BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi Sistem

Pada subbab ini, akan dilakukan implementasi sistem pada sistem yang telah dibangun dengan menggunakan analisis dan perancangan seperti bab sebelumnya. Beberapa subbab yang akan dijelaskan pada subbab implementasi sistem yaitu implementasi perangkat keras, implementasi perangkat lunak, implementasi antarmuka dan implementasi kelas.

4.1.1. Implementasi Perangkat Keras

Implementasi perangkat keras merupakan penjelasan perangkat keras yang digunakan untuk melakukan implementasi pada sistem yang telah dibangun. Berikut ini pada **Tabel 4.1** menjelaskan perangkat keras yang digunakan dalam melakukan implementasi sistem.

Tabel 4.1. Implementasi perangkat keras

Perangkat Keras	Spesifikasi
Processor	Intel core i5 @ 2.2 GHz
Memory	Nvidia Geforce GT930M
Harddisk	8 GB
VGA	500 GB

4.1.2. Implementasi Perangkat Lunak

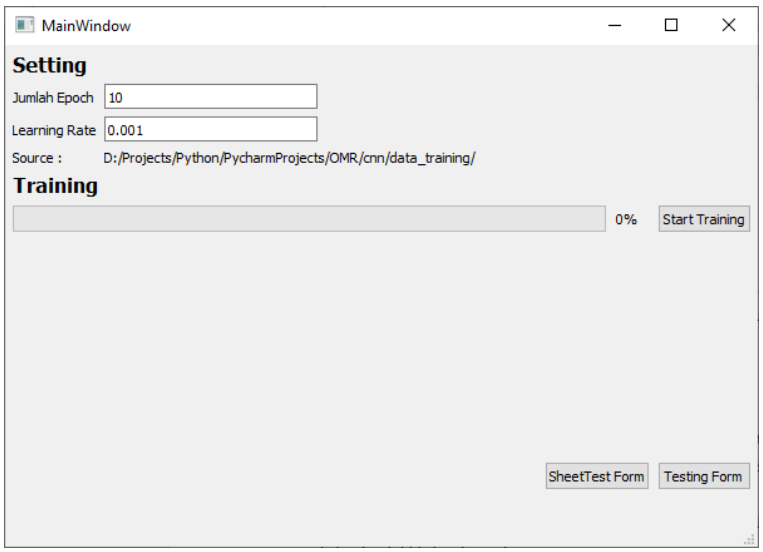
Implementasi perangkat lunak merupakan penjelasan perangkat lunak yang digunakan untuk melakukan implementasi pada sistem yang telah dibangun. Berikut ini pada **Tabel 4.2** menjelaskan perangkat lunak yang digunakan dalam melakukan implementasi sistem.

Tabel 4.2. Implementasi perangkat lunak

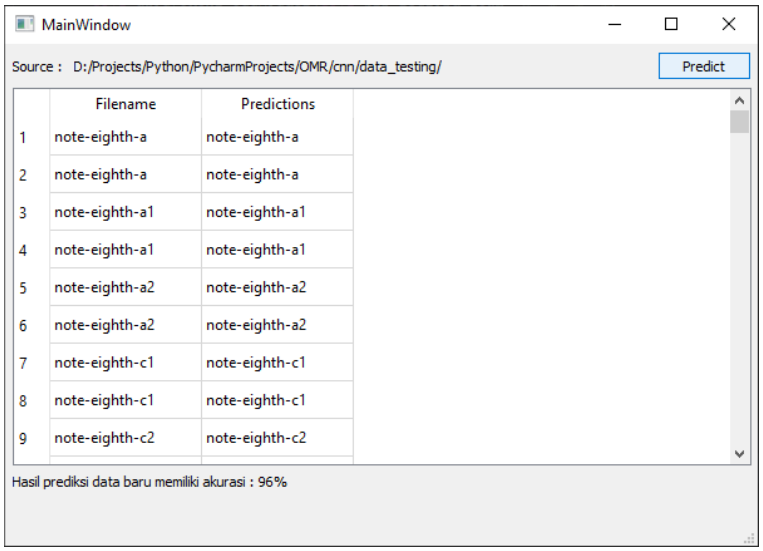
Perangkat Lunak	Spesifikasi Software
Sistem Operasi	Windows 10
IDE	PyCharm
Python version	3.7

4.1.3. Implementasi Antarmuka

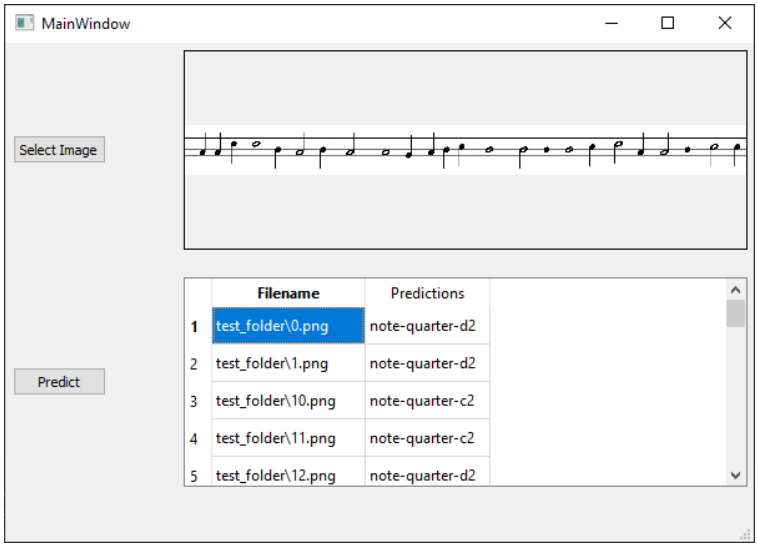
Implementasi antarmuka merupakan hasil dari perancangan yang telah dibangun pada bab sebelumnya. Berikut ini hasil implementasi antarmuka dapat dilihat pada **Gambar 4.1** sampai **Gambar 4.3**.



Gambar 4.1. Implementasi antarmuka *training*



Gambar 4.2. Implementasi antarmuka *testing*



Gambar 4.3. Implementasi antarmuka *sheettest*

4.1.4. Implementasi Kelas

Implementasi kelas merupakan hasil dari perancangan yang telah dibangun pada bab sebelumnya. Karena sistem yang dibangun oleh peneliti berbasis *object oriented programming* (OOP), maka pada subbab ini akan menampilkan implementasi kelas yang digunakan pada sistem. Berikut ini hasil implementasi kelas dapat dilihat pada **Gambar 4.4** sampai **Gambar 4.8**.

```
class Train(object):
    def __init__(self): ...

    def setupUi(self, MainWindow): ...

    def retranslateUi(self, MainWindow): ...

    def startTrain(self): ...

    def testingForm(self): ...

    def sheetTestForm(self): ...
```

Gambar 4.4. Implementasi kelas *Train*

Pada **Gambar 4.4** menjelaskan bahwa kelas *train* memiliki beberapa *method* yang digunakan yaitu __init__(), setupUi(), retranslateUi(), startTrain(), testingForm() dan sheetTestForm(). *Method* __init__() merupakan *method* inisialisasi pada kelas tersebut dimana di dalamnya terdapat *script* untuk memanggil kelas *LoadData*. *Method* setupUi() merupakan *method* untuk

mendesain *graphical user interface* (GUI). *Method* retranslateUi() merupakan *method* untuk melakukan *setting* teks pada *graphical user interface* (GUI). *Method* startTrain() merupakan *method* untuk melakukan proses *training* yang di dalamnya terdapat *script* memanggil kelas CNN yang berperan sebagai model arsitektur lalu akan dilakukan proses *training* dan menampilkan evaluasi pada proses *traning* tersebut serta menyimpan hasil *training* tersebut agar dapat digunakan kembali untuk melakukan prediksi data. *Method* testingForm() merupakan *method* yang berfungsi untuk memanggil kelas *Test* dan menampilkan GUI dari kelas *Test*. *Method* sheetTestForm() merupakan *method* yang berfungsi untuk memanggil kelas *SheetTest* dan menampilkan GUI dari kelas *SheetTest*.

```
class Test(object):
    def __init__(self): ...

    def setupUi(self, MainWindow): ...

    def retranslateUi(self, MainWindow): ...

    def addRow(self, row, itemLabels=[]): ...

    def startPredict(self): ...
```

Gambar 4.5. Implementasi kelas *Test*

Pada **Gambar 4.5** menjelaskan bahwa kelas *Test* memiliki beberapa *method* yang digunakan yaitu __init__(), setupUi(), retranslateUi(), addrow() dan startPredict(). *Method* __init__() merupakan *method* inisialisasi pada kelas tersebut dimana di dalamnya terdapat *script* untuk memanggil kelas *LoadData*. *Method* setupUi() merupakan *method* untuk mendesain *graphical user interface* (GUI). *Method* retranslateUi() merupakan *method* untuk melakukan *setting* teks pada *graphical user interface* (GUI). *Method* addrow() merupakan suatu *method* untuk melakukan penambahan baris baru pada tampilan keluaran tabel. *Method* startPredict() merupakan *method* yang akan melakukan proses prediksi dimana di dalamnya terdapat *script* memuat hasil proses *training* dan melakukan prediksi. Hasil prediksi akan ditampilkan pada tabel serta *file external* berekstensi csv.

```
class LoadData:
    img_rows, img_cols = 50, 30
    seed = 23
    np.random.seed(seed)
    folder_data_train = "D:/Projects/Python/PycharmProjects/OMR/cnn/data_training/"
    folder_data_test = "D:/Projects/Python/PycharmProjects/OMR/cnn/data_testing/"
    folder_data_test_sheet = "D:/Projects/Python/PycharmProjects/OMR/cnn/data_testing_sheet/"

    def loadDataTrain(self): ...

    def loadDataTest(self, folder): ...
```

Gambar 4.6. Implementasi kelas *LoadData*

Pada **Gambar 4.6** menjelaskan bahwa kelas *LoadData* memiliki beberapa *attribute* dan *method* yang digunakan. *Attribute* yang digunakan yaitu img_row, img_cols dan seed merupakan variabel dengan tipe data integer dan folder_data_train dan folder_data_test merupakan variabel dengan tipe data string. *Method* yang digunakan yaitu loadDataTrain() merupakan *method* yang di dalamnya akan melakukan proses memuat dataset *training* untuk keperluan proses *training* maupun proses *testing*. *Method* loadDataTest() merupakan *method* yang di dalamnya akan melakukan proses memuat dataset *testing* baru yang akan digunakan untuk keperluan testing prediksi data baru.

```
class SheetTest(object):
    def setupUi(self, MainWindow): ...

    def retranslateUi(self, MainWindow): ...

    def loadImage(self): ...

    def prePredict(self, fileName): ...

    def addRow(self, row, itemLabels=[]): ...

    def startPredict(self): ...
```

Gambar 4.7. Implementasi kelas *SheetTest*

Pada **Gambar 4.7** menjelaskan bahwa dalam kelas *SheetTest* memiliki beberapa *method* yang digunakan yaitu setupUi(), retranslateUi(), loadImage(), prePredict(), addRow() dan startPredict(). *Method* setupUi() merupakan *method* untuk mendesain *graphical user interface* (GUI). *Method* retranslateUi() merupakan *method* untuk melakukan *setting* teks pada *graphical user interface*

(GUI). *Method* loadImage() merupakan *method* yang di dalamnya terdapat *script* untuk membuka dialog *openfile* agar pengguna dapat melakukan memuat *file* gambar *external*. *Method* prePredict() merupakan *method* yang di dalamnya akan melakukan *preprocessing* terhadap gambar yang dimuat pengguna. *Method* addrow() merupakan suatu *method* untuk melakukan penambahan baris baru pada tampilan keluaran tabel. *Method* startPredict() merupakan *method* yang akan melakukan proses prediksi dimana di dalamnya terdapat *script* memuat hasil proses *training* dan melakukan prediksi. Hasil prediksi akan ditampilkan pada tabel serta *file external* berekstensi csv.

```
class CNN:

    def __init__(self, input_shape, nb_classes, lr):
        self.input_shape = input_shape
        self.nb_classes = nb_classes
        self.lr = lr

    def set_input_shape(self, input_shape):
        self.input_shape = input_shape

    def set_nb_classes(self, nb_classes):
        self.nb_classes = nb_classes

    def set_lr(self, lr):
        self.lr = lr

    def get_input_shape(self):
        print(self.input_shape)

    def get_nb_classes(self):
        print(self.nb_classes)

    def get_lr(self):
        print(self.lr)

    def my_model(self): ...
```

Gambar 4.8. Implementasi kelas CNN

Pada **Gambar 4.8** menjelaskan bahwa dalam kelas CNN memiliki beberapa *method* yang digunakan yaitu __init__(), set_input_shape(), set_nb_classes(), set_lr(), get_input_shape(), get_nb_classes(), get_lr() dan my_model(). *Method* __init__() merupakan *method* inisialisasi pada kelas tersebut dimana di dalamnya terdapat *script* parameter untuk kelas tersebut dengan kata lain berfungsi sebagai

konstruktor untuk kelas CNN. *Method* set_input_shape(), set_nb_classes() dan set_lr() merupakan *method* untuk melakukan perubahan parameter yang telah dimasukkan di awal saat pemanggilan objek. *Method get*_input_shape(), get_nb_classes() dan get_lr() merupakan *method* untuk melakukan pengambilan parameter yang telah dimasukkan di awal atau di set menggunakan *method setter*. *Method* my_model() merupakan *method* yang di dalamnya terdapat *script* model arsitektur CNN yang digunakan.

4.2. Pengujian

Pada subbab ini, akan dilakukan pengujian pada sistem yang telah dibangun dengan menggunakan analisis dan perancangan seperti pada bab sebelumnya. Beberapa subbab yang akan dibahas yaitu pengujian sistem dan pengujian akurasi. Pada subbab pengujian sistem memiliki beberapa tahapan yang akan dibahas yaitu skenario pengujian sistem, pengujian kernel, pengujian jumlah layer pada *convolutional neural network*, pengujian jumlah neuron pada *hidden layer* dan pengujian *learning rate* dan *epoch*. Selanjutnya pada subbab pengujian akurasi akan membahas beberapa tahapan yaitu skenario pengujian akurasi, hasil pengujian akurasi dan pembahasan hasil pengujian akurasi.

4.2.1. Pengujian Sistem

Pada subbab pengujian sistem, peneliti melakukan pengujian pada fungsionalitas sistem yang telah dibangun apakah sistem telah berjalan sesuai yang diharapkan.

4.2.1.1. Skenario Pengujian Sistem

Skenario pengujian sistem adalah tahapan pengujian terhadap perangkat lunak yang telah dibangun. Berikut ini skenario pengujian sistem dapat dilihat pada **Tabel 4.3**.

Tabel 4.3. Skenario pengujian sistem

Kelas Uji	Butir Uji	Jenis Pengujian
Train	Verifikasi epoch	Black Box
	Verifikasi learning rate	Black Box
	Tombol Start Training	Black Box
	Tombol Testing Form	Black Box
	Tombol SheetTest Form	Black Box
Test	Tombol Predict	Balck Box
SheetTest	Tombol Select Image	Black Box
	Tombol Predict	Black Box

4.2.1.2. Hasil Pengujian Sistem

Hasil pengujian sistem didapatkan dengan menguji setiap proses serta kemungkinan kesalahan yang terjadi untuk setiap proses. Pengujian ini dilakukan menggunakan metode *black box* dimana hanya memperhatikan masukkan dan keluaran sistem. Berikut ini hasi dari pengujian sistem dapat dilihat pada **Tabel 4.4** sampai **Tabel 4.6**.

Tabel 4.4. Pengujian Train

Kasus dan Hasil Uji (Data Normal)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Verifikasi *Epoch*	Masukan hanya number	Masukan hanya number	Diterima
Verifikasi *Learning rate*	Masukan berupa double	Masukan hanya double	Diterima
Tombol *Start Training*	Dapat melakukan proses training dan menyimpan hasil proses training	Tombol *start training* dapat berjalan sesuai yang diharapkan	Diterima
Tombol *Testing Form*	Dapat membuka form *Test*	Tombol *testing form* dapat berjalan sesuai yang diharapkan	Diterima
Tombol *SheetTest Form*	Dapat membuka form *SheetTest*	Tombol *sheettest form* dapat berjalan sesuai yang diharapkan	Diterima
Kasus dan Hasil Uji (Data Salah)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Verifikasi *Epoch*	Selain number tidak dapat masuk text box	Selain number tidak dapat dimasukkan	Diterima

Verifikasi *Learning rate*	Selain float tidak dapat masuk text box	Selain number dan titik tidak dapat dimasukkan	Diterima
Tombol *Start Training*	Tidak dapat melakukan proses training dan menyimpan hasil proses training apabila *epoch* atau *learning rate* kosong	Sistem menampilkan informasi *epoch* atau *learning rate* kosong	Diterima
Tombol *Testing Form*	Form *Testing* tidak tampil apabila sistem terjadi error	Form *Testing* tidak tampil	Diterima
Tombol *SheetTest Form*	Form *SheetTest* tidak tampil apabila sistem terjadi error	Form *SheetTest* tidak tampil	Diterima

Tabel 4.5. Pengujian Test

Kasus dan Hasil Uji (Data Normal)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Tombol *Predict*	Dapat melakukan proses prediksi dan menampilkan hasil prediksi pada tabel	Hasil prediksi muncul pada tabel	Diterima
Kasus dan Hasil Uji (Data Salah)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Tombol *Predict*	Tidak dapat melakukan proses prediksi dan tidak menampilkan hasil prediksi pada tabel apabila sistem terjadi error	Hasil prediksi tidak muncul pada tabel	Diterima

Tabel 4.6. Pengujian SheetTest

Kasus dan Hasil Uji (Data Normal)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Tombol *Select Image*	Dapat memilih *image*	Muncul dialog dan memilih *image*	Diterima

Tombol *Predict*	Dapat melakukan prediksi dan menampilkan ke dalam tabel	Hasil prediksi muncul dalam tabel	Diterima
Kasus dan Hasil Uji (Data Salah)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Tombol *Select Image*	Tidak dapat memilih *image* selain berekstensi png	Image muncul hanya berekstensi png	Diterima
Tombol *Predict*	Tidak dapat melakukan prediksi dan tidak menampilkan hasil prediksi ke dalam tabel apabila sistem terjadi error	Hasil prediksi tidak muncul pada tabel	Diterima

4.2.1.3. Pembahasan Pengujian Sistem

Berdasarkan subbab hasil pengujian sistem dapat ditarik kesimpulan bahwa perangkat lunak yang telah dibangun bebas darikesalahan sintaks secara fungsional dan menampilkan hasil yang sesuai yang diharapan.

4.2.2. Pengujian Akurasi

Pada subbab pengujian akurasi akan membahas pengujian prediksi data baru, dimana pada penelitian ini telah disediakan 2 gambar data baru untuk setiap kelasnya. Sebelum dapat melakukan pengujian prediksi dara baru diperlukan pengujian arsitektur yang akan digunakan. Beberapa pengujian yang akan dilakukan peneliti yaitu pengujian ukuran kernel, pengujian jumlah layer pada CNN, pengujian jumlah neuron pada *hidden layer*, pengujian *learning rate* dan *epoch* dan pengujian prediksi data baru.

4.2.2.1. Skenario Pengujian Akurasi

Skenario pengujian akurasi adalah tahapan jalan cerita yang akan dilakukan peneliti untuk melakukan pengujian akurasi. Berikut ini beberapa skenario yang akan dilakukan yaitu:

1. Pengujian dilakukan dengan data yang berbeda dengan perbandingan 80% data *training* dan 20% data *testing*.
2. Untuk proses *training* dengan jumlah data sebanyak 4106.
3. Pengujian ukuran kernel dilakukan dengan melakukan perbandingan ukuran kernel yaitu 3x3 dan 5x5. Hasil terbaik pada pengujian ukuran kernel akan digunakan untuk melakukan pengujian jumlah layer pada *convolutional neural network*.
4. Pengujian jumlah layer pada *convolutional neural network* dilakukan dengan melakukan perbandingan arsitektur 3 layer *convolution layer* dan *pooling layer* dan arsitektur 4 layer *convolution layer* dan *pooling layer*. Hasil terbaik pada pengujian jumlah layer CNN akan digunakan untuk melakukan pengujian jumlah neuron pada *hidden layer*.
5. Pengujian jumlah neuron pada *hidden layer* dilakukan dengan melakukan perbandingan jumlah neuron yaitu 5120, 7168 dan 9216. Hasil terbaik pada pengujian jumlah neuron pada *hidden layer* akan digunakan untuk pengujian *learning rate* dan *epoch*.
6. Pengujian *learning rate* dan *epoch* dilakukan dengan perbandingan *learning rate* dari 0.0005 sampai 0.005 dan *epoch* sebanyak 30.
7. Data *testing* baru yang akan dilakukan prediksi sebanyak 180 gambar dan dilakukan setelah semua pengujian arsitektur dilakukan.

4.2.2.2. Pengujian Ukuran Kernel

Pengujian ukuran kernel dilakukan untuk mendapatkan ukuran kernel yang sesuai dengan kasus penelitian ini. Peneliti mencoba melakukan pengujian dengan beberapa ukuran kernel yaitu 3x3 dan 5x5 dengan parameter pengujian *epoch* sebesar 10 dan *learning rate* sebesar 0.001. Berikut hasil perbandingan ukuran kernel dapat dilihat pada **Tabel 4.7**.

Tabel 4.7. Hasil pengujian ukuran kernel

Ukuran Kernel	Validation Accuracy	Validation Loss
3x3	0.86	0.54
5x5	0.84	0.62

Pada **Tabel 4.7** menunjukkan bahwa menggunakan kernel 3x3 lebih unggul dari pada menggunakan kernel 5x5 dapat dilihat pada akurasi yang didapatkan dimana kernel 3x3 mendapatkan akurasi sebesar 86% dan *loss* paling kecil sebesar 0.54. Ukuran 5x5 kurang baik pada kasus penelitian karena ukuran gambar masukan pada penelitian ini kecil, oleh karena itu banyak informasi yang hilang saat dilakukan *convolution layer* dan *pooling layer*.

4.2.2.3. Pengujian Jumlah Layer pada *Convolutional Neural Network*

Pengujian jumlah layer pada CNN dilakukan untuk mendapatkan jumlah layer yang sesuai dengan kasus penelitian ini. Peneliti mencoba melakukan pengujian dengan menentukan jumlah layer *convolution layer* dan *pooling layer* dengan jumlah yang berbeda. Berikut ini gambar model arsitektur yang akan dibandingkan.

Tabel 4.8. Arsitektur dengan 3 layer pada *convolution layer* dan *pooling layer*

No.	Layer	Output	Parameter	
1	Conv2D_1	50x30x64	(3x3x1+1)x64	640
2	MaxPool_1	25x15x64		0
3	Conv2D_2	25x15x128	(3x3x64+1)x128	73856
4	MaxPool_2	13x8x128		0
5	Conv2D_3	13x8x256	(3x3x128+1)x256	295168
6	MaxPool_3	7x4x256		0
7	Flatten	7168		0
8	Dense	7168	(7168+1)x7168	51387392
9	Output	90	(7168+1)x90	645210
Total Parameter				52402266

Tabel 4.9. Arsitektur dengan 4 layer pada *convolution layer* dan *pooling layer*

No.	Layer	Output	Parameter	
1	Conv2D_1	50x30x64	((3x3x1+1)x64)	640
2	MaxPool_1	25x15x64		0
3	Conv2D_2	25x15x128	((3x3x64+1)x128)	73856
4	MaxPool_2	13x8x128		0
5	Conv2D_3	13x8x256	((3x3x128+1)x256)	295168
6	MaxPool_3	7x4x256		0
7	Conv2D_4	7x4x512	((3x3x256+1)x512)	1180160
8	MaxPool_4	4x2x512		0
9	Flatten	4096		0
10	Dense	4096	(4096+1)x4096	16781312
11	Output	90	(4096+1)x90	368730
Total Parameter				18699866

Arsitektur yang akan dibandingkan dapat dilihat pada **Tabel 4.8** yang merupakan arsitektur dengan jumlah 3 layer pada *convolution layer* dan 3 *pooling layer* dengan filter yang digunakan 64, 128 dan 256 dan **Tabel 4.9** yang merupakan arsitektur dengan jumlah 4 layer pada *convolution layer* dan 4 *pooling layer* dengan filter yang digunakan 64, 128, 256 dan 512. Parameter yang digunakan untuk melakukan pengujian yaitu *epoch* sebesar 10 dan *learning rate* sebesar 0.001. Berikut hasil perbandingan dapat dilihat pada **Tabel 4.10**.

Tabel 4.10. Hasil pengujian jumlah layer pada *convolution layer* dan *pooling layer*

Banyaknya Layer Convolution dan Pooling	Validation Accuracy	Validation Loss
3	0.91	0.56
4	0.85	0.56

Pada **Tabel 4.10** membuktikan bahwa arsitektur dengan jumlah 3 layer pada layer *convolution* dan *pooling* lebih unggul dari pada arsitektur 4 layer pada *convlolution* dan *pooling* untuk kasus penelitian ini dapat dilihat pada akurasi yang didapatkan dimana arsitektur dengan jumlah 3 layer mendapatkan akurasi sebesar 91% lebih besar dari pada arsitektur dengan jumlah 4 layer yang mendapatkan akurasi sebesar 85%. Pada kasus pengujian jumlah layer ini menunjukkan bahwa semakin detail pada proses *feature learning* (*convolution* dan *pooling*) tidak menjamin akan memiliki akurasi yang tinggi.

4.2.2.4. Pengujian Jumlah Neuron pada *Hidden Layer*

Pengujian jumlah neuron pada *hidden layer* dilakukan untuk mendapatkan jumlah neuron yang sesuai pada penelitian ini. Peneliti mencoba melakukan pengujian dengan beberapa jumlah neuron pada *hidden layer* yaitu 5120, 7168 dan 9216 dengan parameter pengujian *epoch* sebesar 10 dan *learning rate* sebesar 0.001. Berikut hasil perbandingan jumlah neuron pada hidden layer dapat dilihat pada **Tabel 4.11**.

Tabel 4.11. Hasil pengujian jumlah neuron pada *hidden layer*

Jumlah Neuron	Validation Accuracy	Validation Loss
5120	0.87	0.66
7168	0.91	0.56
9216	0.89	0.57

Pada **Tabel 4.11** menunjukkan bahwa menggunakan jumlah neuron 7168 lebih baik dari pada menggunakan jumlah neuron 5120 dan 9216 dapat dilihat dari hasil akurasi yang didapatkan dimana untuk jumlah neuron 7168 mendapatkan hasil akurasi sebesar 91% lebih besar dari jumlah neuron yang lainnya dan *loss* sebesar 0.56 paling kecil dari jumlah neuron yang lainnya. Pada kasus pengujian jumlah neuron pada *hidden layer* ini menunjukkan bahwa semakin banyak jumlah neuron tidak menjamin akan mendapatkan hasil akurasi yang lebih baik dari jumlah neuron yang sedikit.

4.2.2.5. Pengujian *Learning Rate* dan *Epoch*

Pengujian *learning rate* dan *epoch* dilakukan untuk mendapatkan hasil *training* yang baik serta *learning rate* yang sesuai dengan kasus penelitian ini. Beberapa *learning rate* yang akan diujikan yaitu 0.0005 sampai 0.0015 dengan batasan *epoch* sebesar 30. Berikut ini hasil pengujian *learning rate* dan *epoch* dapat dilihat pada **Tabel 4.12**.

Tabel 4.12. Hasil pengujian *learning rate* dan *epoch*

No.	Learning Rate	Jumlah Epoch	Berhenti Epoch	Validation Loss	Validation Acc
1	0.0005	30	15	0.57	0.88
2	0.0006	30	15	0.56	0.89
3	0.0007	30	17	0.47	0.91
4	0.0008	30	15	0.51	0.9
5	0.0009	30	15	0.51	0.89
6	0.001	30	15	0.53	0.9
7	0.0011	30	14	0.6	0.89
8	0.0012	30	11	0.56	0.89
9	0.0013	30	11	0.55	0.89
10	0.0014	30	11	0.59	0.87
11	0.0015	30	16	0.65	0.87

Pada **Tabel 4.12** menunjukkan bahwa akurasi terbaik dari pengujian *learning rate* dan *epoch* di atas dipegang oleh *learning rate* 0.0007 dengan akurasi sebesar 91% dan *loss* paling kecil yaitu 0.47. Pada kasus ini semua pengujian ini mengalami konvergen karena pada saat pelatihan *epoch* berhenti saat sebelum batas *epoch* yang telah ditentukan.

4.2.2.6. Pengujian Prediksi Data Baru

Pengujian prediksi data baru dilakukan untuk mendapatkan seberapa baik akurasi arsitektur yang telah dilakukan pengujian pada subbab sebelumnya untuk melakukan prediksi terhadap data baru. Berikut ini hasil pengujian prediksi data baru yang dilakukan peneliti dapat dilihat pada **Tabel 4.13**.

Tabel 4.13. Hasil pengujian akurasi terhadap data baru

No.	Kelas	Jumlah Gambar	Jumlah Prediksi Benar
1	note-eighth-a	2	2
2	note-eighth-a1	2	2
3	note-eighth-a2	2	2
4	note-eighth-c1	2	2
5	note-eighth-c2	2	2
6	note-eighth-c3	2	2
7	note-eighth-d1	2	2
8	note-eighth-d2	2	2
9	note-eighth-e1	2	2
10	note-eighth-e2	2	2
11	note-eighth-f1	2	2
12	note-eighth-f2	2	2
13	note-eighth-g1	2	2

No.	Kelas	Jumlah Gambar	Jumlah Prediksi Benar
14	note-eighth-g2	2	2
15	note-eighth-h	2	2
16	note-eighth-h1	2	2
17	note-eighth-h2	2	1
18	note-half-a	2	2
19	note-half-a1	2	2
20	note-half-a2	2	1
21	note-half-c1	2	2
22	note-half-c2	2	2
23	note-half-c3	2	2
24	note-half-d1	2	2
25	note-half-d2	2	2
26	note-half-e1	2	2
27	note-half-e2	2	2
28	note-half-f1	2	2
29	note-half-f2	2	2
30	note-half-g1	2	2
31	note-half-g2	2	2
32	note-half-h	2	2
33	note-half-h1	2	2
34	note-half-h2	2	2
35	note-quarter-a	2	2
36	note-quarter-a1	2	2
37	note-quarter-a2	2	2
38	note-quarter-c1	2	1
39	note-quarter-c2	2	2
40	note-quarter-c3	2	2
41	note-quarter-d1	2	2
42	note-quarter-d2	2	2
43	note-quarter-e1	2	2
44	note-quarter-e2	2	2
45	note-quarter-f1	2	2
46	note-quarter-f2	2	2
47	note-quarter-g1	2	2
48	note-quarter-g2	2	2
49	note-quarter-h	2	2
50	note-quarter-h1	2	2
51	note-quarter-h2	2	1
52	note-sixteenth-a	2	2
53	note-sixteenth-a1	2	2
54	note-sixteenth-a2	2	2
55	note-sixteenth-c1	2	2
56	note-sixteenth-c2	2	2

No.	Kelas	Jumlah Gambar	Jumlah Prediksi Benar
57	note-sixteenth-c3	2	2
58	note-sixteenth-d1	2	1
59	note-sixteenth-d2	2	2
60	note-sixteenth-e1	2	2
61	note-sixteenth-e2	2	2
62	note-sixteenth-f1	2	2
63	note-sixteenth-f2	2	2
64	note-sixteenth-g1	2	2
65	note-sixteenth-g2	2	2
66	note-sixteenth-h	2	1
67	note-sixteenth-h1	2	2
68	note-sixteenth-h2	2	2
69	note-whole-a	2	2
70	note-whole-a1	2	2
71	note-whole-a2	2	2
72	note-whole-c1	2	2
73	note-whole-c2	2	2
74	note-whole-c3	2	2
75	note-whole-d1	2	1
76	note-whole-d2	2	2
77	note-whole-e1	2	2
78	note-whole-e2	2	2
79	note-whole-f1	2	2
80	note-whole-f2	2	2
81	note-whole-g1	2	2
82	note-whole-g2	2	2
83	note-whole-h	2	2
84	note-whole-h1	2	2
85	note-whole-h2	2	1
86	rest-eighth	2	2
87	rest-half	2	2
88	rest-quarter	2	2
89	rest-sixteenth	2	2
90	rest-whole	2	2
Total		180	172

4.2.2.7. Pembahasan Hasil Pengujian Akurasi

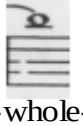
Setelah dilakukan pengujian menggunakan data baru pada **Tabel 4.13** maka hasil yang didapatkan jumlah data benar sebanyak 172 gambar dari 180 gambar. Untuk mendapatkan hasil akurasi dapat dihitung dengan menggunakan persamaan **(4.1)**.

$$\frac{Jumlah\ data\ benar}{Jumlah\ data\ test} * 100\% \quad \textbf{(4.1)}$$

$$\frac{172}{180} * 100\% = 95.56\%$$

Pada hasil pengujian akurasi **Tabel 4.13** ada 8 gambar yang tidak dapat dikenali. Berikut gambar yang tidak dikenali dapat dilihat pada **Tabel 4.14**.

Tabel 4.14. Data testing yang gagal diprediksi

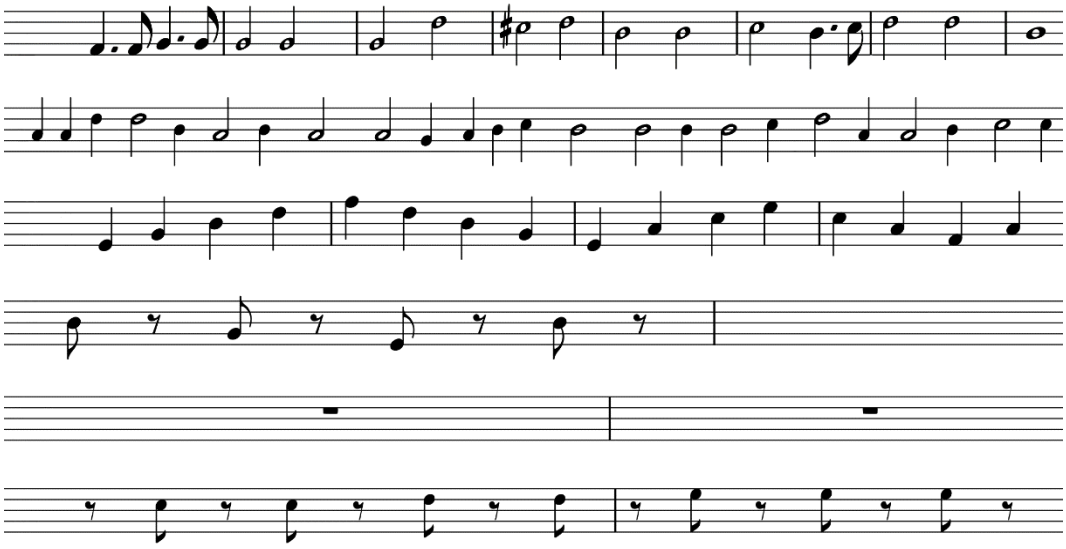
No.	Nama Gambar	Prediksi
1	 note-eighth-h2	note-sixteenth-h2
2	 note-half-a2	note-whole-a2
3	 note-quarter-c1	note-half-c1
4	 note-quarter-h2	note-quarter-a2
5	 note-sixteenth-d1	note-sixteenth-f1
6	 note-sixteenth-h	note-eighth-h
7	 note-whole-d1	note-whole-g1
8	 note-whole-h2	note-quarter-c3

Pada **Tabel 4.14**, 2 dari 8 gambar yang tidak dikenali yaitu note-quarter-c1 dan note-sixteenth-d1 memiliki jumlah *dataset* yang cukup banyak. Asumsi peneliti untuk gambar *testing* note-quarter-c1 dan note-sixteenth-d1 tidak dapat dikenali

karena objek pada gambar yang dikenali kurang jelas dan banyak objek lain pada gambar tersebut. Untuk 6 gambar lainnya gambar tidak dapat dikenali salah satu penyebabnya dikarenakan jumlah *dataset* pada kelas tersebut tidak banyak. Note-eighth-h2 tidak dapat dikenali karena kemungkinan gambar kurang jelas dan memiliki objek lain di sekitar not. Note-half-a2 tidak dapat dikenali karena kemungkinan pada kepala not sama seperti note-whole-a2 dan mengabaikan badan not. Note-quarter-h2 tidak dapat dikenali kemungkinan karena kemungkinan kepala not terdeteksi sejajar dengan garis bantu dan diprediksi note-quarter-a2. Note-sixteenth-h tidak dapat dikenali kemungkinan pemotongan gambar yang terlalu pendek dan terdeteksi hanya memiliki satu garis bendera. Note-whole-d1 tidak dapat dikenali kemungkinan memiliki objek lain pada gambar tersebut. Note-whole-h2 tidak dapat dikenali kemungkinan gambar kurang jelas.

4.2.3. Pengujian Akurasi Pada Baris Notasi Musik

Pada subbab pengujian akurasi pada baris notasi musik akan membahas pengujian prediksi pada baris notasi musik yang telah melalui proses segmentasi, dimana pada penelitian ini telah disediakan 6 gambar data baru berupa per baris notasi musik. Berikut ini baris notasi musik yang akan dimasukkan dapat dilihat pada **Gambar 4.9**.



Gambar 4.9. Citra masukkan baris notasi musik

4.2.3.1. Skenario Pengujian Akurasi Pada Baris Notasi Musik

Skenario pengujian akurasi pada baris notasi musik adalah tahapan jalan cerita yang akan dilakukan peneliti untuk melakukan pengujian akurasi untuk baris notasi musik. Berikut ini beberapa skenario yang akan dilakukan yaitu:

1. Arsitektur yang digunakan sama seperti pada subbab pengujian akurasi.
2. Baris notasi musik yang akan dimasukkan dengan ukuran 2250 *pixel* x 200 *pixel.*
3. Baris notasi musik akan dilakukan proses segmentasi pemotongan untuk setiap objek.
4. Pelabelan objek dilakukan secara manual.

4.2.3.2. Hasil Pengujian Prediksi Pada Baris Notasi Musik

Pengujian prediksi pada baris notasi musik dilakukan untuk mendapatkan seberapa baik akurasi arsitektur yang telah dilakukan pengujian pada subbab sebelumnya untuk melakukan prediksi terhadap baris notasi musik. Berikut ini hasil pengujian prediksi pada baris notasi musik yang dilakukan peneliti dapat dilihat pada **Lampiran C**.

4.2.3.3. Pembahasan Hasil Pengujian Akurasi Pada Baris Notasi Musik

Setelah dilakukan pengujian terhadap baris notasi musik pada **Lampiran C** maka hasil yang didapatkan jumlah data benar sebanyak 22 gambar dari 84 gambar. Untuk mendapatkan hasil akurasi dapat dihitung dengan menggunakan persamaan **(4.1)**.

$$\frac{Jumlah\ data\ benar}{Jumlah\ data\ test} * 100\% = \frac{22}{84} * 100\% = 26.19\%$$

Pada hasil pengujian akurasi terhadap notasi musik pada **Lampiran C** banyak prediksi yang kurang tepat karena menurut asumsi peneliti pada notasi gambar yang akan diprediksi garis paranada tidak komplit atau garis paranada pada posisi paling ujung atas gambar atau ujung bawah gambar sehingga tidak terbaca oleh sistem. Asumsi lainnya notasi objek yang akan diprediksi kurang jelas atau terdapat objek lain disekitar objek yang akan diprediksi.

4.2.4. Kesimpulan Pengujian

Dari pengujian yang telah dilakukan di atas maka dapat dilakukan penarikan kesimpulan, bahwa *optical music recognition* menggunakan *convolutional neural netwok* dengan melakukan data *testing* sebanyak 180 gambar mendapatkan akurasi sebesar 95.56% dan pengujian terhadap baris notasi musik dengan 84 objek pada citra notasi musik mendapatkan akurasi sebesar 26.19%.