BAB 3
METODE PENELITIAN

3.1 ALAT YANG DIGUNAKAN

Alat yang digunakan pada penilitian terdiri dari perangkat keras (*hardware*), dan perangkat lunak (*software*), dan Dataset. Adapun perincian dari alat yang digunakan adalah sebagai berikut.

3.1.1 PERANGKAT KERAS (*HARDWARE*)

Perangkat keras yang digunakan pada penelitian ini yaitu sebuah laptop. Adapun spesifikasi laptop yang digunakan adalah:

Tabel 3. 1 Spesifikasi Laptop

No	Detail	Spesifikasi
1.	Merek (*tipe*)	Asus A456U
2.	*Prosesor*	Intel(R) Core TM i5-7200U CPU @ 2.50GHz 2.71GHz
3.	*Installed memory* / RAM	4GB
4.	*Tipe system*	64-bitt
5.	*Operating System*/OS	Ubuntu 20.04.3 LTS
6.	VGA	NVIDIA GeForece 930 MX

3.1.2 PERANGKAT LUNAK (SOFTWARE)

Perangkat luanak yang digunakan pada penelitian ini yaitu sebagai berikut :

Tabel 3. 2 Perangkat lunak yang digunakan

No	Nama	Detail
1	*Spyder*	5.3.1
2	Bahasa pemrograman	Python 3.8
3	*Library*	*time, pandas, numpy, tensorflow, keras, sklearn, mlxtend plotting, matplotlib*.

3.1.3 DATA SET SINYAL EEG

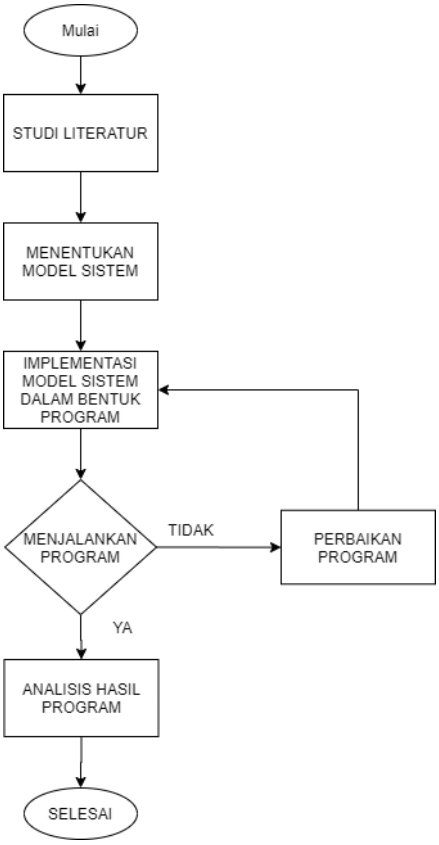
Penulisan ini menggunakan dataset sinyal EEG yang diperoleh dari penelitian [10]. Proses perekaman *Electroenchepalography* untuk imajinasi pergerakan jari tangan menggunakan sistem EEG-1200 JE-921A EEG (Nihon, Kohden, Jepang). Perekaman sinyal EEG terdapat 22 *elektrode* yang akan di tempelkan pada kulit kepala diantaranya Fp1, Fp2, F3, F4, C3, C4, P3, P4, O1, O2, A1, A2, F7, F8, T3, T4, T5, T6, Fz, Cz, Pz, X5. Dari 22 *elektrode* yang di tempelkan kepada kulit kepala terdapat 3 *elektrode* yang tidak dapat digunakan untuk tahap klasifikasi dikarenakan *elektrode* A1 dan A2 merupakan *referensi* dan *elektrode* X5 merupakan *sinkronisasi*. Pada penulisan ini menggunakan data dengan label 5F (5 *Fingers*) untuk membayangkan pergerakan (*Motor imagery*) lima jari yaitu : ibu jari, jari telunjuk, jari tengah, jari manis, dan jari kelingking. Sinyal EEG yang diperoleh dari perekaman ini disimpan dan ditandai menggunakan sinyal marker untuk menandakan imajinasi pergerakan pada jari tangan.

Tabel 3. 3 Nilai *marker*

Marker	Keterangan
1	Ibu Jari
2	Jari Telunjuk
3	Jari Tengah
4	Jari Manis
5	Jari Kelingking

3.2 DIGRAM ALUR PENELITIAN

Diagram alir penelitian mempresentasikan gambaran tahapan yang akan dilakukan peneliti pada penelitian ini. Adapun diagram alir dari penelitian ini disajikan pada gambar sebagai berikut.



Gambar 3. 1 Alir Penelitian

3.2.1 STUDI LITERATURE

Studi literature untuk mencari sumber *referensi* yang terkait dengan penelitian ini. Sumber yang digunakan berupa buku, jurnal, situs stack overflow dan GitHub.

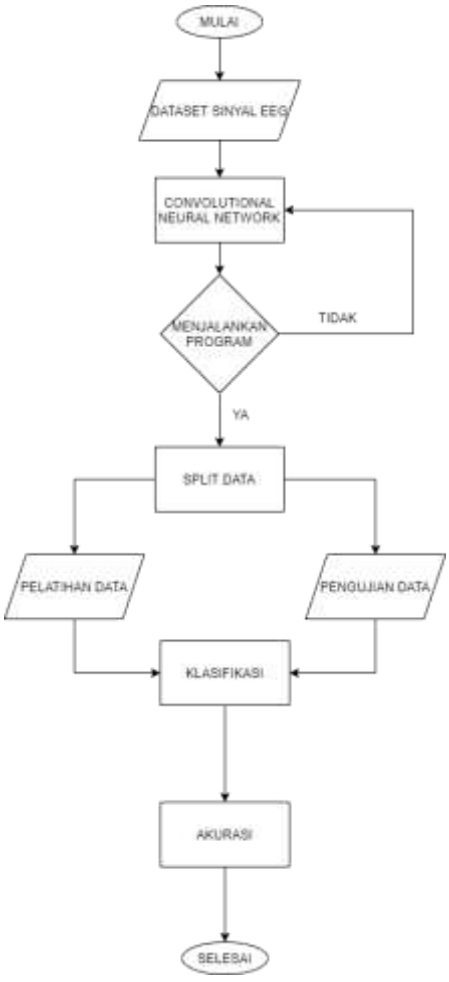
3.2.2 MENENTUKAN MODEL SISTEM

Dalam penelitian ini akan membahas terkait klasifikasi imajinasi pergerakan jari tangan dengan menggunakan *metode deep learning* yaitu *one dimensional convolutional neural network* (CNN) yang mampu melakukan pembelajaran mandiri untuk melakukan klasifikasi imajinasi pada jari tangan.

Metode ini memiliki dua tahapan yang akan di lakukan. Tahapan pertama yaitu *feature learning* dalam tahapan ini terdapat dua lapisan utama yaitu *convolution layer* dan *pooling layer*. Setelah melalui lapisan utama terdapat fungsi aktivasi yang akan digunakan pada penelitian ini menggunakan fungsi aktivasi *ReLu*. Tahapan kedua yaitu *classification* pada tahapan ini terdapat 1 *layer fully connected* dan fungsi aktivasi yang digunakan yaitu *softmax*. Setelah menentukan

metode yang akan digunakan peneliti meriset kode program dari penelitian sebelumnya. Peneliti memodifikasi kode program sesuai dengan kebutuhan pada penelitian ini. Selanjutnya akan menjalankan program pada *sypder* 3.8 atau google colab jika kode program tidak berjalan dan mengalami kesalahan maka peneliti akan meriset kembali kode program yang bermasalah akan tetapi jika kode program tidak mengalami kendala makan peneliti akan melanjutkan ketahapan berikutnya.

3.2.3 IMPLEMENTASI MODEL SISTEM



Gambar 3. 2 Impementasi model sistem

A. DATA SET SINYAL EEG

Hasil perekaman sinyal EEG imajinasi pergerakan lima jari digunakan sebagai data set dengan panjang sampel yang berbeda. Hal tersebut mengakibatkan masukan *one dimensional convolution neural network* tidak dapat langsung menggunakan data set hasil perekaman karna setiap data memiliki panjang set yang

berbeda oleh karna itu diperlukan proses segmentasi pada sinyal EEG agar memiliki panjang sampel yang sama. Proses segmentasi pada sinyal EEG menggunakan panjang sampel 200Hz. Data yang digunakan pada proses pelatian dan pengujian sebanyak 5700 sinyal yang diperoleh dari 6 subjek diantaranya data kelas 1, 2, 3, 4, 5, dan 6. Berikut adalah kode program proses segmentasi pada sinyal EEG.

```
data1_numpy = data1.to_numpy()
a = 0
data_test_1 = np.zeros((959,200,19))
daftar_kelas_1 = np.zeros((959,1))
for x in range(len(kelas1)):
    if kelas1.iloc[x,0]==1 or kelas1.iloc[x,0]==2 or kelas1.iloc[x,0]==3
or kelas1.iloc[x,0]==4 or kelas1.iloc[x,0]==5:
        daftar_kelas_1[a:a+1,0] = kelas1.iloc[x,0]
        batas_bawah = kelas1.iloc[x,1]
        data_test_1[a:a+1,:,:]=
data1_numpy[batas_bawah+20:batas_bawah+220,2:21]
        a = a + 1
data2_numpy = data2.to_numpy()
a = 0
data_test_2 = np.zeros((958,200,19))
daftar_kelas_2 = np.zeros((958,1))
for x in range(len(kelas2)):
    if kelas2.iloc[x,0]==1 or kelas2.iloc[x,0]==2 or kelas2.iloc[x,0]==3
or kelas2.iloc[x,0]==4 or kelas2.iloc[x,0]==5:
        daftar_kelas_2[a:a+1,0] = kelas2.iloc[x,0]
        batas_bawah = kelas2.iloc[x,1]
        data_test_2[a:a+1,:,:]=
data2_numpy[batas_bawah+20:batas_bawah+220,2:21]
        a = a + 1
data3_numpy = data3.to_numpy()
a = 0
data_test_3 = np.zeros((959,200,19))
daftar_kelas_3 = np.zeros((959,1))
for x in range(len(kelas3)):
    if kelas3.iloc[x,0]==1 or kelas3.iloc[x,0]==2 or kelas3.iloc[x,0]==3
or kelas3.iloc[x,0]==4 or kelas3.iloc[x,0]==5:
        daftar_kelas_3[a:a+1,0] = kelas3.iloc[x,0]
        batas_bawah = kelas3.iloc[x,1]
        data_test_3[a:a+1,:,:]=
data3_numpy[batas_bawah+20:batas_bawah+220,2:21]
        a = a + 1
data4_numpy = data4.to_numpy()
a = 0
data_test_4 = np.zeros((958,200,19))
daftar_kelas_4 = np.zeros((958,1))
for x in range(len(kelas4)):
    if kelas4.iloc[x,0]==1 or kelas4.iloc[x,0]==2 or kelas4.iloc[x,0]==3
or kelas4.iloc[x,0]==4 or kelas4.iloc[x,0]==5:
        daftar_kelas_4[a:a+1,0] = kelas4.iloc[x,0]
        batas_bawah = kelas4.iloc[x,1]
```

```
        data_test_4[a:a+1,:,:]=
data4_numpy[batas_bawah+20:batas_bawah+220,2:21]
        a = a + 1
data5_numpy = data5.to_numpy()
a = 0
data_test_5 = np.zeros((957,200,19))
daftar_kelas_5 = np.zeros((957,1))
for x in range(len(kelas5)):
    if kelas5.iloc[x,0]==1 or kelas5.iloc[x,0]==2 or kelas5.iloc[x,0]==3
or kelas5.iloc[x,0]==4 or kelas5.iloc[x,0]==5:
        daftar_kelas_5[a:a+1,0] = kelas5.iloc[x,0]
        batas_bawah = kelas5.iloc[x,1]
        data_test_5[a:a+1,:,:]=
data5_numpy[batas_bawah+20:batas_bawah+220,2:21]
        a = a + 1
data6_numpy = data6.to_numpy()
a = 0
data_test_6 = np.zeros((958,200,19))
daftar_kelas_6 = np.zeros((958,1))
for x in range(len(kelas6)):
    if kelas6.iloc[x,0]==1 or kelas6.iloc[x,0]==2 or kelas6.iloc[x,0]==3
or kelas6.iloc[x,0]==4 or kelas6.iloc[x,0]==5:
        daftar_kelas_6[a:a+1,0] = kelas6.iloc[x,0]
        batas_bawah = kelas6.iloc[x,1]
        data_test_6[a:a+1,:,:]=
data6_numpy[batas_bawah+20:batas_bawah+220,2:21]
        a = a +
data = np.concatenate([data_test_1[0:950,:,:], data_test_2[0:950,:,:],
data_test_3[0:950,:,:], data_test_4[0:950,:,:], data_test_5[0:950,:,:],
data_test_6[0:950,:,:]])
daftar_kelas = np.concatenate([daftar_kelas_1[0:950,0:1],
daftar_kelas_2[0:950,0:1], daftar_kelas_3[0:950,0:1],
daftar_kelas_4[0:950,0:1], daftar_kelas_5[0:950,0:1],
daftar_kelas_6[0:950,0:1]])
kelas_full_min1 = daftar_kelas - 1
kelas_full = to_categorical(kelas_full_min1)
```

B. *CONVOLUTIONAL NEURAL NETWORK* (CNN)

Peneliti menggunakan metode *one-dimensional convolutional neural network*. Metode ini akan melakukan beberapa tahapan yang dilakukan untuk mendapatkan hasil dari *inputan* yang diberikan. Pada penelitian ini melakukan evaluasi *hyper parameter one dimensional convolutional neural network* dengan memvariasikan *karnel* pada laisan konvolusi.

Tabel 3. 4 Konfigurasi hyper parameter 1D-CNN

Hyper Parameter	*Konfigurasi*
Jumlah *Karnel*	5-25
Panjang *Karnel*	1-5
Pooling	*Maximum pooling*
Jumlah *Karnel*	1-5
Panjang *Karnel*	1-5
Dense	64
Fungsi Aktivasi	ReLu
Neuron Output Layer	5
Optimasi MPL	Adam

Pada tabel 3.4 Perbedaan setiap *karnel* pada proses konvolusi dilakukan untuk mengetahui pengaruhnya terhadap nilai akurasi klasifikasi pergerakan jari tangan. Berikut kode program yang digunakan pada model *one-dimensional convolutional neural network*.

```
model = Sequential()
    model.add(Conv1D(filters=n, kernel_size=n, activation='relu',
input_shape=(n_timesteps,n_features)))
    model.add(Dropout(0.20))
    model.add(MaxPooling1D(pool_size=1))
    model.add(Conv1D(filters=1, kernel_size=1))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.20))
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.20))
    model.add(Dense(n_outputs, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='Adam',
metrics=['accuracy'])
```

C. TAHAPAN PELATIHAN DAN PENGUJIAN SISTEM

Pada tahapan pelatihan dan pengujian sistem *one-dimensional convolutional neural network* untuk klasifikasi sinyal EEG pergerakan lima jari tangan menggunakan *cross validation*. Pada proses *Kflod cross validation* untuk nilai k-nya sama dengan 10 yang nantinya akan di bagi menjadi data latih dan data uji. 9 bagian akan digunakan sebagai data latih dan 1 bagian akan digunakan sebagai data uji. Berikut *flowchart* tahapan pelatihan dan pengujian sistem.

Pada proses pelatihan *one-dimensional convolutional neural network* akan menghasilkan kesalahan klasifikasi yang akan di tunjukan pada grafik *traning loss*. Setelah proses pelatihan data pada model *one-dimensional convolution neural network* selanjutnya melakukan proses pengujian data yang akan di tampilkan pada *confusion matrix*. Berikut kode program yang digunakan pada tahapan pelatihan dan pengujian data.

```
seed = 7
np.random.seed(seed)
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
cvscores = []
class_names = ['Ibu Jari', 'Jari Telunjuk', 'Jari Tengah', 'Jari Manis',
'Jari Kelingking']
for train, test in kfold.split(data, daftar_kelas):
    verbose, epochs, batch_size = 1, 100, 50
    n_timesteps, n_features, n_outputs = data.shape[1], data.shape[2],
kelas_full.shape[1]
    history = model.fit(data[train,:,:], kelas_full[train,:],
epochs=epochs, batch_size=batch_size, verbose=verbose,
validation_split=0.2)
    accuracy = history.history['accuracy']
    val_accuracy = history.history['val_accuracy']
    epochs = range(1, len(accuracy) + 1)
    plt.figure()
    plt.plot(epochs, accuracy, 'b', label='Training')
    plt.plot(epochs, val_accuracy, 'r', label='Validation')
    plt.title('Training & Validation acc')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()
    y_pred=np.argmax(model.predict(data[test,:,:]), axis=-1)
    cm=confusion_matrix(kelas_full_min1[test],y_pred)
    print(cm)
    fig, ax = plot_confusion_matrix(conf_mat=cm,
                                    colorbar=False,
                                    show_absolute=True,
                                    show_normed=False,
                                    class_names=class_names)
    plt.show()
    fig, ax = plot_confusion_matrix(conf_mat=cm,
                                    colorbar=False,
                                    show_absolute=False,
                                    show_normed=True,
                                    class_names=class_names)
    plt.show()
    scores = model.evaluate(data[test,:,:], kelas_full[test,:],
batch_size=batch_size, verbose=verbose)
    print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    cvscores.append(scores[1] * 100)
```

3.3 TINGKAT AKURASI

Pada tahapan ini data hasil pengujian akan memperlihatkan tingkat akurasi di setiap *Kfold* dari satu hingga sepuluh yang akan di tampilkan pada *confusion matrix*. Pada penelitian ini akan dilakukan dua tahapan proses konvolusi dengan tujuan untuk mendapatkan tingkat akurasi terbaik. Pada penelitian ini akan menampilkan rata rata akurasi dari setiap *karnel*. Berikut kode program untuk menampilkan rata-rata akurasi.

```
scores = model.evaluate(data[test,:,:], kelas_full[test,:],
batch_size=batch_size, verbose=verbose)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    cvscores.append(scores[1] * 100)
print("%.2f%% (+/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
end_time = time.perf_counter()
print (end_time, "seconds - All CNN Processing time")
```

Pada penelitian ini akan menampilkan hasil data prediksi setiap kelas pada *confusion matrix multiclass*. Berikut tabel confusion matrix untuk lima kelas jari tangan.

Tabel 3. 5 *Confusion matrix* pada lima kelas jari tangan

		Predicted Clasification				
Actual Clasification	Class	Ibu Jari	Jari Telunjuk	Jari Tengah	Jari Manis	Jari Kelingking
	Ibu Jari	TP	FN	FN	FN	FN
	Jari Telunjuk	FP	TP	TN	TN	TN
	Jari Tengah	FP	TN	TP	TN	TN
	Jari Manis	FP	TN	TN	TP	TN
	Jari Kelingking	FP	TN	TN	TN	TP

Keterangan :

TP (*True Positive*) : kondisi dimana hasil prediksi sesuai dengan kelas sebenarnya

TN (*True Negative*) : kondisi dimana hasil prediksi tidak sesuai dengan kelas sebenarnya

FP (*False Positive*) : kondisi dimana bukan kelas sebenarnya tapi di prediksi sebagai kelas sebenarnya

FN (*False Negative*) : kondisi dimana seharusnya kelas sebenernya tapi di prediksi bukan kelas sebenarnya