



LAMPIRAN

Variables - base_data																																																		
PLOTS		VARIABLE		VIEW																																														
New from Selection	Open	Rows	Columns	Insert	Delete	Sort	Transpose																																											
base_data																																																		
12x54 double																																																		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21																														
1	-0.0299	-0.0181	0.0172	-0.1878	0.0070	-0.1237	-0.0448	-0.0918	-1.5856	-0.4149	-0.0298	0.0114	0.0394	-0.0474	-0.2121	-0.0730	-0.1335	-0.1296	-0.0338	-0.2402	-0.0419																													
2	-0.2802	-0.4090	-0.1635	-0.3768	-0.1499	-0.3124	-0.1545	-0.3307	-0.4114	-0.1321	-0.2204	-0.1429	-0.0440	-0.2223	-0.0870	-0.3406	-0.5555	-0.7684	-0.5160	-0.8566	-0.5157																													
3	-0.1580	-0.1406	-0.0934	-0.2059	-0.1638	-0.1655	-0.1341	-0.2365	-1.1667	-0.3826	-0.1906	-0.0970	-0.1427	-0.1115	-0.3201	-0.1793	-0.3055	-0.2138	-0.2754	-0.3432	-0.2694																													
4	0.4173	0.6257	0.6136	0.7716	0.6021	0.7429	0.7869	0.7668	-0.2692	0.8096	0.7095	0.6584	0.7218	0.8236	0.0580	0.7820	-0.1741	0.1213	0.2372	0.2293	0.2715																													
5	-0.0771	0.1675	0.2449	0.1492	0.2533	0.1501	0.2467	0.3042	-1.3580	0.3012	0.1308	0.2890	0.2386	0.4999	-0.6546	0.2058	-0.1378	-0.0262	-0.1433	-0.0783	0.1195																													
6	-0.2556	-0.3741	-0.0739	-0.3382	-0.0992	-0.2675	-0.0387	-0.1932	-0.2568	-0.1776	-0.0376	-0.1343	-0.1665	-0.4914	-0.2402	-0.1866	-0.3421	-0.1366	-0.3240	-0.1413																														
7	-0.4903	-0.1340	-0.2186	-0.4150	-0.3120	-0.4997	-0.4166	-0.4978	-1.4435	-0.7347	-0.5905	-0.5255	-0.4645	-0.4168	-0.8078	-0.8223	-0.9705	-0.9645	-0.9196	-0.7389	-0.7236																													
8	-0.4667	-0.5127	-0.1623	-0.4756	-0.2780	-0.3725	-0.1009	-0.3635	-0.6373	0.2290	-0.1961	-0.2953	-0.1315	-0.1168	-0.7659	-0.4023	-0.6035	-0.6470	-0.3161	-0.6505	-0.3945																													
9	-0.1087	-0.3083	-0.2310	-0.3552	-0.2459	-0.3305	-0.1564	-0.1730	-0.6224	-0.1318	-0.2410	-0.2657	-0.0804																																					

Variables - label_1

Plots VARIABLE View

New from Selection Open Rows Columns Insert Delete Transpose

1 1

data data_seconds_list label_1

1x60 double

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2																				
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				
11																				
12																				
13																				
14																				
15																				
16																				
17																				
18																				
19																				
20																				
21																				
22																				
23																				
24																				
25																				
26																				
27																				
28																				

Variables - label_2

Plots VARIABLE View

New from Selection Open Rows Columns Insert Delete Transpose

1 1

data data_seconds_list label_1 label_2

1x60 double

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2																				
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				
11																				
12																				
13																				
14																				
15																				
16																				
17																				
18																				
19																				
20																				
21																				
22																				
23																				
24																				
25																				
26																				
27																				
28																				

Variables - label_3

Plots VARIABLE View

New from Selection Open Rows Columns Insert Delete Transpose

1 1

data data_seconds_list label_1 label_2 label_3

1x60 double

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0
2																				
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				
11																				
12																				
13																				
14																				
15																				
16																				
17																				
18																				
19																				
20																				
21																				
22																				
23																				
24																				
25																				
26																				
27																				
28																				

Lampiran 2

The top panel of the Jupyter Notebook displays a 4-D array of zeros, created using the following code:

```
val(:, :, 1, 1) =
```

The array is visualized as a 4D plot with axes labeled 0, 1, 2, and 3. The plot shows a grid of 21x21x21x21 points, all of which are zero. The bottom panel shows the variable explorer with the following variables:

- data
- data_second_list
- label_1

The bottom panel also displays a table with 22 columns (labeled 0 to 21) and 22 rows (labeled 1 to 22). The first row (row 1) contains the values 0, 0. The remaining rows (rows 2 to 22) are empty.

Lampiran 3. Proses Segmentasi Data



```
1 def read_file(file):  
2     data = sio.loadmat(file)  
3     return data  
4 trial_signal = data[:, trial][0][:, channel]  
5 base_signal = data[:, trial][0][:300, channel]
```



Lampiran 4. Proses Dekomposisi Data



```
1 def butter_bandpass(lowcut, highcut, fs, order=5):
2     nyq = 0.5 * fs
3     low = lowcut / nyq
4     high = highcut / nyq
5     b, a = butter(order, [low, high], btype='band')
6     return b, a
7
8 def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
9     b, a = butter_bandpass(lowcut, highcut, fs, order=order)
10    y = lfilter(b, a, data)
11    return y
12
13 #Dekomposisi Baseline
14 base_theta = butter_bandpass_filter(base_signal, 4, 8, frequency, order=3)
15     base_alpha = butter_bandpass_filter(base_signal, 8, 14, frequency, order=3)
16     base_beta = butter_bandpass_filter(base_signal, 14, 31, frequency, order=3)
17     base_gamma = butter_bandpass_filter(base_signal, 31, 45, frequency, order=3)
18
19 #Dekomposisi Trial
20 theta = butter_bandpass_filter(trial_signal, 4, 8, frequency, order=3)
21     alpha = butter_bandpass_filter(trial_signal, 8, 14, frequency, order=3)
22     beta = butter_bandpass_filter(trial_signal, 14, 31, frequency, order=3)
23     gamma = butter_bandpass_filter(trial_signal, 31, 45, frequency, order=3)
```



Lampiran 5. Proses *Feature Extraction* Data

```
1 for trial in range(12):
2     temp_base_DE = np.empty([0])
3     temp_base_theta_DE = np.empty([0])
4     temp_base_alpha_DE = np.empty([0])
5     temp_base_beta_DE = np.empty([0])
6     temp_base_gamma_DE = np.empty([0])
7
8     data_seconds = (data[:, trial][0].shape[0])
9     temp_de = np.empty([0, data_seconds])
10
11     for channel in range(16):
12
13         #Feature Extraction
14         base_theta_DE =(compute_DE(base_theta[:100])+compute_DE(base_theta[100:200])+compute_DE(base_theta[200:]))/3
15         base_alpha_DE =(compute_DE(base_alpha[:100])+compute_DE(base_alpha[100:200])+compute_DE(base_alpha[200:]))/3
16         base_beta_DE =(compute_DE(base_beta[:100])+compute_DE(base_beta[100:200])+compute_DE(base_beta[200:]))/3
17         base_gamma_DE =(compute_DE(base_gamma[:100])+compute_DE(base_gamma[100:200])+compute_DE(base_gamma[200:]))/3
18
19         temp_base_theta_DE = np.append(temp_base_theta_DE, base_theta_DE)
20         temp_base_gamma_DE = np.append(temp_base_gamma_DE, base_gamma_DE)
21         temp_base_beta_DE = np.append(temp_base_beta_DE, base_beta_DE)
22         temp_base_alpha_DE = np.append(temp_base_alpha_DE, base_alpha_DE)
23
24         DE_theta = np.zeros(shape=[0], dtype = float)
25         DE_alpha = np.zeros(shape=[0], dtype = float)
26         DE_beta = np.zeros(shape=[0], dtype = float)
27         DE_gamma = np.zeros(shape=[0], dtype = float)
28
29         for index in range(data_seconds):
30             DE_theta =np.append(DE_theta, compute_DE(theta[index*frequency:(index+1)*frequency]))
31             DE_alpha =np.append(DE_alpha, compute_DE(alpha[index*frequency:(index+1)*frequency]))
32             DE_beta =np.append(DE_beta, compute_DE(beta[index*frequency:(index+1)*frequency]))
33             DE_gamma =np.append(DE_gamma, compute_DE(gamma[index*frequency:(index+1)*frequency]))
34
35             temp_de = np.vstack([temp_de, DE_theta])
36             temp_de = np.vstack([temp_de, DE_alpha])
37             temp_de = np.vstack([temp_de, DE_beta])
38             temp_de = np.vstack([temp_de, DE_gamma])
39
40             temp_trial_de = temp_de.reshape(-1, 4,)
41             decomposed_de = np.vstack([decomposed_de, temp_trial_de])
42
43         temp_base_DE = np.append(temp_base_theta_DE, temp_base_alpha_DE)
44         temp_base_DE = np.append(temp_base_DE, temp_base_beta_DE)
45         temp_base_DE = np.append(temp_base_DE, temp_base_gamma_DE)
46         base_DE = np.vstack([base_DE, temp_base_DE])
47         data_seconds_list = np.append(data_seconds_list, data_seconds)
48         decomposed_de = decomposed_de.reshape(-1, 16, 4).transpose([0, 2, 1]).reshape(-1, 4, 16).reshape(-1, 64)
49         print("base_DE shape:", base_DE.shape)
50         print("trial_DE shape:", decomposed_de.shape)
51         return base_DE, decomposed_de, data_seconds_list
```


Lampiran 6. Proses *Baseline Reduction*



```
1 def read_file(file):
2     data = sio.loadmat(file)
3     return (data['data'], data["base_data"], data["label_1"],
4           data["label_2"], data["label_3"], data["data_seconds_list"])
5
6 def get_vector_deviation(trial_vec, base_vec):
7     return trial_vec / base_vec
8
9 def get_dataset_deviation(trial_data, base_data, seconds_list):
10    new_dataset = np.empty((0, 64))
11    idx = 0
12    for i, dur in enumerate(seconds_list[0]):
13        for j in range(int(dur)):
14            diff = get_vector_deviation(trial_data[idx + j], base_data[i])
15            new_dataset = np.vstack([new_dataset, diff.reshape(1, 64)])
16        idx += int(dur)
17    return new_dataset
```



Lampiran 7. Proses *Feature Representation*

```
1 def data_1Dto2D(data, Y=9, X=9):
2     data_2D = np.zeros([Y, X])
3     data_2D[0] = (0, 0, 0, data[0], 0, data[1], 0, 0, 0)
4     data_2D[1] = (0, 0, 0, 0, 0, 0, 0, 0, 0)
5     data_2D[2] = (data[8], 0, data[2], 0, 0, 0, data[3], 0, data[9])
6     data_2D[3] = (0, 0, 0, 0, 0, 0, 0, 0, 0)
7     data_2D[4] = (data[10], 0, data[4], 0, 0, 0, data[5], 0, data[11])
8     data_2D[5] = (0, 0, 0, 0, 0, 0, 0, 0, 0)
9     data_2D[6] = (data[12], 0, data[6], 0, 0, 0, data[7], 0, data[13])
10    data_2D[7] = (0, 0, 0, 0, 0, 0, 0, 0, 0)
11    data_2D[8] = (0, 0, 0, data[14], 0, data[15], 0, 0, 0)
12    return data_2D
13
14 def pre_process(path, use_baseline='yes'):
15     sub_len = 16
16     data_3D = np.empty((0, 9, 9))
17
18     trial_data, base_data, v_label, a_label, d_label, seconds_list = read_file(path)
19
20     if use_baseline == "yes":
21         data = get_dataset_deviation(trial_data, base_data, seconds_list)
22     else:
23         data = trial_data
24
25     data = preprocessing.scale(data, axis=1)
26
27     for vec in data:
28         for band in range(4):
29             temp_2D = data_1Dto2D(vec[band*sub_len:(band+1)*sub_len])
30             data_3D = np.vstack([data_3D, temp_2D.reshape(1, 9, 9)])
31
32     data_3D = data_3D.reshape(-1, 4, 9, 9) # (N, 4, 9, 9)
33     print("Final shape:", data_3D.shape)
34     return data_3D, v_label, a_label, d_label
```



Lampiran 8. Proses *Classification*, dan *Accuracy Calculation*

```
1 import os
2 import numpy as np
3 import pandas as pd
4 import scipy.io as sio
5 import matplotlib.pyplot as plt
6 from sklearn.model_selection import KFold, train_test_split
7 from sklearn.preprocessing import LabelEncoder
8 from sklearn.metrics import precision_score, recall_score, f1_score
9 from tensorflow.keras.models import Sequential
10 from tensorflow.keras.layers import Conv2D, Dense, Flatten, Dropout
11 from tensorflow.keras.optimizers import Adam
12 from tensorflow.keras.utils import to_categorical
13 from tensorflow.keras.regularizers import l2
14 from tensorflow.keras.callbacks import EarlyStopping
15
16 input_height = 9
17 input_width = 9
18 time_step = 1
19 window_size = 1
20 with_or_not = "with"
21 inputs = [1, 2, 3, 4]
22 bands = [i - 1 for i in inputs]
23 input_channel_num = len(bands) * time_step
24 dataset_dir = "Dataset_3D/"
25 mat_files = sorted([f for f in os.listdir(dataset_dir) if f.endswith(".mat") and f.startswith("DE_")])
26
27 for file_name in mat_files:
28     input_file = file_name.replace(".mat", "")
29     print("\n" + "=" * 50)
30     print(f"Processing file: {input_file}")
31     print("=" * 50)
32
33     data_file = sio.loadmat(os.path.join(dataset_dir, input_file + ".mat"))
34     cnn_datasets = data_file["data"]
35     label_valence = data_file['label_1'][0]
36     label_arousal = data_file['label_2'][0]
37     label_dominance = data_file['label_3'][0]
38
39     def preprocess_three_labels(v, a, d):
40         mapping = {
41             (0, 0, 0): '000',
42             (0, 0, 1): '001',
43             (0, 1, 0): '010',
44             (0, 1, 1): '011',
45             (1, 0, 0): '100',
46             (1, 1, 0): '110',
47         }
48         return mapping.get((v, a, d), 'invalid')
49
50     raw_labels = list(map(preprocess_three_labels, label_valence, label_arousal, label_dominance))
51     valid_idx = [i for i, lbl in enumerate(raw_labels) if lbl != 'invalid']
52
53     cnn_datasets = cnn_datasets[valid_idx]
54     raw_labels = [raw_labels[i] for i in valid_idx]
55
56     encoder = LabelEncoder()
57     encoded = encoder.fit_transform(raw_labels)
58     labels = to_categorical(encoded)
59
60     cnn_datasets = cnn_datasets.transpose(0, 2, 3, 1)
61     cnn_datasets = cnn_datasets[:, :, :, bands]
62     cnn_datasets = cnn_datasets.reshape(-1, input_height, input_width, input_channel_num)
63
64     indices = np.arange(len(labels))
65     np.random.shuffle(indices)
66     cnn_datasets = cnn_datasets[indices]
67     labels = labels[indices]
68
69     def build_model(input_shape, num_classes=6, enable_penalty=True):
70         model = Sequential()
71         model.add(Conv2D(64, kernel_size=4, strides=1, padding='same', activation='relu',
72             input_shape=input_shape))
73         model.add(Conv2D(128, kernel_size=4, strides=1, padding='same', activation='relu'))
74         model.add(Conv2D(256, kernel_size=4, strides=1, padding='same', activation='relu'))
75         model.add(Conv2D(64, kernel_size=1, strides=1, padding='same', activation='relu'))
76         model.add(Flatten())
77         model.add(Dense(1024, activation='relu'))
78         if enable_penalty:
79             model.add(Dense(num_classes, activation='softmax', activity_regularizer=l2(0.5)))
80         else:
81             model.add(Dense(num_classes, activation='softmax'))
82     return model
```

```

1 # === K-FOLD TRAINING ===
2 labels_argmax = np.argmax(labels, axis=1)
3 kf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
4
5 fold = 1
6 accuracy_list, precision_list, recall_list, f1_list, epoch_list = [], [], [], [], []
7
8 result_dir = os.path.join("CNN Result", f"with_or_not_{input_file}_Kernel2x2")
9 os.makedirs(result_dir, exist_ok=True)
10
11 model = build_model(cnn_datasets.shape[1:], labels.shape[1])
12 model.compile(loss="categorical_crossentropy", optimizer=Adam(1e-4), metrics=['accuracy'])
13 initial_weights = os.path.join(result_dir, f"initial_model_{input_file}.weights.h5")
14 model.save_weights(initial_weights)
15 model.summary()
16
17 for train_val_idx, _ in kf.split(cnn_datasets, labels_argmax):
18     model.load_weights(initial_weights)
19
20     print(f"==== Training Fold {fold} ====")
21
22     X_train, X_val, y_train, y_val = train_test_split(
23         cnn_datasets[train_val_idx], labels[train_val_idx],
24         test_size=0.1, stratify=labels_argmax[train_val_idx],
25         random_state=fold
26     )
27
28     early_stop = EarlyStopping(
29         monitor='val_accuracy',
30         patience=15,
31         verbose=1,
32         restore_best_weights=True
33     )
34
35     hist = model.fit(
36         X_train, y_train,
37         batch_size=100, epochs=50,
38         validation_data=(X_val, y_val),
39         callbacks=[early_stop],
40         verbose=1
41     )
42
43     # === Evaluation ===
44     loss, accuracy = model.evaluate(X_val, y_val, verbose=0)
45     y_pred = model.predict(X_val)
46     y_pred_classes = np.argmax(y_pred, axis=1)
47     y_true = np.argmax(y_val, axis=1)
48
49     precision = precision_score(y_true, y_pred_classes, average='macro', zero_division=0)
50     recall = recall_score(y_true, y_pred_classes, average='macro', zero_division=0)
51     f1 = f1_score(y_true, y_pred_classes, average='macro', zero_division=0)
52     total_epochs = len(hist.epoch)
53
54     accuracy_list.append(accuracy)
55     precision_list.append(precision)
56     recall_list.append(recall)
57     f1_list.append(f1)
58     epoch_list.append(total_epochs)
59
60     print(f'Fold {fold}: Acc={accuracy:.4f}, Prec={precision:.4f}, Recall={recall:.4f}, F1={f1:.4f}, Epochs={total_epochs}')
61
62 # === Save Model ===
63
64 # === Save History and Metrics to Excel ===
65 df_result = pd.DataFrame({
66     'epoch': hist.epoch,
67     'loss': hist.history['loss'],
68     'val_loss': hist.history['val_loss'],
69     'accuracy': hist.history['accuracy'],
70     'val_accuracy': hist.history['val_accuracy'],
71 })
72
73 final_metrics = pd.DataFrame({
74     'final_accuracy': [accuracy],
75     'final_precision': [precision],
76     'final_recall': [recall],
77     'final_f1': [f1],
78     'total_epoch': [total_epochs]
79 })
80
81 with pd.ExcelWriter(os.path.join(result_dir, f'{input_file}_fold_{fold}.xlsx')) as writer:
82     df_result.to_excel(writer, sheet_name='training_curve', index=False)
83     final_metrics.to_excel(writer, sheet_name='final_metrics', index=False)
84
85 # === Save Plot ===
86 plt.figure(figsize=(10, 5))
87 plt.plot(hist.history['loss'], label='Train Loss')
88 plt.plot(hist.history['val_loss'], label='Val Loss')
89 plt.title(f'Loss Curve - Fold {fold}')
90 plt.xlabel('Epoch')
91 plt.ylabel('Loss')
92 plt.legend()
93 plt.tight_layout()
94 plt.savefig(os.path.join(result_dir, f'{input_file}_fold_{fold}_loss.png'))
95 plt.close()
96
97 fold += 1
98
99 # === Save Summary ===
100 print("\n==== FINAL AVERAGE METRICS ====")
101 print(f'Average Accuracy : {np.mean(accuracy_list):.4f}')
102 print(f'Average Precision: {np.mean(precision_list):.4f}')
103 print(f'Average Recall : {np.mean(recall_list):.4f}')
104 print(f'Average F1-score : {np.mean(f1_list):.4f}')
105 print(f'Average Epochs : {np.mean(epoch_list):.2f}')
106
107 summary_df = pd.DataFrame({
108     'Accuracy': accuracy_list,
109     'Precision': precision_list,
110     'Recall': recall_list,
111     'F1-Score': f1_list,
112     'Epoch': epoch_list
113 })
114 summary_df.loc['Average'] = summary_df.mean(numeric_only=True)
115 summary_df.to_excel(os.path.join(result_dir, f'{input_file}_summary.xlsx'))
116

```

RIWAYAT HIDUP



Made Agastya Maheswara, lahir di Singaraja pada tanggal 16 Desember 2002. Penulis lahir dari pasangan suami istri Bapak Nyoman Suriawan dan Ibu Ketut Sri Indrawati. Penulis merupakan anak kedua dari tiga bersaudara. Penulis berkewarganegaraan Indonesia dan beragama Hindu. Pada tahun 2009 – 2015 penulis menempuh pendidikan Sekolah Dasar di SD Negeri 2 Nyitdah, Kediri, Tabanan. Pada tahun 2015 – 2018 penulis melanjutkan pendidikan Sekolah Menengah Pertama di SMP Negeri 2 Kediri. Selanjutnya, pada tahun 2018 – 2021 penulis menempuh pendidikan Sekolah Menengah Atas di SMA Negeri 1 Tabanan. Pada tahun 2021 penulis melanjutkan pendidikan tinggi di Universitas Pendidikan Ganesha pada Fakultas Teknik dan Kejuruan (FTK), Jurusan Teknik Informatika, Program Studi Sistem Informasi. Dengan tekad, ketekunan, serta semangat belajar yang tinggi, penulis berhasil menyelesaikan skripsi yang berjudul *“IMPLEMENTASI METODE CONTINUOUS CONVOLUTIONAL NEURAL NETWORK UNTUK PENGENALAN POLA PENGANGGE SUARA AKSARA BALI PADA SINYAL ELECTROENCEPHALOGRAM”* pada tahun 2025. Penulis berharap karya ini dapat memberikan kontribusi positif dalam pengembangan penelitian di bidang Sistem Informasi, khususnya pada pengolahan sinyal EEG dan pengenalan pola aksara Bali.