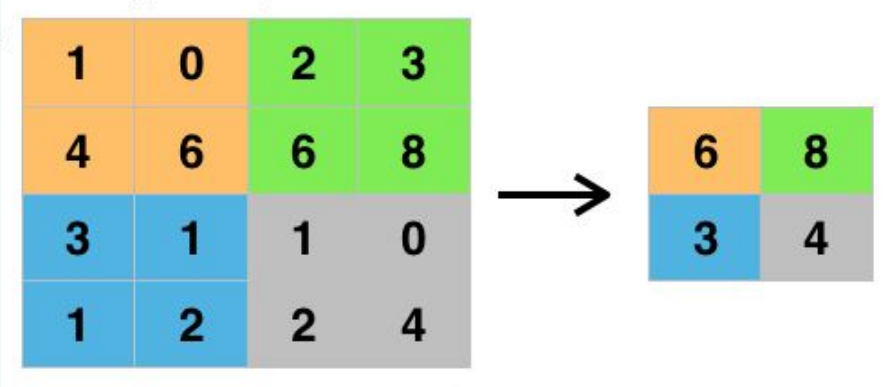
Convolutional Layer





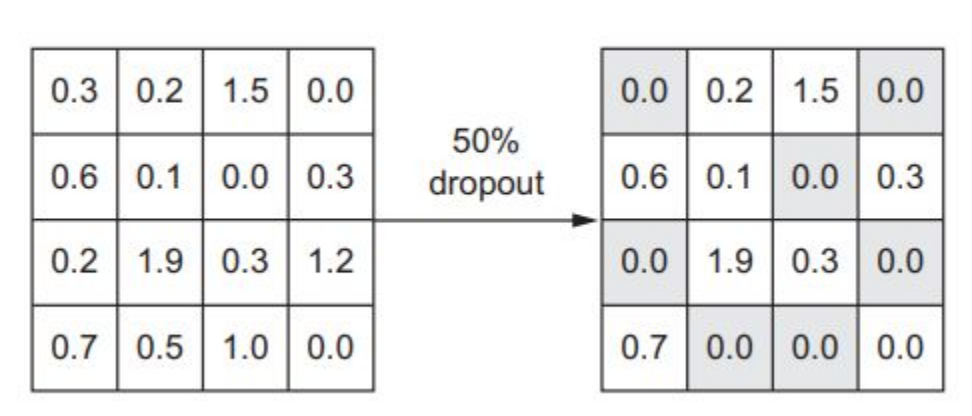
Pooling Layer

Pooling layer digunakan untuk proses reduksi sample (down-sampling). Keuntungan menggunakan pooling layer, kita dapat merepresentasikan data menjadi lebih kecil, mudah dikelola dan mudah mengontrol overfitting. Ada beberapa teknik yang dapat digunakan, diantaranya max pooling , L2-norm pooling dan average pooling.



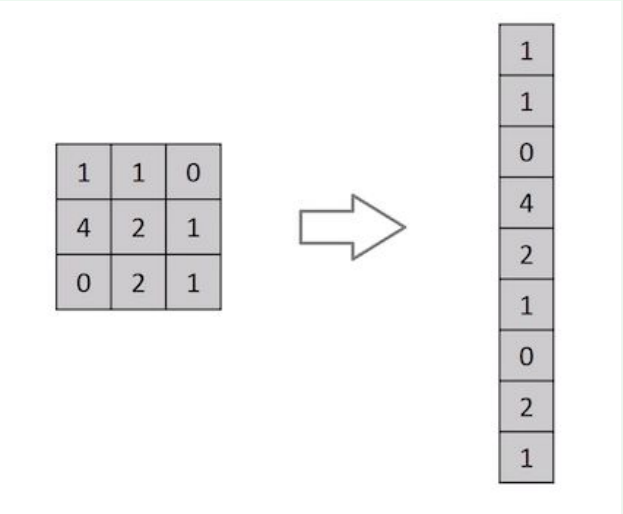
Dropout Layer

Salah satu cara paling efektif dalam mencegah overfitting adalah dengan menambah dropout layer, ia berfungsi sebagai penambah noise pada feature map agar network tidak mempelajari pola-pola yang dapat menciptakan overfitting.



Flatten Layer

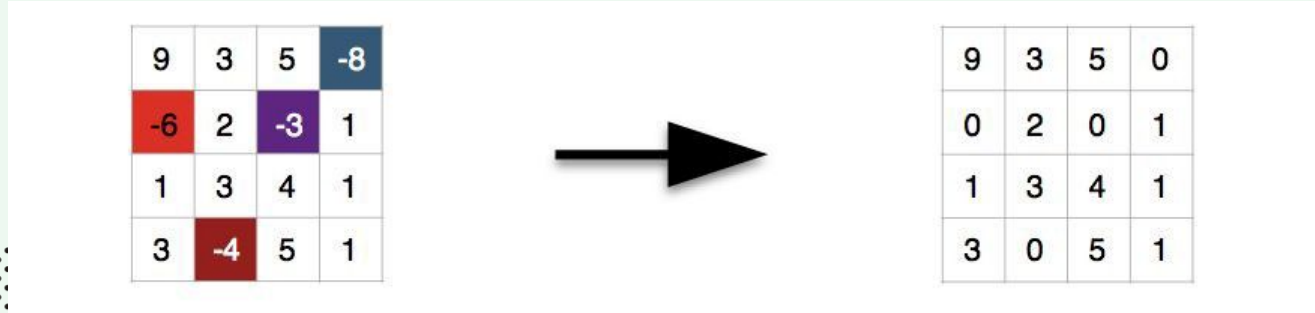
Flatten layer berfungsi untuk mengubah feature map menjadi 1D agar bisa menjadi input untuk fully connected layer (dense layer)



Fungsi Aktivasi

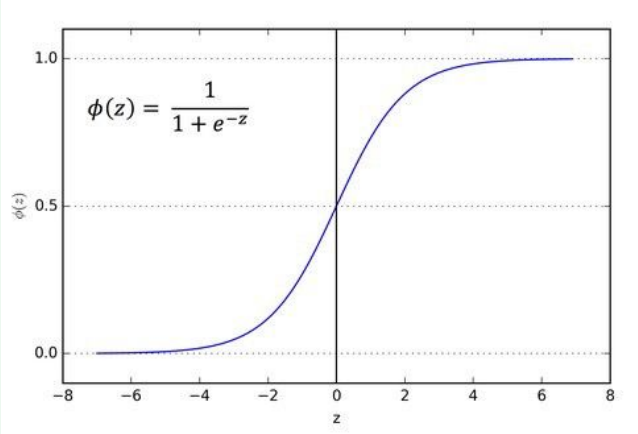
ReLU atau Rectified Activation Function merupakan non-saturating activation function yang dapat meningkatkan non-linearitas decision function dan network secara keseluruhan, tanpa harus mempengaruhi bidang-bidang reseptif pada convolution layer. ReLU sangat efektif untuk menghapus nilai negatif pada feature/activation map dan dijadikan 0. Dirumuskan sebagai,

$f(x) = \max(0,x)$



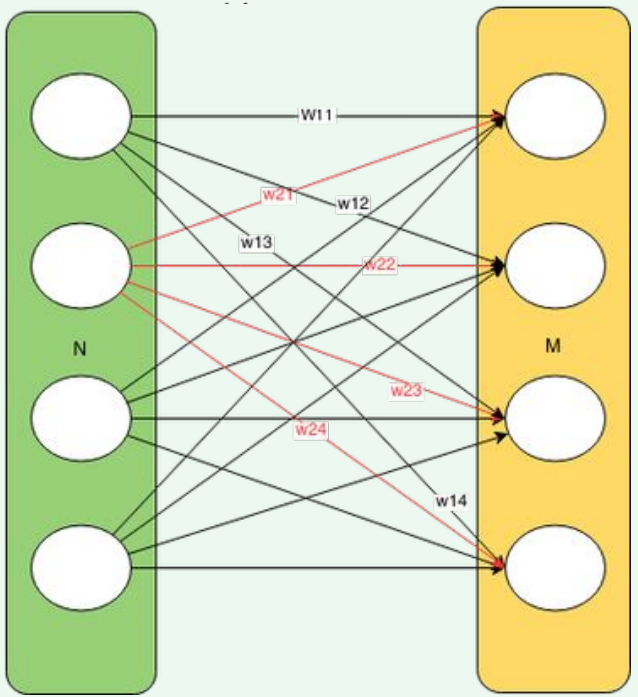
Fungsi Aktivasi

Fungsi aktivasi sigmoid disebut juga fungsi logistik. Ini adalah fungsi yang sama digunakan dalam algoritma klasifikasi regresi logistik. Fungsi mengambil nilai riil apa pun sebagai nilai input dan output dalam rentang 0 hingga 1. Fungsi digunakan untuk model dimana kita harus memprediksi probabilitas sebagai output. Karena probabilitas sesuatu hanya ada antara kisaran 0 dan 1.



Fully Connected Layer

Layer ini memiliki hidden layer, activation function, output layer, dan loss function. Layer ini adalah layer yang biasanya digunakan dalam penerapan multi layer perceptron dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear.



Optimizer (Adam)

Adam (“Adaptive moment estimation”) bekerja dengan menggunakan estimasi gradien momen pertama dan kedua untuk mengadaptasi learning rate untuk setiap bobot jaringan saraf.

Loss Function (SCCE)

Sparse Categorical Cross Entropy (SCCE) adalah fungsi yang menghitung jarak antara output dengan memperhatikan jumlah kelas (2>) yang membedakan dengan Categorical Cross Entropy (CCE) adalah pada SCCE tidak perlu dilakukan one hot encoding dan proses komputasi SCCE lebih murah.

Precision

Merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif.

Precision = (TP) / (TP+FP)

Recall

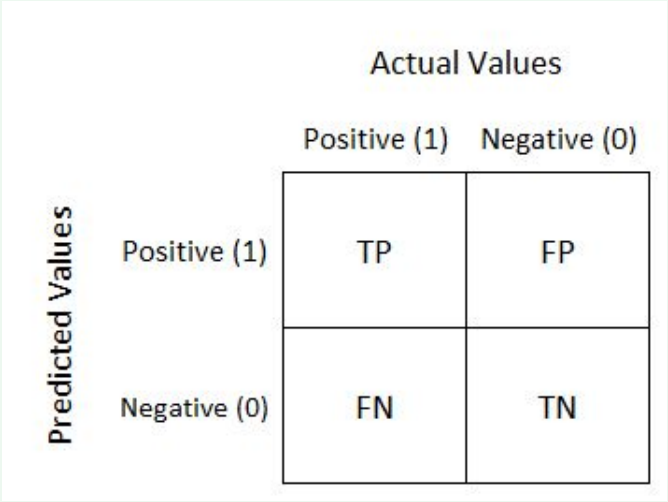
Merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif.

Recall = (TP) / (TP + FN)

F1 Score

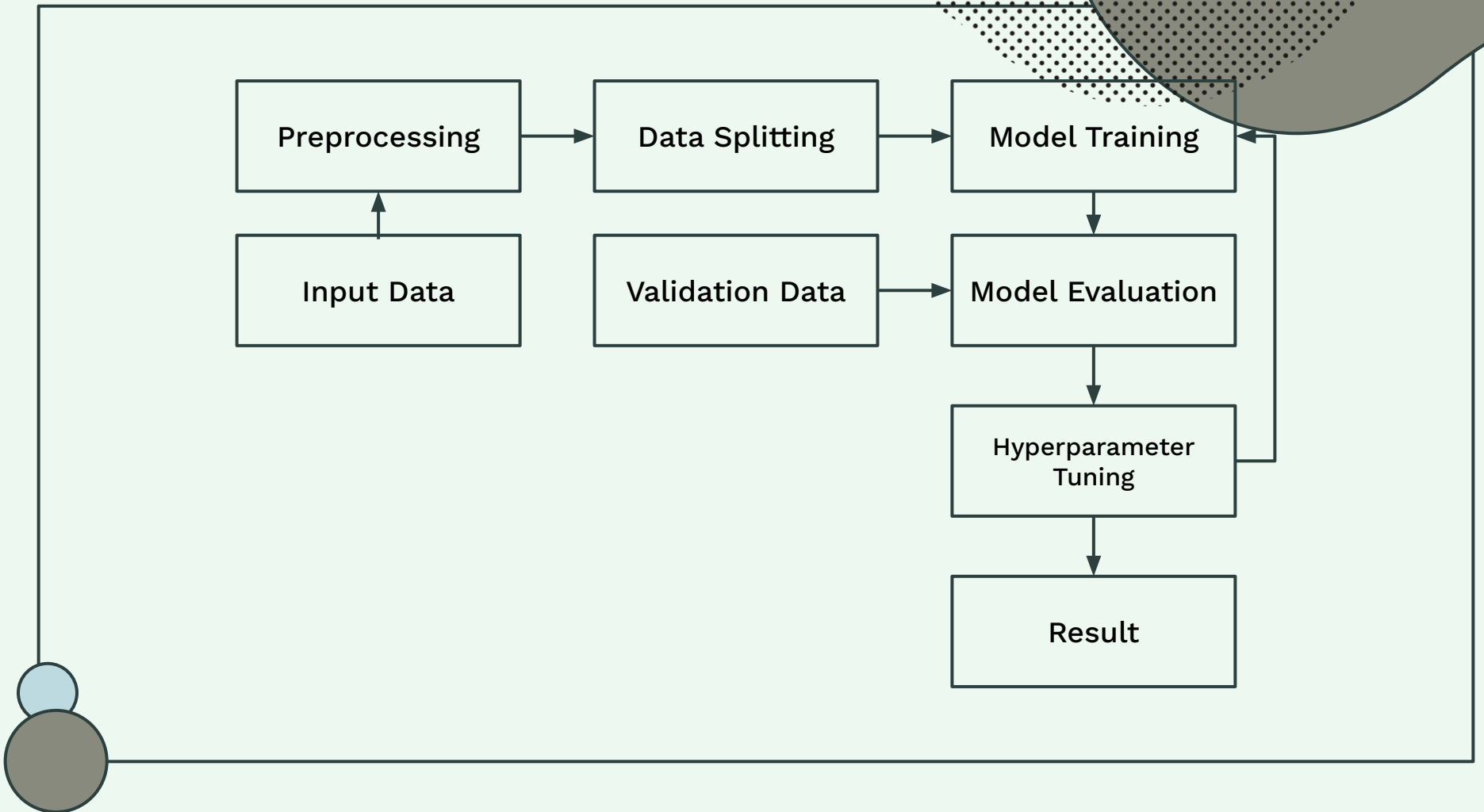
Mengukur keseimbangan antara Precision – Recall

Confusion Matrix



04.

IMPLEMENTASI


Preprocessing
Data Splitting
Model Training
Input Data
Validation Data
Model Evaluation
Hyperparameter Tuning
Result

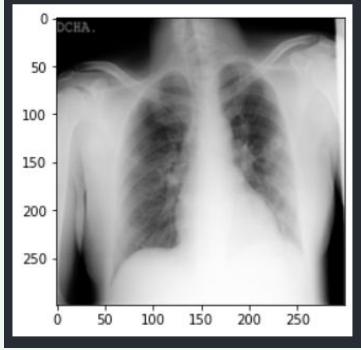
Input Data

```
print('Jumlah X-Ray Normal: ', len(os.listdir(normPath)))
print('Jumlah X-Ray COVID: ', len(os.listdir(covPath)))
```

```
Jumlah X-Ray Normal:  10192
Jumlah X-Ray COVID:  3616
```

```
#Random image info
img = tf.keras.utils.load_img(img_path)
img = np.asanyarray(img)
print('Shape: ', img.shape)

plt.imshow(img)
plt.show()
```

```
Shape:  (299, 299, 3)
```



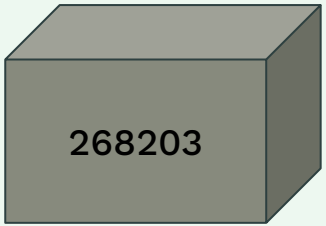

```
# Preparing COVID X-Ray Images

for image in tqdm(os.listdir(covPath)):
    img=Image.open(os.path.join(covPath,image))
    img=ImageOps.grayscale(img)
    img=img.resize((64,64))
    img=np.asarray(img)
    img=img.reshape((64,64,1))
    tot_images.append(img)
```

```
100%|██████████| 3616/3616 [00:51<00:00, 70.20it/s]
```

```
# Preparing Normal X-Ray Images

for image in tqdm(os.listdir(normPath)):
    img=Image.open(os.path.join(normPath,image))
    img=ImageOps.grayscale(img)
    img=img.resize((64,64))
    img=np.asarray(img)
    img=img.reshape((64,64,1))
    tot_images.append(img)
```

```
100%|██████████| 10192/10192 [02:22<00:00, 71.53it/s]
```

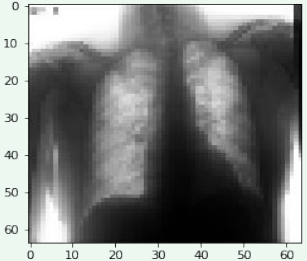
Preprocessing



Vektorisasi dan Normalisasi

Data Splitting

299 x 299 x 3 → 64 x 64 x 1 →



```
array([[[  0],
        [  0],
        [  0],
        ...,
        [ 11],
        [ 15],
        [ 44]],

       [[  0],
        [  0],
        [  0],
        ...,
        [  0],
        [  0],
        [  0]],

       [[  0],
        [  0],
        [  0],
        ...,
        [  0],
        [  0],
        [  0]],

       ...,
...
```

```
array([[[0.        ],
        [0.        ],
        [0.        ],
        ...,
        [0.        ],
        [0.73333335],
        [0.76862746]],

       [[0.22352941],
        [0.34117648],
        [0.21568628],
        ...,
        [0.        ],
        [0.7529412 ],
        [0.78431374]],

       [[0.21960784],
        [0.4117647 ],
        [0.09411765],
        ...,
        [0.        ],
        [0.75686276],
        [0.78431374]],

       ...,
...
```

```
Test Data
(2762, 64, 64, 1) (2762,)
Train Data
(11046, 64, 64, 1) (11046,)

0 = Negative, 1 = Positive
Covid Distribution (Test)
(array([0, 1]), array([2039,  723], dtype=int64))
Covid Distribution (Train)
(array([0, 1]), array([8153, 2893], dtype=int64))
```

Preprocessing

Resampling Train Data

```
Total Data
13860 13860
Data Distribution 10:7
(array([0, 1]), array([8153, 5707], dtype=int64))
```

SMOTE

```
Data Distribution 1:1
(array([0, 1]), array([5707, 5707], dtype=int64))
Total Train Data
(11414, 64, 64, 1) (11414,)
```

Random Undersampling

Model

```python
# CNN model architecture

cnn_model = models.Sequential()
cnn_model.add(layers.Conv2D(filters = 32, kernel_size = (3, 3), activation = 'relu', input_shape = (64, 64, 1)))
cnn_model.add(layers.MaxPooling2D((2, 2)))
cnn_model.add(layers.Dropout(0.3))

cnn_model.add(layers.Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
cnn_model.add(layers.MaxPooling2D((2, 2)))
cnn_model.add(layers.Dropout(0.5))

cnn_model.add(layers.Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
cnn_model.add(layers.Flatten())
cnn_model.add(layers.Dense(units = 64, activation = 'relu'))
cnn_model.add(layers.Dropout(0.2))

cnn_model.add(layers.Dense(units = 1, activation = 'sigmoid'))

cnn_model.compile(optimizer = 'adam',
          loss = tf.keras.losses.BinaryCrossentropy(from_logits = False),
          metrics = ['accuracy'])

cnn_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 dropout (Dropout)           (None, 31, 31, 32)        0

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 14, 14, 64)        0

 conv2d_2 (Conv2D)           (None, 12, 12, 64)        36928

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 64)                589888

 dropout_2 (Dropout)         (None, 64)                0
...
Total params: 645,697
Trainable params: 645,697
Non-trainable params: 0
```

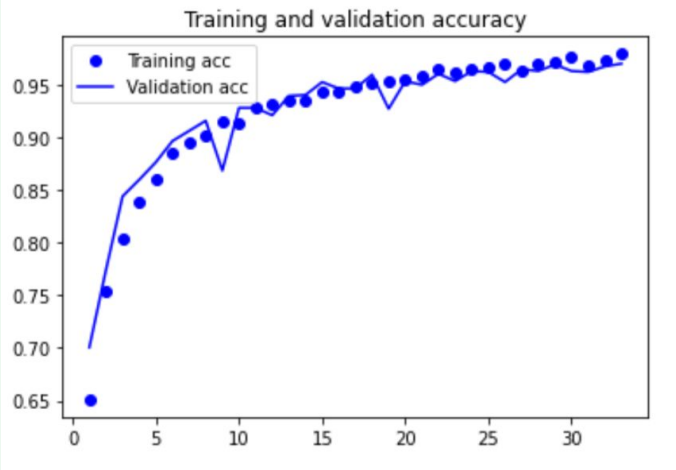
Fitting

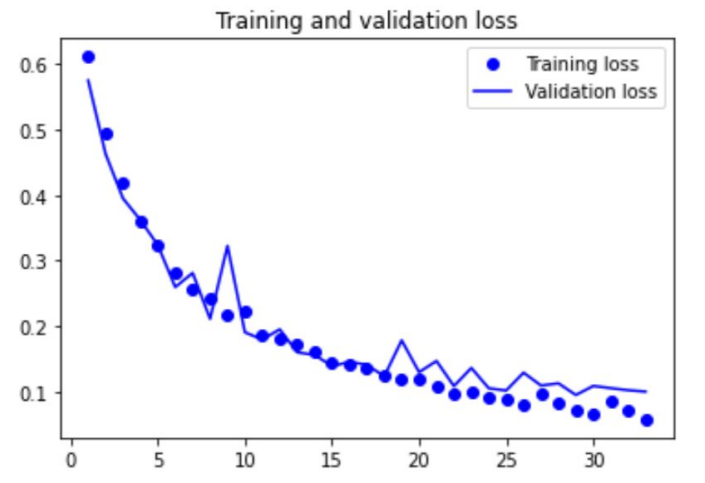
```
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size = 0.1, random_state=1)
```

```
# Model fitting

es = tf.keras.callbacks.EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 4)

history = cnn_model.fit(x_train, y_train,
                        epochs = 50, batch_size = 256,
                        validation_data = (x_val, y_val),
                        callbacks = [es])
```

```
Epoch 30/50
41/41 [==============================] - 19s 467ms/step - loss: 0.0647 - accuracy: 0.9769 - val_loss: 0.1086 - val_accuracy: 0.9632
Epoch 31/50
41/41 [==============================] - 18s 451ms/step - loss: 0.0847 - accuracy: 0.9689 - val_loss: 0.1052 - val_accuracy: 0.9623
Epoch 32/50
41/41 [==============================] - 19s 454ms/step - loss: 0.0725 - accuracy: 0.9742 - val_loss: 0.1021 - val_accuracy: 0.9676
Epoch 33/50
41/41 [==============================] - 18s 447ms/step - loss: 0.0576 - accuracy: 0.9797 - val_loss: 0.0999 - val_accuracy: 0.9702
Epoch 00033: early stopping
```

Grafik Evaluasi





Performa Model

----------Convolution Neural Network----------

Classification Report for Train Data

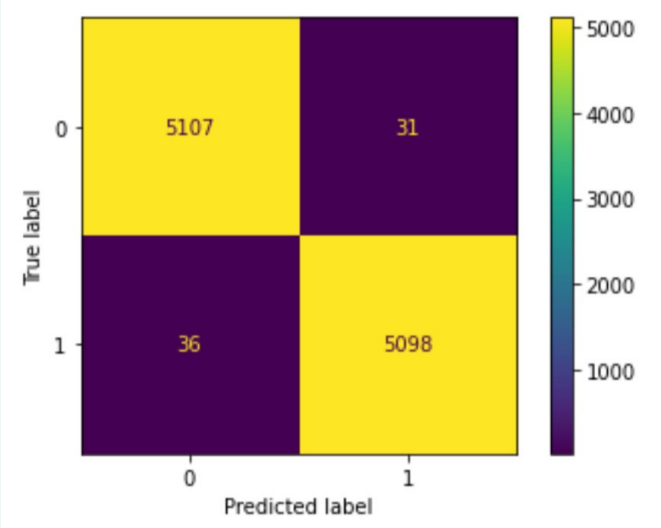
	precision	recall	f1-score	support
0	0.99	0.99	0.99	5138
1	0.99	0.99	0.99	5134
accuracy			0.99	10272
macro avg	0.99	0.99	0.99	10272
weighted avg	0.99	0.99	0.99	10272

Classification Report for Test Data

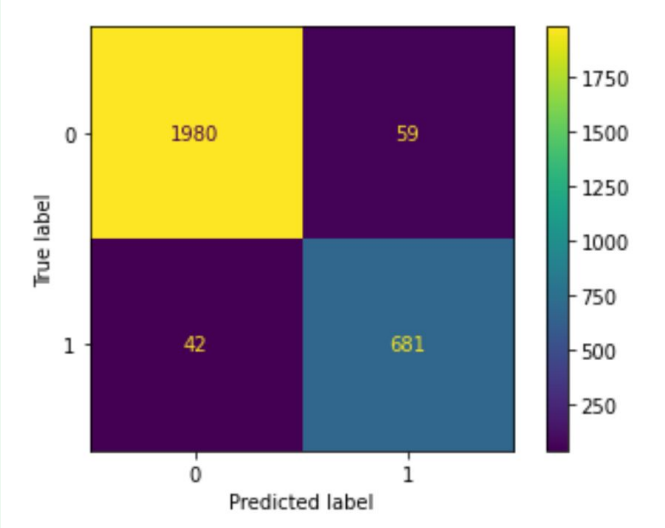
	precision	recall	f1-score	support
0	0.98	0.97	0.98	2039
1	0.92	0.94	0.93	723
accuracy			0.96	2762
macro avg	0.95	0.96	0.95	2762
weighted avg	0.96	0.96	0.96	2762

Performa Model

Train Data



Test Data




```
test_acc: 0.9634323120117188
test_loss: 0.10932224988937378
```

Kesimpulan

Berdasarkan metode dan analisis dari dataset COVID-19 Radiography Database dengan menggunakan arsitektur CNN dengan 11 layer didapat model yang dapat memprediksi COVID-19 menggunakan CXR dengan tingkat akurasi mencapai 96%. Dapat disimpulkan bahwa deteksi COVID-19 dapat dilakukan menggunakan Deep Learning dengan hasil yang sangat akurat.

THANKS

https://www.kaggle.com/code/rafialvanzah/covid-19-detection-using-cnn