BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi

Berdasarkan identifikasi masalah yang telah dibahas pada bab I ditemukan rumusan bagaimana mengimplementasikan klasifikasi citra buah apel menggunakan *Convolutional Neural Network* serta cara meningkatkan tingkat akurasinya. Dengan rumusan masalah diatas maka peneliti membuat program dengan menggunakan Google Colab dan bahasa pemrograman yang bahasa *python*.

Berikut tahap implementasi pada Google Colab

4.1.1 Inisialisasi Google Colab

Pada tahap ini dilakukan dengan menginisialisasi direktori penyimpanan dataset penelitian pada *google drive*. Sebelum melakukan inisialisasi pada *google colab* kita perlu melakukan sinkronisasi ke *google drive* dimana dataset penelitian disimpan. Sinkronisasi dilakukan dengan cara mengisi kode otorisasi pada prompt yang muncul pada google colab dengan menekan URL lalu masuk ke akun *google drive* dan menyalin kode otorisasi. Berikut tahap sinkronisasi google colab ke google drive :

1. Sebelum masuk ke google colab, dilakukan pengaturan pada notebook colab yaitu dengan mengubah akselerator *Hardware* ke GPU. untuk mengubahnya pilih menu edit, lalu notebook setting, lalu atur *hardware accelerator* ke GPU

Gambar 4.1 Pengaturan Notebook

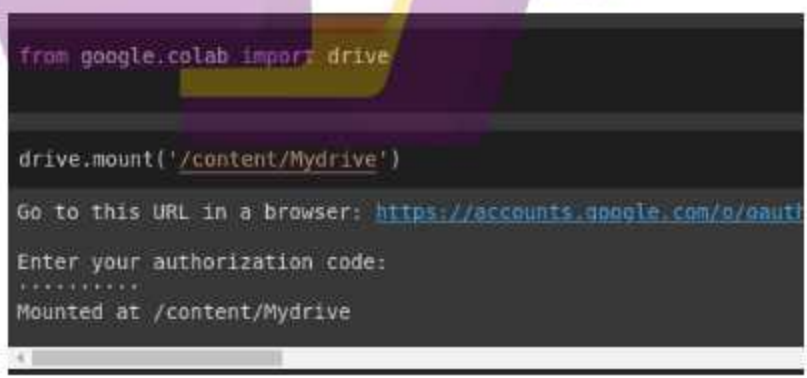
2. Selanjutnya untuk melakukan sinkronisasi *colab* ke *drive* dilakukan dengan menulis kode seperti dibawah ini, sehingga akan muncul URL dan input text untuk mengisi kode otorisasi.

Gambar 4.2 Proses Otorisasi

3. Klik URL tersebut untuk sinkronisasi akun google drive yang akan digunakan dan akan mendapatkan kode otorisasi lalu salin dan tempel pada input teks sebelumnya

Gambar 4.3 Kode Otorisasi

4. Setelah menyalin dan menempel kode otorisasi pada input teks lalu tekan enter dan akan muncul "Mounted at /content/Mydrive" maka proses sinkronisasi ke google drive telah selesai



Gambar 4. 4 Setelah Melakukan Otorisasi

4.1.2 Inisialisasi Direktori Google Drive

Setelah menghubungkan ke google drive, langkah selanjutnya adalah menginisialisasi direktori penyimpanan dataset yang akan digunakan dalam penelitian ini. Berikut *source code* untuk menginisialisasi direktori dan menampilkan banyaknya data training pada masing - masing jenis buah apple.

```
[5] import zipfile, os

[6] # Menginisialisasi direktori penyimpanan dataset
    zip_location = "/content/drive/My Drive/Script sweet/dataset/fullapple.zip"

[7] # Membaca file zip menggunakan method ZipFile
    zip_read = zipfile.ZipFile(zip_location,'r')

[9] # Mengekstrak data yang ada pada file zip
    zip_read.extractall('/content/drive/My Drive/Script sweet/dataset')

[10] # mengakhiri proses ekstraksi file zip
     zip_read.close()
```

Gambar 4. 5 Inisialisasi Direktori Dataset

Setelah inisialisasi direktori utama kemudian akan dilakukan pembagian data training dan validasi menggunakan library *splitfolderx* berikut source code pembagian data dan inisialisasi masing - masing direktori jenis buah apel, pembagian data training dan validasi adalah sejumlah 80% dan 20%

```
[26] print("jumlah data training braeburn", len(os.listdir(braeburn_dir_train)))
     print("jumlah data training PinkLady", len(os.listdir(pinklady_dir_train)))

     jumlah data training braeburn 352
     jumlah data training PinkLady 364
```

Gambar 4.6 Total data Training

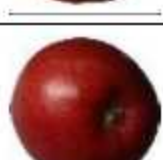
```
[16]
    import splitfolders

    # melakukan pembagian data training dan validasi
    splitfolders.ratio('/content/drive/My Drive/Scriptsweet/dataset/fullapple',
                       output="/content/drive/My Drive/Scriptsweet/dataset/split_data",
                       seed=1337, ratio=(.8, .2))

[19] # initialisasi direktori data yang akan digunakan
    base_dir='/content/drive/My Drive/Scriptsweet/dataset/split_data'

[22] # data training terletak pada path train didalam direktori utama
    train_dir = os.path.join(base_dir, 'train')
    # data validasi terletak pada path val didalam direktori utama
    val_dir = os.path.join(base_dir, 'val')

[23] # path data training apel Braeburn
    braeburn_dir_train = os.path.join(train_dir, 'Braeburn')
    # path data training apel PinkLady
    pinklady_dir_train = os.path.join(train_dir, 'Pinklady')

[25] # path data validasi apel Braeburn
    braeburn_dir_validation = os.path.join(val_dir, 'Braeburn')
    # path data validasi apel PinkLady
    pinklady_dir_validation = os.path.join(val_dir, 'Pinklady')
```

Gambar 4 7 Pembagian direktori data training dan validasi

4.2 Data Testing

Pada penelitian ini dilakukan pengujian dengan menggunakan data citra digital sebanyak 80 data, 40 data Apel *Braeburn* dan 40 data Apel *Pink Lady* yang sudah dilakukan *resize* menjadi ukuran 64 x 64 piksel. Berikut contoh gambar dataset training Apel *Braeburn* dan *Pink Lady*.

Tabel 4.1 Apel *Braeburn*

No	Gambar
	
2.	
3.	
4.	
5.	
6.	
7.	

8.	
9.	

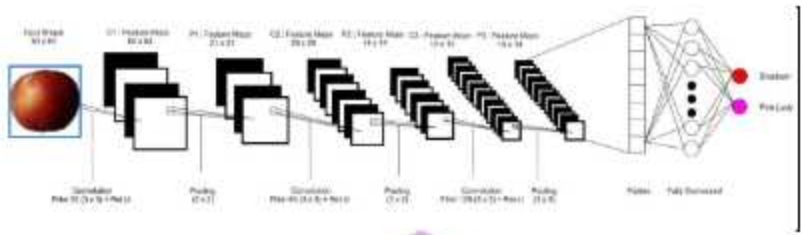
Tabel 4.2 Apel *Pink Lady*

No	Gambar
1.	
2.	
3.	
4.	

5.	
6.	
7.	
8.	
9.	

4.3 Preprocessing dan Pembuatan Model CNN

Setelah melakukan sinkronisasi dan inisialisasi direktori penyimpanan dataset pada *google colab*, langkah selanjutnya adalah tahap preprocessing dan pembuatan model CNN. pada tahap preprocessing ukuran semua gambar pada input akan diubah menjadi 64 x 64 pixel. Kemudian dataset akan dilatih menggunakan algoritma CNN dengan model yang sudah dibuat sebagai berikut



Gambar 4.8 *Arsitektur Jaringan*

Gambar 4.8 merupakan arsitektur jaringan pada proses training untuk mendapatkan model yang optimal. Pada penelitian ini menggunakan input gambar dengan ukuran 64 x 64

1. Proses konvolusi pertama menggunakan kernel size 3 x 3 dan jumlah filter sebanyak 32 filter. Proses konvolusi adalah proses kombinasi dua buah matriks yang berbeda untuk menghasilkan suatu nilai matriks yang baru. Fungsi aktivasi *ReLU (Rectified Linear Unit)* ditambahkan setelah proses *konvolusi* yang bertujuan untuk mengubah nilai negatif menjadi nol atau bisa dikatakan menghilangkan nilai negatif dalam sebuah matriks hasil konvolusi. Hasil konvolusi pertama memiliki matriks baru berukuran 62 x 62
2. Proses *Pooling* pertama menggunakan kernel ukuran 2x2. Proses *pooling* merupakan proses pengurangan ukuran pada matriks dengan menggunakan operasi *pooling*. Pooling layer terdiri dari filter dengan ukuran tertentu yang secara bergantian bergeser pada bagian *feature map*. Pada penelitian ini menggunakan *Max Pooling* pada proses *pooling*. Pada proses *Max Pooling* nilai maksimum dari matriks konvolusi pertama diambil berdasarkan pergeseran kernel

dengan stride yaitu 2. Berdasarkan hasil dari operasi *pooling*, dihasilkan matriks baru berukuran 31 x 31 dari hasil konvolusi pertama berukuran 62 x 62

3. Proses konvolusi kedua yaitu adalah meneruskan hasil dari proses *pooling* pertama yakni dengan input matriks gambar sebesar 31 x 31 dengan jumlah filter sebanyak 64 dan dengan ukuran kernel 3 x 3. Proses konvolusi kedua ini sama - sama menggunakan fungsi aktivasi *ReLU* dan menghasilkan gambar berukuran 29 x 29
4. Setelah konvolusi kedua, proses berikutnya adalah proses *pooling* yang kedua dimana prosesnya hampir sama dengan proses *pooling* pertama. Hasil output dari *pooling* kedua memiliki ukuran gambar 14 x 14
5. Proses selanjutnya adalah konvolusi ketiga, dimana input matriks gambar berukuran 14 x 14 dengan jumlah filter sebanyak 128 menggunakan kernel berukuran 3 x 3. Proses konvolusi ketiga ini juga menggunakan fungsi aktivasi *ReLU*. Hasil dari konvolusi ketiga adalah gambar berukuran 12 x 12
6. Setelah konvolusi ketiga dilanjutkan dengan proses *pooling* ketiga. Proses ini hampir sama dengan pooling sebelumnya dan menghasilkan output gambar berukuran 6 x 6
7. Selanjutnya adalah proses *Flatten* atau *Fully Connected*. Flatten disini adalah mengubah *output pooling* layer menjadi sebuah vektor.

8. Proses terakhir adalah menggunakan aktivasi fungsi *Softmax*. Secara spesifik fungsi ini biasa digunakan pada metode klasifikasi *multinomial logistic regression* dan *multiclass linear discriminant analysis*. Fungsi *sigmoid* akan mengubah nilai linear menjadi non - linear dan bernilai nol sampai satu.

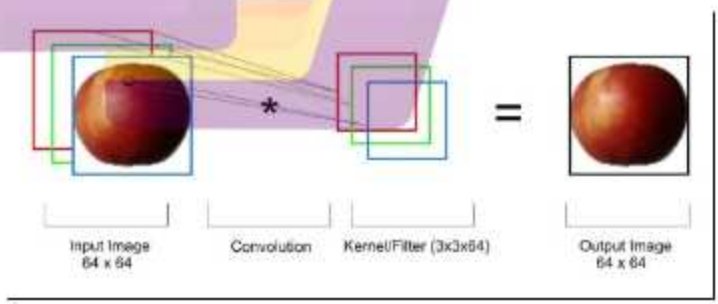
Berdasarkan dari uraian arsitektur jaringan yang digunakan untuk proses pelatihan. Didapatkan total parameter pada model arsitektur tersebut sebanyak 2.453.569

4.4 Pelatihan

Pada sub bab akan menjelaskan proses pelatihan pada data training dengan model CNN yang telah dibuat.

A. Proses Convolution Layer

Berdasarkan penguraian dari arsitektur jaringan pada sub bab sebelumnya berikut ini adalah pembahasan mengenai proses konvolusi.



Gambar 4.9 Proses Konvolusi

```
model1 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu',
                           input_shape=(64, 64, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Gambar 4. 10 Source Code Konvolusi

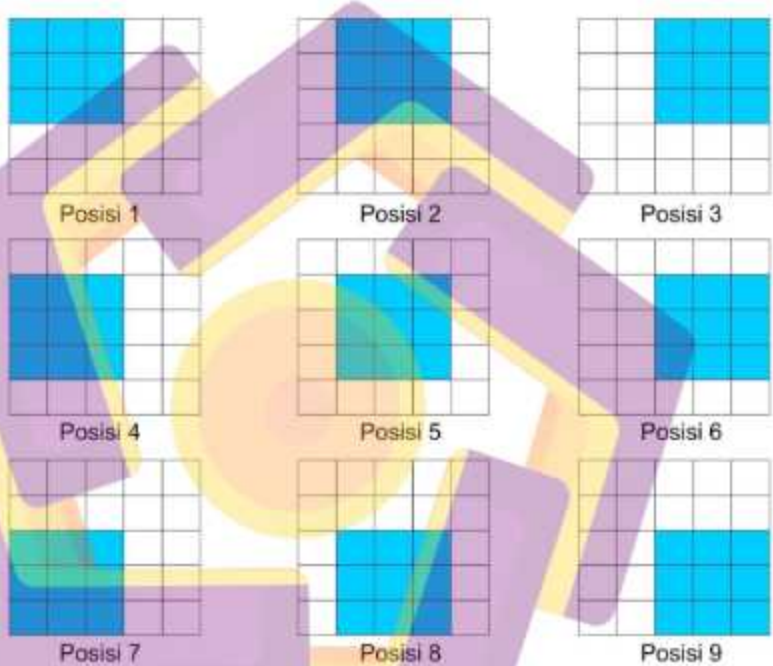
Konvolusi merupakan mengkombinasi dua buah deret angka yang menghasilkan deret angka yang ketiga. Jika diimplementasikan angka pada konvolusi ini adalah berbentuk matrik *array*. Pada input, gambar memiliki ukuran piksel 64 x 64 x 3, ini menunjukan bahwa tinggi dan lebar piksel dari gambar sebesar 64 dan gambar tersebut memiliki 3 channel warna yaitu *reed*, *green*, dan *blue* atau yang sering disebut RGB. pada setiap channel piksel memiliki nilai matriks yang berbeda - beda. Input akan di konvo dengan nilai filter yang sudah ditentukan. Filter merupakan blok lain atau kubus dengan tinggi dan lebar yang lebih kecil namun kedalamannya sama dengan yang tersapu di atas gambar dasar atau gambar asli. Filter digunakan untuk menentukan pola apa yang akan dideteksi yang selanjutnya di konvolusi atau dikalikan dengan nilai pada matriks input, nilai pada masing - masing kolom dan baris pada matriks

sangat bergantung pada jenis pola yang akan dideteksi. Jumlah filter pada konvo ini sebanyak 64 piksel dengan ukuran kernel 3 x 3, ini berarti gambar yang dihasilkan dari konvolusi akan sebanyak 64 fitur map.

Agar dapat lebih memahami cara kerja dari proses konvolusi, peneliti akan menggunakan sampel matriks pada *input image*. Karena pada penelitian ini menggunakan input ukuran 64 x 64 piksel, maka peneliti hanya mengambil sebagian nilai matriks saja yang akan dijadikan sampel pada proses konvolusi.

Gambar 4.11 Perhitungan Proses Konvolusi

Gambar 4.11 menunjukan proses konvolusi dengan menggunakan ukuran kernel 3 x 3, dengan menggunakan *stride* 1. Stride disini artinya jumlah pergeseran kernel terhadap matriks input berjumlah satu. Jika divisualisasikan sebagai berikut:



Gambar 4.12 Posisi Kernel pada Konvolusi

Gambar 4.12 menunjukan perhitungan dot *product* pad proses konvolusi dimana sebuah kernel ukuran 3 x 3 yang dimulai pada sisi bagian kiri. Proses ini disebut dengan *sliding windows*, namun pada penelitian ini diberikan nilai *padding* 1, yaitu adanya penambahan nilai 0 di sekeliling nilai matriks input supaya input dan output memiliki nilai matriks yang sama, sehingga tidak mengurangi

informasi pada gambar. Proses ini dilakukan dari ujung kiri atas hingga sampai ujung kiri bawah. Perhitungan *dot product* dapat dilihat sebagai berikut:

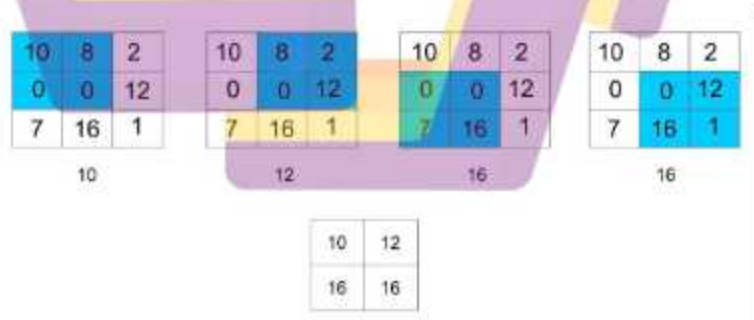
1. Posisi 1 = (3x1) + (5x(-1) + (1x1) + (1x(-1)) + (5x1) + (3x(-1)) + (3x1) + (7x(-1)) + (8x1) = -10
2. Posisi 2 = (1x1) + (5x(-1)) + (3x1) + (3x (-1) + (7x1) + (8x(-1)) + (1x1) + (1x(-1)) + (1x1) = -8
3. Posisi 3 = (3x1) + (7x(-1)) + (8x1) + (1x(-1) + (1x1) + (1x(-1)) + (3x1) + (2x(-1)) + (6x1) = -2
4. Posisi 4 = (5x1) + (1x(-1)) + (2x1) + (5x(-1)) + (3x1) + (2x(-1)) + (7x1) + (8x(-1)) + (1x1) = 0
5. Posisi 5 = (5x1) + (3x(-1)) + (2x1) + (7x(-1)) + (8x1) + (1x(-1)) + (1x1) + (1x(-1)) + (4x1) = 0
6. Posisi 6 = (7x1) + (8x(-1)) + (1x1) + (1x(-1) + (1x1) + (4x(-1)) + (2x1) + (6x-1) + (4x1) = -12
7. Posisi 7 = (1x1) + (2x(-1)) + (5x1) + (3x(-1)) + (2x1) + (3x(-1)) + (8x1) + (1x(-1)) + (2x1) = 7
8. Posisi 8 = (3x1) + (2x(-1)) + (3x1) + (8x(-1)) + (1x1) + (2x(-1)) + (1x1) + (4x(-1)) + (6x 1) = -16
9. Posisi 9 = (8x1) + (1x(-1) + (2x1) + (1x(-1)) + (4x1) + (6x(-1)) + (6x1) + (4x(-1)) + (7x1) = 1

Kemudian sebelum dilanjutkan ke proses pooling layer, untuk menghilangkan nilai negatif pada hasil, pada arsitektur

jaringan digunakan aktivasi ReLU (*Rectified Linear Unit*) setelah proses konvolusi. Fungsi dari aktivasi tersebut adalah melakukan "*treshold*" dari 0 hingga *infinity*. Nilai yang ada pada hasil konvolusi yang bernilai negatif akan diubah dengan aktivasi tersebut menjadi nol dan nilai yang lainnya sampai *infinity*.

B. Proses *Pooling*

Pooling merupakan pengurangan ukuran matriks dengan menggunakan operasi *pooling* (penggabungan). Metode yang digunakan dalam proses *pooling* ini menggunakan *max pooling*. Dalam penelitian [12] menunjukan bahwa penggunaan metode *max pooling* lebih unggul dibanding dengan metode *sub sampling*. Penggunaan metode ini menjadi salah satu metode terbaik dalam proses *pooling*. Berikut ini gambar dari proses *pooling* :

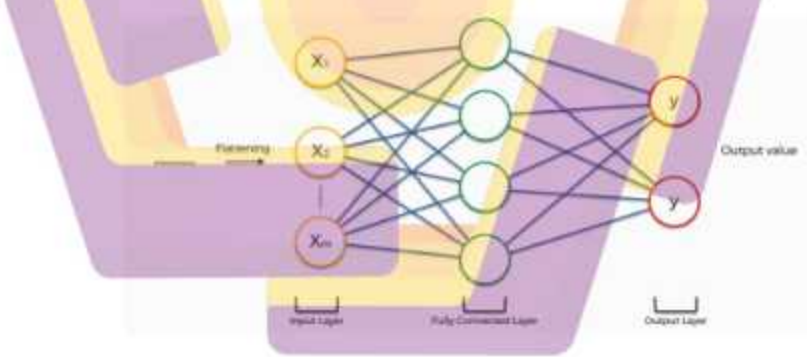


Gambar 4.13 Proses *Pooling*

Proses *pooling* ini menggunakan ukuran 2 x 2 dengan stride 1 dimana jumlah pergeseran kernel terhadap matriks input berjumlah satu. Dalam proses *pooling* ini digunakan metode *max-pooling*, dimana *window* akan bergeser sesuai dengan ukurannya dan juga *stride*nya agar mendapatkan nilai yang paling maksimum. Dapat dilihat pada gambar x.x *output* dari proses ini memiliki nilai paling maksimum yang diambil dari matriks fitur map hasil konvolusi, kemudian hasil dari *max-pooling* tersebut berukuran 2 x 2.

C. Proses *Fully Connected*

Proses selanjutnya adalah *Fully Connected layer*. Ini bertujuan untuk melakukan transformasi pada dimensi data agar dapat diklasifikasikan secara linear.



Gambar 4.14 Proses Fully Connected Layer

Gambar 4.14 merupakan proses *converting* hasil dari fitur map *max-pooling* menjadi *flatten* atau vektor. Dalam proses ini nilai input matriks dari layer sebelumnya akan diubah menjadi vektor. Proses ini sama dengan proses MLP (*Multilayer Perceptron*). Jaringan ini

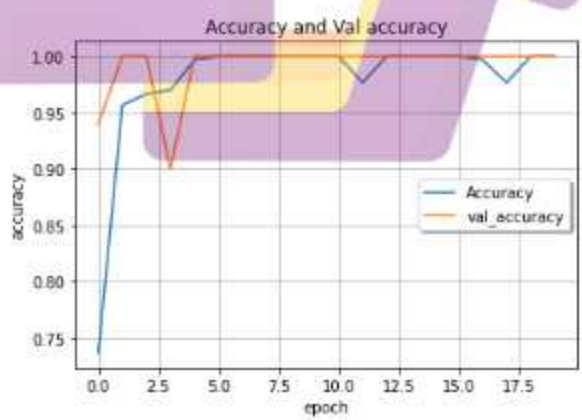
pada umumnya menggunakan lapisan yang terhubung sepenuhnya di mana setiap piksel dianggap sebagai neuron terpisah. Dalam proses ini biasanya diterapkan metode "*dropout*". Metode ini bertujuan untuk menonaktifkan beberapa edge yang terhubung ke setiap neuron untuk menghindari *overfitting*. Setelah itu proses terakhir adalah klasifikasi. Dalam proses ini digunakan aktivasi fungsi *softmax*. Aktivasi ini akan membantu MLP untuk mengklasifikasi input terhadap targetnya, yaitu kedalam 2 kelas buah apel (*Braeburn* dan *Pink Lady*).

D. Hasil *Training*

Setelah melewati beberapa proses dalam algoritma *Convolutional Neural Network* (CNN) didapatkan hasil training dan validation. Proses ini menggunakan jumlah 20 *epoch*, nilai *learning rate* 0.001, berikut grafik hasil *training* dan *validation* menggunakan *matplotlib* :

```
Epoch 1/20
30/30 - 143s - loss: 0.6910 - accuracy: 0.7467 - val_loss: 0.2319 - val_accuracy: 0.9800
Epoch 2/20
30/30 - 9s - loss: 0.2206 - accuracy: 0.9467 - val_loss: 0.0444 - val_accuracy: 1.0000
Epoch 3/20
30/30 - 6s - loss: 0.0665 - accuracy: 0.9780 - val_loss: 0.0024 - val_accuracy: 1.0000
Epoch 4/20
30/30 - 4s - loss: 0.0336 - accuracy: 0.9867 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 5/20
30/30 - 3s - loss: 4.7137e-04 - accuracy: 1.0000 - val_loss: 8.9417e-05 - val_accuracy: 1.0000
Epoch 6/20
30/30 - 4s - loss: 6.7065e-05 - accuracy: 1.0000 - val_loss: 1.4160e-05 - val_accuracy: 1.0000
Epoch 7/20
30/30 - 3s - loss: 9.6884e-06 - accuracy: 1.0000 - val_loss: 2.3936e-05 - val_accuracy: 1.0000
Epoch 8/20
30/30 - 3s - loss: 0.2016 - accuracy: 0.9833 - val_loss: 0.0054 - val_accuracy: 1.0000
Epoch 9/20
30/30 - 3s - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 10/20
30/30 - 3s - loss: 1.1476e-04 - accuracy: 1.0000 - val_loss: 1.8762e-05 - val_accuracy: 1.0000
Epoch 11/20
30/30 - 3s - loss: 8.2780e-06 - accuracy: 1.0000 - val_loss: 8.4653e-06 - val_accuracy: 1.0000
Epoch 12/20
30/30 - 3s - loss: 1.0763e-06 - accuracy: 1.0000 - val_loss: 3.7836e-07 - val_accuracy: 1.0000
Epoch 13/20
30/30 - 3s - loss: 5.2416e-07 - accuracy: 1.0000 - val_loss: 5.0775e-07 - val_accuracy: 1.0000
Epoch 14/20
30/30 - 3s - loss: 1.6086e-07 - accuracy: 1.0000 - val_loss: 2.5757e-08 - val_accuracy: 1.0000
Epoch 15/20
30/30 - 3s - loss: 2.5988e-08 - accuracy: 1.0000 - val_loss: 1.9404e-08 - val_accuracy: 1.0000
Epoch 16/20
30/30 - 3s - loss: 7.3575e-09 - accuracy: 1.0000 - val_loss: 1.0191e-08 - val_accuracy: 1.0000
Epoch 17/20
30/30 - 3s - loss: 0.2988 - accuracy: 0.9867 - val_loss: 3.0588e-05 - val_accuracy: 1.0000
Epoch 18/20
30/30 - 3s - loss: 3.6357e-05 - accuracy: 1.0000 - val_loss: 4.0051e-05 - val_accuracy: 1.0000
Epoch 19/20
30/30 - 3s - loss: 1.4705e-05 - accuracy: 1.0000 - val_loss: 1.4903e-05 - val_accuracy: 1.0000
Epoch 20/20
30/30 - 3s - loss: 5.0408e-06 - accuracy: 1.0000 - val_loss: 8.2111e-06 - val_accuracy: 1.0000
```

Gambal 4.14 Hasil Training



Gambar 4.15 Grafik *accuracy* dan *val accuracy*

Gambar 4.16 Grafik *loss* dan val *loss*

Berdasarkan gambar 4.15 *accuracy* dari *training* model mencapai 100% dengan nilai *loss* 0.000013. Proses training disini menggunakan learning rate 0.001 dengan input gambar sebesar 64 x 64 piksel. Waktu pelatihan yang dibutuhkan untuk 20 *epoch* dalam menjalankan *training* model yaitu 65 detik. Semakin banyak epoch maka semakin lama juga waktu yang dibutuhkan untuk *training* model. Kemudian *accuracy* dari data validation mencapai 100 % dengan nilai loss sebesar 0.0000056.

E. Hasil Testing

Pada proses testing menggunakan data uji sebanyak 80 data. Untuk setiap kelas jenis buah apel sebanyak 40 gambar. Hasil *confusion matriks* adalah sebagai berikut :

Tabel 4.3 *Confusion* Matriks

Matriks		Predict Class	
		Braeburn	Pink Lady
Actual Class	Braeburn	40	0
	Pink Lady	2	38

Berdasarkan tabel diatas hasil prediksi dari model terhadap data *testing* data baru menunjukan hasil yang baik. Prediksi terhadap jenis buah apel *Braeburn* diklasifikasikan ke dalam *Braeburn*, ini artinya klasifikasi terhadap gambar tersebut adalah benar. Prediksi pada jenis buah apel yang kedua *pink lady* diklasifikasikan benar sebagai *pink lady* sebanyak 38 dan *missing data* dari input pink lady diklasifikasikan sebagai *Braeburn* sebanyak 2 data. Perhitungan menggunakan metode *overall Accuracy* keseluruhan matriks diatas adalah sebagai berikut:

$$Overall\ Accuracy = \frac{TTP_{all}}{Total\ Number\ of\ Testing\ Entries}$$

Overall accuracy = 78/80 = 97.5 %

Jadi akurasi yang dihasilkan oleh model dengan input gambar 64x64 piksel dengan nilai *learning rate* sebesar 0.001 dan jumlah sampel testing 80 data didapatkan nilai akurasi sebesar 97.5 %

4.5 Penentuan Parameter Model

Penentuan model terbaik, harus dicari nilai terbaik parameter dalam model CNN. parameter yang dimaksud adalah pengaruh jumlah *epoch*,

pengaruh ukuran input gambar, pengaruh jumlah data training, pengaruh skenario data, ukuran kernel dan *learning rate*. Tujuan dari penentuan parameter model ini adalah ingin membandingkan model mana yang paling terbaik dengan memperhatikan nilai parameternya.

4.5.1 Pengaruh jumlah *epoch*

Epoch adalah ketika seluruh *dataset* sudah melalui proses *training* pada *Neural Network* sampai dikembalikan ke awal dalam satu putaran. Dalam *Neural Network* satu epoch itu terlalu besar dalam proses pelatihan karena terlalu besar dalam proses pelatihan karena seluruh data dimasukan kedalam proses *training* sehingga akan membutuhkan waktu cukup lama. Untuk mempermudah dan mempercepat proses *training* biasanya, dataset dibagi per *Batch* (*Batch Size*) . Berikut adalah hasil perbandingan epoch dari hasil *training*.

Tabel 4.4Akurasi berdasarkan *Epoch*

Epoch	*Val Accuracy*	*Loss Validation*	*Time (Seconds)*
20	100%	0.0000056	65
30	100%	2.456 x 10-10	90
30	100%	2.0905 x 10-11	150

Berdasarkan tabel diatas dengan menggunakan nilai *learning rate* 0.001 didapatkan akurasi yang sama pada setiap nilai *epoch* yang

berbeda. Jika dilihat dari tabel semakin tinggi nilai epoch yang digunakan maka nilai *loss* akan semakin kecil.

4.5.2 Pengaruh nilai *Learning Rate*

Penelitian ini juga melakukan uji coba dengan menggunakan nilai *Learning Rate* yang berbeda dan epoch 20. Dalam klasifikasi gambar pada umumnya banyak menggunakan nilai *learning rate sebesar* 0.1 sampai 0.0001 [3]. Penentuan nilai *learning rate* biasanya ditentukan oleh peneliti. Peneliti menggunakan tiga nilai yaitu 0.1, 0.01, dan 0.001. Penentuan nilai dari *learning rate* ini sangat berpengaruh pada performa akurasi. Hasil *learning rate* adalah sebagai berikut

Tabel 4.5 Akurasi berdasarkan *Learning Rate*

Learning Rate	*Val Accuracy*	*Loss Validation*	*Time (Seconds)*
0.1	46%	0.7039	62
0.01	54%	0.6931	68
0.001	100%	0.0000056	65

Berdasarkan Tabel 4.5 penggunaan learning rate 0.1 menghasilkan tingkat akurasi yang tidak optimal yaitu sebesar 46% dan nilai *loss* sebesar 0.7039, kemudian ketika menggunakan learning rate sebesar 0.01 hasilnya tidak terlalu jauh dari sebelumnya yaitu akurasinya sebesar 54% dan nilai *loss* sebesar 0.6931, hal ini karena Ketika menggunakan *learning rate* dengan nilai cukup besar, maka nilai *loss* akan semakin meningkat ketika menjalankan beberapa iterasi pada saat

training, kemudian pada nilai *learning rate* 0.001 yang digunakan pada penelitian ini menghasilkan nilai akurasi 100% dan nilai *loss* sebesar 0.0000056.