Laporan UAS Pengantar Pembelajaran Mesin

Implementasi CNN dalam Klasifikasi Sinyal EEG untuk Mengetahui Kemampuan Kognitif Seseorang

Nama Kelompok:

- I Made Alit Darma Putra (2008561045)
- I Nengah Oka Darmayasa (2008561070)
- I Made Teja Sarmandana (2008561098)

1. Pendahuluan

Otak manusia merupakan kunci yang dapat digunakan untuk memahami pikiran, emosi, dan niat yang dimiliki seseorang. Saat ini telah dikembangkan berbagai alat yang mampu merekam sinyal yang dihasilkan oleh otak manusia. Electroencephalogram (EEG) merupakan perangkat khusus yang dapat digunakan untuk melakukan pengukuran potensi listrik dari otak dan sindrom elektrokimia spesifik dalam bentuk sinyal (Kusumaningrum, 2020). Sinyal EEG terdiri dari komponen-komponen frekuensi yang direpresentasikan dalam domain waktu (Anggara, 2020). Sinyal EEG memiliki bentuk yang kompleks serta nilai amplitudo yang rendah sehingga sulit untuk dilakukan pengamatan secara langsung (Yulia, 2019). Oleh karena itu, diperlukan penerapan metode pembelajaran mesin untuk mempermudah mengetahui pola serta mengidentifikasi keadaan dari otak manusia.

Salah satu bidang penerapan pembelajaran mesin terkait dengan pemrosesan sinyal EEG adalah *EEG Classification*. Klasifikasi EEG merupakan perpaduan antara ilmu saraf dan pembelajaran mesin, di mana suatu algoritma diterapkan untuk menyaring data yang rumit, mengekstraksi pola yang bermakna dan memasukkan ke dalam kategori tertentu. Dengan menerapkan pemrosesan komputer, proses identifikasi pola dapat dilakukan secara cepat dan akurat (Khosla, dkk., 2020). Pengukuran sinyal EEG yang dilakukan untuk melihat pola gelombang ketika seseorang berpikir dalam menyelesaikan tugas adalah contoh kasus dalam klasifikasi EEG. Pengukuran ini perlu dilakukan karena akan bermanfaat dalam membantu mengetahui kemampuan kognitif seseorang. Berangkat dari permasalahan tersebut, penulis tertarik mengembangkan sistem dan membangun model pembelajaran mesin menggunakan metode CNN untuk

melakukan klasifikasi sinyal EEG untuk menentukan apakah seseorang merupakan penghitung dalam hati yang baik atau buruk.

2. Metode Penelitian

2.1 Dataset

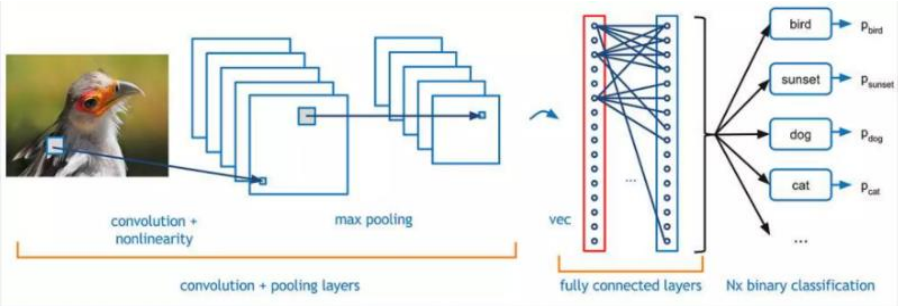
Dataset yang digunakan merupakan data sekunder yang diperoleh dari Kaggle berupa dataset rekaman EEG. Berdasarkan deskripsi pada dataset, file raw dari dataset tersebut bersumber dari physionet.org (Zyma I, 2019). Berikut merupakan link dataset yang digunakan:

https://www.kaggle.com/datasets/amananandrai/complete-eeg-dataset

Dataset ini terdiri dari 36 file berformat .csv yang merupakan hasil perekaman 36 subjek yang berbeda. Pada masing-masing file, terdapat 19 kolom data yang merupakan channels rekaman yang digunakan, meliputi Fp1, Fp2, F3, F4, F7, F8, T3,T4, C3, C4, T5, T6, P3, P4, O1, O2, Fz, Cz, dan Pz. Terdapat 2 label kelas yang digunakan pada dataset ini, yaitu seorang penghitung dalam hati yang baik (Good Counter) atau penghitung dalam hati yang buruk (Bad Counter).

2.2 Convolutional Neural Network (CNN)

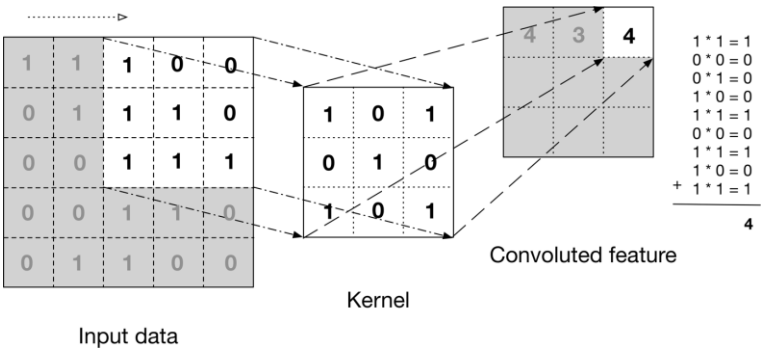
Convolutional Neural Network (CNN) telah digunakan beberapa tahun terakhir dan menjadi populer dalam bidang deep learning. CNN merupakan salah satu metode yang sering digunakan dalam hal image recognition, image classification, object detection, face recognition, dan lain sebagainya. Hal ini didukung karena kemampuan CNN yang baik untuk digunakan melakukan pemrosesan data berdimensi 2D seperti citra atau gambar. CNN tidak terlalu jauh berbeda dengan neural network biasanya. CNN terdiri dari neuron yang memiliki weight, bias dan activation function (Nugroho, 2020). Weight atau bobot digunakan untuk mengukur kekuatan koneksi antara dua neuron atau node dalam jaringan, bias adalah nilai tambahan yang ditambahkan ke hasil kalkulasi di setiap neuron, dan fungsi aktivasi menentukan apakah suatu neuron harus diaktifkan atau tidak. Perbedaan utama antara CNN dengan neural network biasa adalah adanya convolutional layer. CNN memiliki empat layer utama, yaitu convolutional layer, pooling layer, dan fully connected layer (Mao, 2020).



Gambar 1. Layer pada CNN

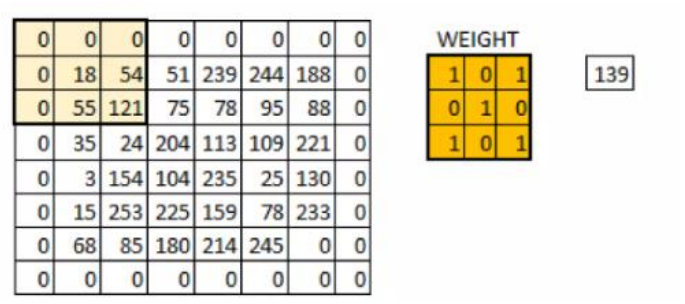
2.2.1 Convolutional Layer

CNN menggunakan convolutional layer untuk mengekstraksi fitur dari data input. Filter atau kernel kecil bergerak di seluruh gambar dan melakukan operasi konvolusi untuk mendeteksi pola atau fitur khusus. Proses konvolusi memungkinkan model untuk belajar fitur-fitur lokal yang dapat digunakan untuk pengenalan pola yang lebih kompleks.



Gambar 2. Convolutional Layer

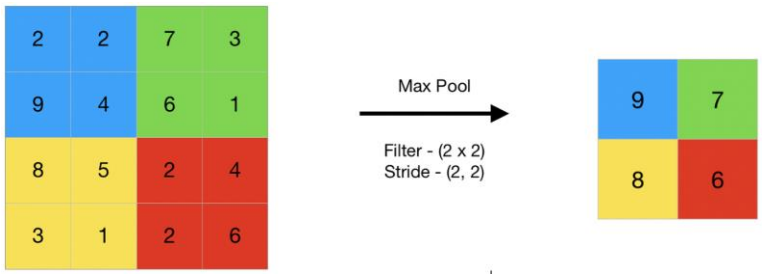
Pada contoh diatas input data dengan dimensi 5x5 setelah melewati convolutional layer mengalami perubahan dimensi menjadi 3x3, hal ini menunjukkan bahwa operasi konvolusi dapat mengurangi dimensi dari input. Pengurangan dimensi ini dapat menjadi suatu masalah apabila data input memiliki dimensi yang kecil. Padding merupakan cara yang dapat diterapkan untuk menyelesaikan permasalahan tersebut. Penambahan padding akan dilakukan terhadap data input untuk meningkatkan dimensi data dan menjaga dimensi output tetap sama.



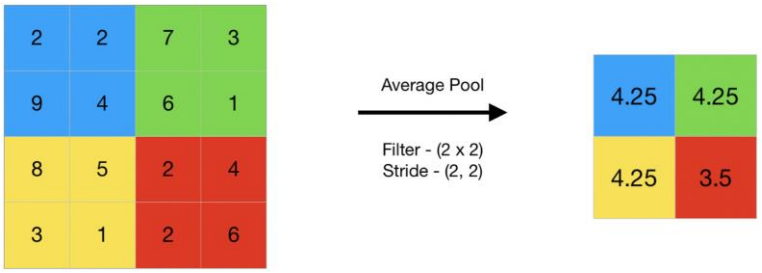
Gambar 3. Padding

2.2.2 Pooling Layer

Setelah melalui layer convolutional, data diteruskan ke pooling layer. Pooling layer digunakan untuk mengurangi dimensi peta fitur dan meningkatkan ketahanan ekstraksi fitur (Mao, 2020). Pooling dapat dilakukan dengan metode seperti max pooling (memilih nilai maksimum dalam suatu area) atau average pooling (mengambil rata-rata nilai).



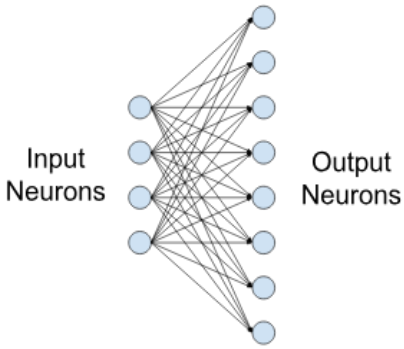
Gambar 4. Max Pooling



Gambar 5. Average Pooling

2.2.3 Fully Connected Layer

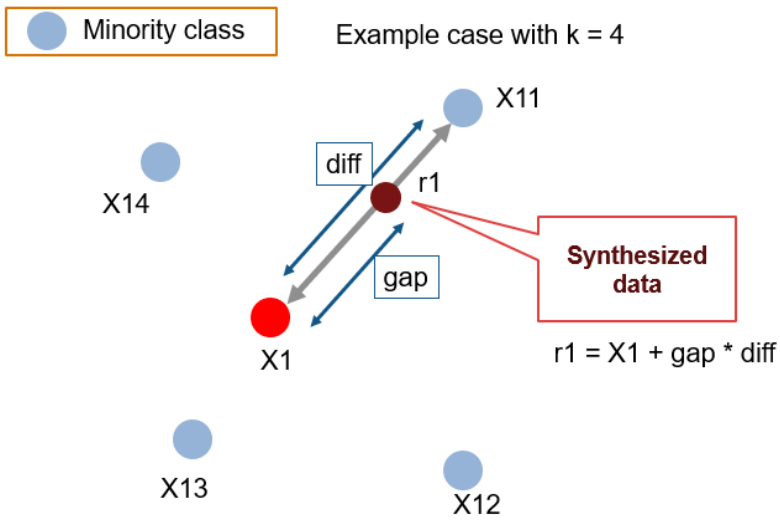
Setelah beberapa convolutional dan pooling layer, hasilnya diaplikasikan ke fully connected layer untuk klasifikasi. Layer ini bekerja seperti lapisan-lapisan dalam neural network biasa, dimana setiap neuron terhubung dengan setiap neuron di layer sebelumnya.



Gambar 6. Fully Connected Layer

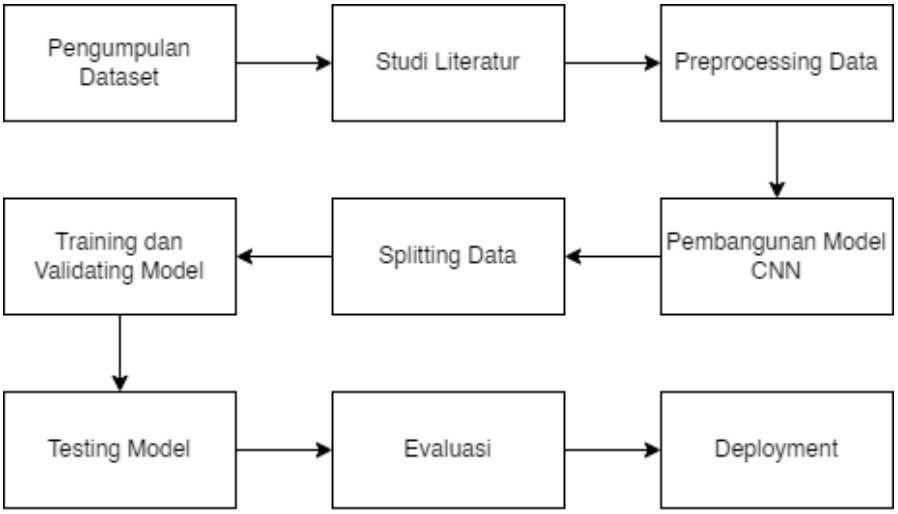
2.3 Synthetic Minority Oversampling Technique (SMOTE)

Dalam kasus data yang tidak seimbang (imbalance) salah satu teknik yang dapat dilakukan untuk mengatasi permasalahan tersebut adalah melakukan sampling. SMOTE dirancang khusus untuk mengatasi kumpulan data yang tidak seimbang dengan menghasilkan sampel sintetis untuk kelas minoritas (Satpathy, 2023). Dalam SMOTE sampel ditentukan menggunakan konsep KNN. Terlebih dahulu setiap node ditentukan K node dengan jarak terdekat. Kemudian data baru akan ditempatkan diantara node yang berdekatan tersebut dengan cara mengalikan jarak dengan nilai acak apa pun dalam (0,1] dan ditambahkan ke vektor fitur sebelumnya.



Gambar 7. SMOTE

2.4 Tahapan Penelitian

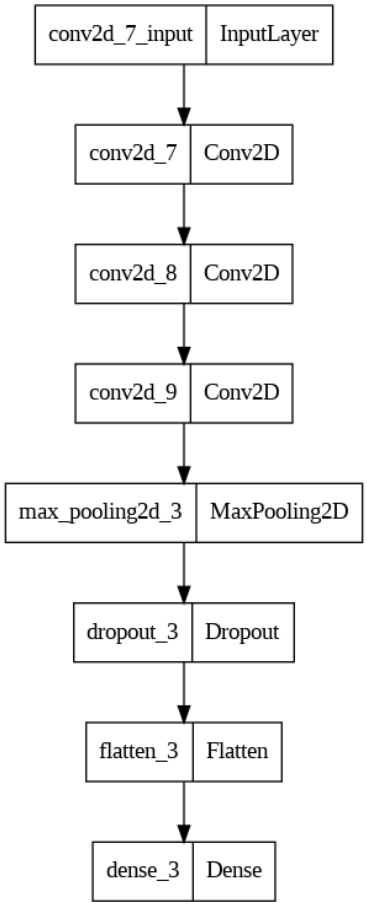


Gambar 8. Tahapan Penelitian

Tahapan penelitian yang dilakukan adalah sebagai berikut:

1. Mengumpulkan dataset
2. Melakukan eksplorasi dataset dan menambah pemahaman mengenai *subject matter*, yaitu pemrosesan sinyal EEG.
3. Melakukan preprocessing data dan visualisasi data menjadi gambar dengan mengambil sampel dari channel-channel yang ada dalam dataset serta melakukan sampling.
4. Membangun model klasifikasi dengan arsitektur CNN.
5. Membagi dataset menjadi data training, validasi, dan testing.
6. Melakukan pelatihan model dengan menggunakan data training dan data validasi. Pada proses ini dilakukan hyperparameter tuning seperti mengubah jumlah layer dalam arsitektur model, mengubah jumlah epoch, mengubah jumlah batch_size, dan mengubah jumlah node pada masing-masing layer.
7. Melakukan evaluasi hasil dengan menggunakan data testing pada masing-masing hasil hyperparameter tuning.
8. Memilih model terbaik berdasarkan hasil evaluasi.
9. Melakukan deployment model menggunakan Streamlit.

2.5 Arsitektur CNN



Gambar 9. Arsitektur Model yang Digunakan

Layer pertama yang merupakan input layer berupa layer Convolutional2d. Layer ini merupakan layer konvolusi dua dimensi yang berfungsi untuk mengekstraksi fitur dari gambar. Di sini terdapat 40 filter konvolusi. Layer selanjutnya adalah layer Convolutional2d yang kedua. Di sini digunakan 20 buah filter ukuran (1, 1). Layer selanjutnya masih sama, yaitu layer Convolutional2d yang ketiga. Di sini digunakan 10 buah filter ukuran (1, 1). Layer selanjutnya adalah MaxPooling2D yang digunakan untuk mereduksi dimensi spasial dari data gambar. Max pooling mengambil nilai maksimum dari sekelompok nilai di setiap wilayah yang nantinya akan menyebabkan reduksi ukuran citra dan akan mengonsentrasikan informasi fitur yang paling penting. Selanjutnya ditambahkan layer Dropout sebagai teknik regularisasi yang membantu mencegah overfitting dengan membuang beberapa neuron selama proses pelatihan. Lalu dilakukan Flatten untuk mengubah data dari format matriks ke vektor agar bisa diproses oleh lapisan Dense berikutnya. Layer selanjutnya adalah layer Dense yang merupakan

fully connected layer yang digunakan untuk menemukan hubungan yang lebih kompleks di dalam data. Layer terakhir adalah layer Dense yang juga sebagai output layer dengan menggunakan fungsi aktivasi sigmoid karena 2 kelas label yang ada.

3. Analisa dan Evaluasi Hasil

3.1 Pengumpulan Data

Data dikumpulkan oleh penulis melalui platform kaggle. Setelah data dikumpulkan, proses pengembangan model dimulai dari membuat kode untuk membaca data yang digunakan. Pada bagian ini, digunakan library Pandas dengan metode read_csv() yang akan membaca file csv dan menyimpannya ke dalam sebuah pandas dataframe.

```
s00 = pd.read_csv('./input/s00.csv',header=None)
s01 = pd.read_csv('./input/s01.csv',header=None)
s02 = pd.read_csv('./input/s02.csv',header=None)
s03 = pd.read_csv('./input/s03.csv',header=None)
s04 = pd.read_csv('./input/s04.csv',header=None)
s05 = pd.read_csv('./input/s05.csv',header=None)
s06 = pd.read_csv('./input/s06.csv',header=None)
s07 = pd.read_csv('./input/s07.csv',header=None)
s08 = pd.read_csv('./input/s08.csv',header=None)
s09 = pd.read_csv('./input/s09.csv',header=None)
s10 = pd.read_csv('./input/s10.csv',header=None)
s11 = pd.read_csv('./input/s11.csv',header=None)
s12 = pd.read_csv('./input/s12.csv',header=None)
s13 = pd.read_csv('./input/s13.csv',header=None)
s14 = pd.read_csv('./input/s14.csv',header=None)
s15 = pd.read_csv('./input/s15.csv',header=None)
s16 = pd.read_csv('./input/s16.csv',header=None)
s17 = pd.read_csv('./input/s17.csv',header=None)
s18 = pd.read_csv('./input/s18.csv',header=None)
s19 = pd.read_csv('./input/s19.csv',header=None)
s20 = pd.read_csv('./input/s20.csv',header=None)
s21 = pd.read_csv('./input/s21.csv',header=None)
s22 = pd.read_csv('./input/s22.csv',header=None)
s23 = pd.read_csv('./input/s23.csv',header=None)
s24 = pd.read_csv('./input/s24.csv',header=None)
```

```
s25 = pd.read_csv('./input/s25.csv',header=None)
s26 = pd.read_csv('./input/s26.csv',header=None)
s27 = pd.read_csv('./input/s27.csv',header=None)
s28 = pd.read_csv('./input/s28.csv',header=None)
s29 = pd.read_csv('./input/s29.csv',header=None)
s30 = pd.read_csv('./input/s30.csv',header=None)
s31 = pd.read_csv('./input/s31.csv',header=None)
s32 = pd.read_csv('./input/s32.csv',header=None)
s33 = pd.read_csv('./input/s33.csv',header=None)
s34 = pd.read_csv('./input/s34.csv',header=None)
s35 = pd.read_csv('./input/s35.csv',header=None)
```

3.2 Preprocessing Data

Setelah membaca file dataset berhasil dilakukan, selanjutnya adalah proses untuk mengubah format data dengan melakukan reshape. Reshape yang dilakukan akan mengubah format data yang awalnya 19 kolom dan 31.000 baris menjadi matriks berukuran 760x775.

```
s00=s00.transpose().to_numpy()
s00=s00.reshape(760,775)
s01=s01.transpose().to_numpy()
s01=s01.reshape(760,775)
s02=s02.transpose().to_numpy()
s02=s02.reshape(760,775)
s03=s03.transpose().to_numpy()
s03=s03.reshape(760,775)
s04=s04.transpose().to_numpy()
s04=s04.reshape(760,775)
s05=s05.transpose().to_numpy()
s05=s05.reshape(760,775)
s06=s06.transpose().to_numpy()
s06=s06.reshape(760,775)
s07=s07.transpose().to_numpy()
s07=s07.reshape(760,775)
s08=s08.transpose().to_numpy()
s08=s08.reshape(760,775)
s09=s09.transpose().to_numpy()
s09=s09.reshape(760,775)
s10=s10.transpose().to_numpy()
s10=s10.reshape(760,775)
```

```
s11=s11.transpose().to_numpy()
s11=s11.reshape(760,775)
s12=s12.transpose().to_numpy()
s12=s12.reshape(760,775)
s13=s13.transpose().to_numpy()
s13=s13.reshape(760,775)
s14=s14.transpose().to_numpy()
s14=s14.reshape(760,775)
s15=s15.transpose().to_numpy()
s15=s15.reshape(760,775)
s16=s16.transpose().to_numpy()
s16=s16.reshape(760,775)
s17=s17.transpose().to_numpy()
s17=s17.reshape(760,775)
s18=s18.transpose().to_numpy()
s18=s18.reshape(760,775)
s19=s19.transpose().to_numpy()
s19=s19.reshape(760,775)
s20=s20.transpose().to_numpy()
s20=s20.reshape(760,775)
s21=s21.transpose().to_numpy()
s21=s21.reshape(760,775)
s22=s22.transpose().to_numpy()
s22=s22.reshape(760,775)
s23=s23.transpose().to_numpy()
s23=s23.reshape(760,775)
s24=s24.transpose().to_numpy()
s24=s24.reshape(760,775)
s25=s25.transpose().to_numpy()
s25=s25.reshape(760,775)
s26=s26.transpose().to_numpy()
s26=s26.reshape(760,775)
s27=s27.transpose().to_numpy()
s27=s27.reshape(760,775)
s28=s28.transpose().to_numpy()
s28=s28.reshape(760,775)
s29=s29.transpose().to_numpy()
s29=s29.reshape(760,775)
s30=s30.transpose().to_numpy()
s30=s30.reshape(760,775)
s31=s31.transpose().to_numpy()
```

```
s31=s31.reshape(760,775)
s32=s32.transpose().to_numpy()
s32=s32.reshape(760,775)
s33=s33.transpose().to_numpy()
s33=s33.reshape(760,775)
s34=s34.transpose().to_numpy()
s34=s34.reshape(760,775)
s35=s35.transpose().to_numpy()
s35=s35.reshape(760,775)
```

Setelah dilakukan reshape, proses selanjutnya adalah mengumpulkan semua data menjadi 1 agar memudahkan pengaksesan datanya.

```
dataset= np.array([[s00],[s01],[s02],[s03],[s04],[s05],[s06],[s07],
                   [s08],[s09],[s10],[s11],[s12],[s13],[s14],[s15],[s16],[s17],[s18],[s19],
                   [s20],[s21],[s22],[s23],[s24],[s25],[s26],[s27],
                   [s28],[s29],[s30],[s31],[s32],[s33],[s34],[s35]])
data.shape
```

Output:

(36, 1, 760, 775)

Proses berikutnya adalah membuat arsitektur model seperti yang sudah dijelaskan di bagian sebelumnya. Setelah arsitektur model dibuat, langkah selanjutnya adalah mendefinisikan label kelas untuk masing-masing subjek data. Ada 2 kelas label, yaitu 1 yang menandakan subjek tergolong orang yang Good Counter, dan 0 yang menandakan subjek tergolong orang yang Bad Counter.

```
y=np.array([0,1,1,1,0,1,0,1,1,0,0,1,1,1,0,1,1,1,1,0,1,0,0,1,1,1,
1,1,1,1,0,1,1,1,1,1])
```

Proses selanjutnya adalah melakukan oversampling untuk menambah jumlah data. Teknik oversampling yang digunakan adalah dengan menggunakan Synthetic Minority Oversampling Technique (SMOTE). Dari proses ini, berhasil membuat jumlah data menjadi 52 data.

```
original = dataset.shape
print(original)
```

Output:

(36, 589000)

```
dataset= np.reshape(dataset, (36,-1 ))
dataset.shape
```

Output :

(36, 589000)

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(dataset, y)
X_res=X_res.reshape(52,1,760,775)
y_res
```

Output:

array([0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0])

3.3 Membangun Model CNN

Model CNN dibangun menggunakan library tensorflow keras. Adapun implementasi dari pendefinisian model adalah sebagai berikut.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(40, (1, 1), activation = 'relu', input_shape = (1,760,775), padding='Valid',data_format='
    tf.keras.layers.Conv2D(20, (1, 1), activation = 'relu', padding='Valid',data_format='channels_first'),
    tf.keras.layers.Conv2D(10, (1, 1), activation = 'relu', padding='Valid',data_format='channels_first'),
    tf.keras.layers.MaxPool2D(2, 2),      #max pool untuk mendapat nilai max dari masing-masing filter
    tf.keras.layers.Dropout(0.2),         #dropout untuk mencegah overfit
    tf.keras.layers.Flatten(),            #flatten datanya agar bisa dimasukkan ke layer dense
    tf.keras.layers.Dense(1, activation = 'sigmoid') #menggunakan aktivasi sigmoid karena binary class classificatio
])

model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['acc'])
```

Gambar 10. Pendefinisian Model CNN

Arsitektur model yang digunakan dapat dilihat pada Gambar 7. Dengan rincian masing-masing layernya dapat dilihat pada summary model di bawah ini setelah diimplementasikan.

Tabel 1. Model Summary

Layer (type)	Output Shape	Param
conv2d_4 (Conv2D)	(None, 40, 760, 775)	80
conv2d_5 (Conv2D)	(None, 20, 760, 775)	820
conv2d_6 (Conv2D)	(None, 10, 760, 775)	210

```
max_pooling2d_2 (MaxPoolin  (None, 5, 380, 775)       0
g2D)
dropout_2 (Dropout)         (None, 5, 380, 775)       0
flatten_2 (Flatten)         (None, 1472500)           0
dense_2 (Dense)             (None, 1)           1472501
=================================================================
```

3.4 *Splitting Data*

Langkah selanjutnya adalah memecah data menjadi data training dan data testing. Data yang digunakan sebanyak 52 subjek yang merupakan hasil oversampling dari data awal sebanyak 36 subjek. Rasio dari data training dan testing adalah 75:25. Dari 75% data training ini, akan dipecah lagi menjadi data validasi dengan rasio 80:20. Hal ini membuat terdapat 39 data yang akan digunakan dalam proses training dengan 8 digunakan sebagai data validasi saat proses training. Dan data testing sejumlah 13 data yang merupakan data yang belum pernah dilihat oleh model karena tidak digunakan sebagai data training maupun data validasi selama proses training.

```
x_train, x_test, y_train, y_test = train_test_split(X_res, y_res,
test_size=0.25, random_state=12,stratify=y_res)
```

```
y_train
```

Output：Training set terdiri dari 39 data.

array([0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1,
1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1])

```
y_test
```

Output：Training set terdiri dari 13 data.

array([1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1])

3.5 *Training* dan *Validating* Model

Langkah selanjutnya adalah melakukan proses training dengan data training dan validasi, epoch sebesar 30 dan batch_size 1.

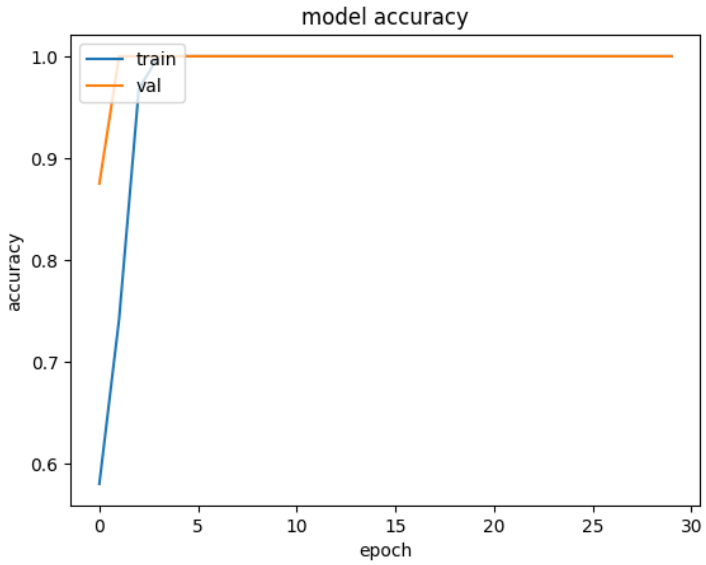
```
history =
model.fit(x_train,y_train,batch_size=1,epochs=30,validation_spli
t=0.20)
```

Output:

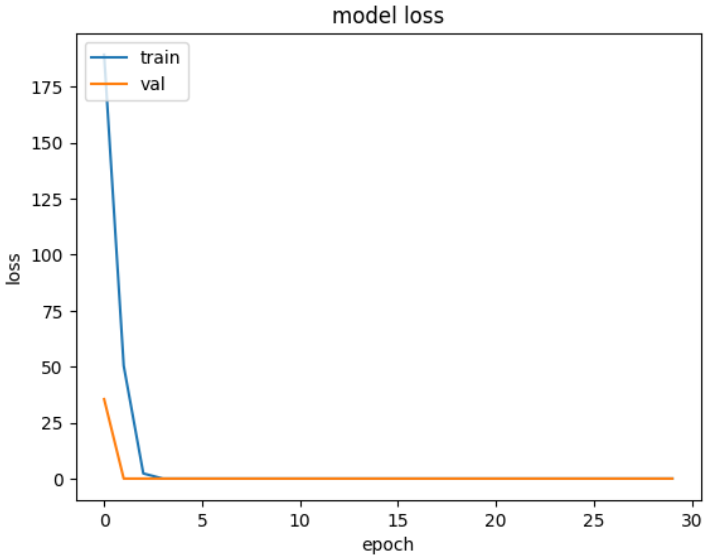
```
history = model.fit(x_train,y_train,batch_size=1,epochs=30,validation_split=0.20)
Epoch 2/30
31/31 [==============================] - 1s 18ms/step - loss: 15.9314 - acc: 0.8065 - val_loss: 1.9758e-10 - val_acc: 1.0000
Epoch 3/30
31/31 [==============================] - 1s 18ms/step - loss: 2.1826 - acc: 0.9677 - val_loss: 15.8673 - val_acc: 0.6250
Epoch 4/30
31/31 [==============================] - 1s 18ms/step - loss: 3.2975 - acc: 0.9677 - val_loss: 7.8991e-21 - val_acc: 1.0000
Epoch 5/30
31/31 [==============================] - 1s 18ms/step - loss: 4.8801e-04 - acc: 1.0000 - val_loss: 3.2615e-25 - val_acc: 1.0000
Epoch 6/30
31/31 [==============================] - 1s 18ms/step - loss: 5.7789e-04 - acc: 1.0000 - val_loss: 5.5176e-25 - val_acc: 1.0000
Epoch 7/30
31/31 [==============================] - 1s 18ms/step - loss: 7.2627e-05 - acc: 1.0000 - val_loss: 1.6463e-24 - val_acc: 1.0000
Epoch 8/30
31/31 [==============================] - 1s 18ms/step - loss: 2.8566e-05 - acc: 1.0000 - val_loss: 1.8248e-24 - val_acc: 1.0000
Epoch 9/30
31/31 [==============================] - 1s 22ms/step - loss: 1.4239e-05 - acc: 1.0000 - val_loss: 1.9822e-24 - val_acc: 1.0000
Epoch 10/30
31/31 [==============================] - 1s 19ms/step - loss: 1.8615e-05 - acc: 1.0000 - val_loss: 2.1821e-24 - val_acc: 1.0000
Epoch 11/30
31/31 [==============================] - 1s 20ms/step - loss: 2.0378e-05 - acc: 1.0000 - val_loss: 2.2389e-24 - val_acc: 1.0000
Epoch 12/30
31/31 [==============================] - 1s 19ms/step - loss: 8.5872e-06 - acc: 1.0000 - val_loss: 2.4602e-24 - val_acc: 1.0000
Epoch 13/30
31/31 [==============================] - 1s 23ms/step - loss: 8.9681e-06 - acc: 1.0000 - val_loss: 2.5633e-24 - val_acc: 1.0000
Epoch 14/30
31/31 [==============================] - 1s 23ms/step - loss: 1.1792e-05 - acc: 1.0000 - val_loss: 2.6823e-24 - val_acc: 1.0000
Epoch 15/30
31/31 [==============================] - 1s 23ms/step - loss: 2.4013e-05 - acc: 1.0000 - val_loss: 2.9596e-24 - val_acc: 1.0000
Epoch 16/30
31/31 [==============================] - 1s 19ms/step - loss: 2.0171e-05 - acc: 1.0000 - val_loss: 3.1717e-24 - val_acc: 1.0000
```

Gambar 11. Proses Training Model

Model dilatih dengan jumlah epochs = 30 dan batch_size = 1. Setelah dilatih, model mencapai validation accuracy 100% hanya dalam 4 epochs. Apabila dilihat dari loss saat training, terlihat bahwa training loss dan validation lossnya hampir mencapai 0 yang berarti sangat bagus.



Gambar 12. Grafik Model Accuracy saat Training



Gambar 13. Grafik Model Loss saat Training

3.6 *Testing* dan Evaluasi

Saat dilakukan evaluasi pada data testing yang sudah dipisahkan dari awal, performa model memiliki akurasi, recall, dan precision yang lumayan tinggi, yaitu 92.31%, 91.67%, dan 87.50%. Hal ini menandakan model memiliki kemampuan yang baik dan tidak overfit saat memprediksi data baru yang belum pernah dilihat sebelumnya.

```
loss,acc,rec,auc,pre=model.evaluate(x_test, y_test)
```

Output:

1/1 [==============================] - 0s 370ms/step - loss: 3.4639 - acc: 0.9231 - recall_2: 1.0000 - auc_2: 0.9167 - precision_2: 0.8750

Setelah model selesai dibangun, model kemudian disimpan untuk dipergunakan dalam proses deployment.

```
model.save('CNN_eeg_model.h5')
```

3.7 Deployment

Setelah model tersimpan, maka langkah selanjutnya adalah men-deploy model ke aplikasi web. Penulis menggunakan Streamlit untuk men-deploy model ini. Proses pengembangan aplikasi web ini dimulai dengan melakukan import beberapa library yang dibutuhkan.

```
import streamlit as st
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Setelah library di-load, dilanjutkan dengan melakukan load model yang sebelumnya telah disimpan.

```
model = keras.models.load_model("CNN_eeg_model.h5")
model.compile(loss='binary_crossentropy',
optimizer='adam',metrics=['acc',tf.keras.metrics.Recall(),tf.ker
as.metrics.AUC(),tf.keras.metrics.Precision()])
```

Langkah selanjutnya adalah menambahkan beberapa interface seperti judul aplikasi, deskripsi, dan input yang dibutuhkan.

```
st.sidebar.subheader('About the App')
st.sidebar.write('EEG Classification App with Streamlit using a
trained CNN model')
st.sidebar.write('This app will classify EEG signal and determine
whether the subject is a Good counter or Bad counter (for whom the
mental task required excessive efforts).')

#start the user interface
st.title("EEG Classification App")
st.write("Upload csv file containing EEG signal with 19 channel
[Fp1, Fp2, F3, F4, F7, F8, T3,T4, C3, C4, T5, T6, P3, P4, O1, O2,
Fz, Cz, Pz]")
st.write("Input                                                 example
[s00.csv](https://drive.google.com/file/d/1wrWdREzw4z6rSK0kO3zkc
YuHqjCz21Tp/view?usp=sharing)")

uploaded_file = st.file_uploader("Upload file with format .csv",
type=”csv”)
```

Untuk mempermudah pengguna dalam memahami data, penulis menambahkan modul visualisasi untuk data EEG yang telah diinputkan

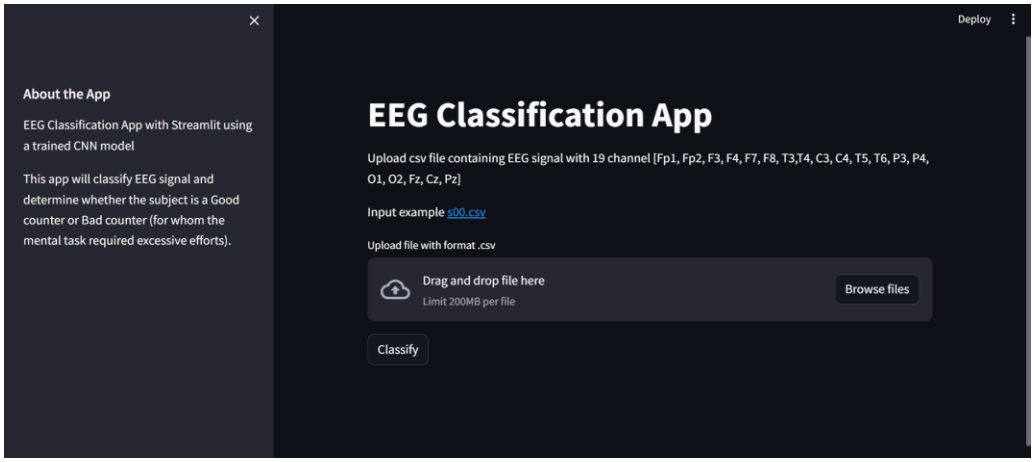
pengguna. Pada langkah ini, data yang unggah juga dilakukan penyesuaian dimensi untuk proses klasifikasi nantinya.

```
if uploaded_file is not None:
    # read csv
    signal = pd.read_csv(uploaded_file, header=None)
    signal = signal.transpose().to_numpy()
    signal = signal.reshape(760,775)

    channel_index = st.selectbox("Select Channel", options=[i+1
for i in range(signal.shape[0])])

    # visualize signal
    plt.plot(signal[channel_index-1])

    plt.title(f'EEG Signal - Channel {channel_index}')
    plt.xlabel('Time')
    plt.ylabel('EEG Signal Amplitude')
    st.pyplot(plt.gcf())
```

Setelah pengguna mengunggah file signal yang ingin diklasifikasi, pengguna diminta untuk menekan tombol classify untuk memulai proses. Adapun proses yang dijalankan ketika pengguna menekan tombol adalah model akan melakukan proses prediksi dan hasil prediksi dari model ditampilkan ke pengguna.

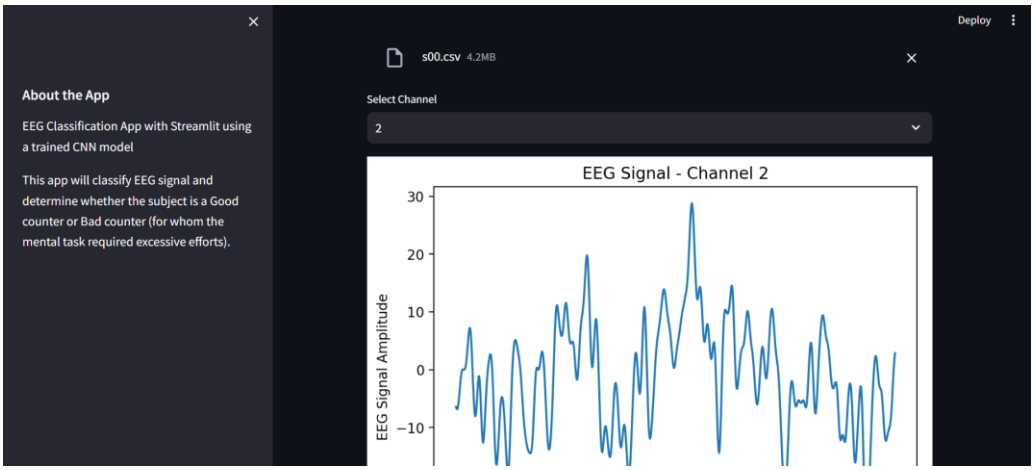
```
if st.button('Classify', key='classify_button'):
    if uploaded_file is not None:
        x_test = np.array([[signal]])

        predict = model.predict(x_test)

        if predict[0] > 0:
        # the predicted class is 1
            st.write(f"The subject is a Good Counter")
        else:
        # otherwise the predicted class is 0
            st.write(f"The subject is a Bad Counter")
    else:
        st.warning("you need to upload a csv file.")
```

Berikut merupakan hasil tangkapan layar dari aplikasi web yang dibangun dan digunakan untuk melakukan proses klasifikasi.



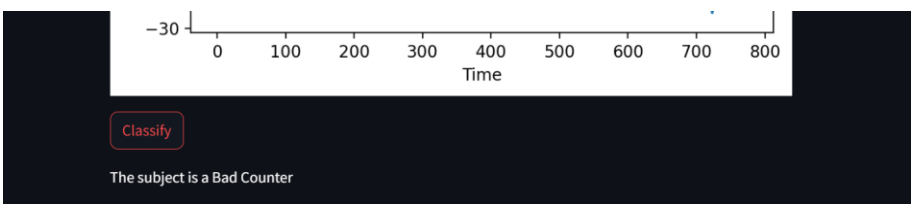
Gambar 14. Tampilan Halaman Awal

Apabila pengguna mengunggah file csv yang berisikan sinyal EEG maka visualisasi grafik akan ditampilkan. Pengguna dapat berpindah-pindah diantara channel-channel yang ada.



Gambar 15. Tampilan Ketika Data Sinyal di-Input

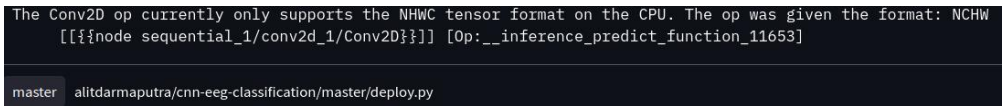
Setelah data diunggah, pengguna dapat melakukan proses klasifikasi dengan menekan tombol classify.



Gambar 16. Tampilan Hasil Klasifikasi

Pada contoh diatas, data sinyal EEG yang diunggah oleh pengguna menunjukkan bahwa subject merupakan seorang penghitung dalam hati yang buruk. Karena sistem yang dikembangkan telah berjalan dengan baik, maka proses pengembangan model hingga penerapannya dalam sistem telah selesai dilakukan.

Penulis telah mencoba untuk melakukan deploy sistem ke platform streamlit untuk dapat diakses melalui internet pada link berikut https://cnn-eeg-classification-cvt97jfsqylrjgysntadrj.streamlit.app/. Namun, platform streamlit tidak memberikan dukungan berupa GPU *runtime* yang dibutuhkan oleh model 2D CNN. Sehingga, sistem terbatas hanya dapat dijalankan pada lingkungan lokal yang mendukung adanya GPU *runtime*.



Gambar 17. Error Saat Deploy Model Karena Memerlukan GPU

3.8 Link Github

Adapun kode dari pembuatan model CNN dan pengembangan sistem dapat diakses pada repositori berikut:
https://github.com/alitdarmaputra/CNN-EEG-Classification/

3.9 Link Youtube

Video penjelasan terkait tahapan-tahapan dalam penelitian serta hasil dari model yang dibangun dapat dilihat pada link youtube berikut:

https://youtu.be/-flTOeOAaMA

4. Kesimpulan

Dari penelitian yang telah dilakukan, dapat disimpulkan bahwa penerapan metode Convolutional Neural Network (CNN) dapat digunakan untuk melakukan proses klasifikasi sinyal EEG dalam kasus menentukan apakah seseorang merupakan penghitung dalam hati yang baik atau buruk. Adapun hasil evaluasi dari model yang dibangun memiliki nilai akurasi sebesar 92.31%, recall 91.67%, dan presisi 87.50%.

Model yang dibangun kemudian diterapkan dalam aplikasi web yang mampu melakukan proses klasifikasi sinyal EEG dan telah berjalan dengan baik.

5. Daftar Pustaka

Anggara, R., & Rahayu, Y. (2020). Sistem Electroencephalogram (EEG) Untuk Analisis Sinyal Gelombang Otak Pada Pasien Depresi. Jurnal Online Mahasiswa (JOM) Bidang Teknik dan Sains, 7, 1-6.

Khosla, A., Khandnor, P., & Chand, T. (2020). A comparative analysis of signal processing and classification methods for different applications based on EEG signals. Biocybernetics and Biomedical Engineering, 40(2), 649-690.

Kusumaningrum, D., & Imah, E. M. (2020). Studi Komparasi Algoritma Klasifikasi Mental Workload Berdasarkan Sinyal EEG. Jurnal Sistem Cerdas, 3(2), 133-143.

Mao, W. L., Fathurrahman, H. I. K., Lee, Y., & Chang, T. W. (2020). EEG dataset classification using CNN method. In Journal of physics: conference series (Vol. 1456, No. 1, p. 012017). IOP Publishing.

Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (Cnn) Pada Ekspresi Manusia. Algor, 2(1), 12-20.

Satpathy, Swastik. (2023). SMOTE for Imbalanced Classification with Python. Analyticsvidhya. Diakses melalui https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/ pada 29 Desember 2023.

Yulia, M., Anita, A., & Miranda, C. (2019). Classification of EEG Signals with Aromatic Stimuli Using the Support Vector Machine Method. Jurnal Ilmu Komputer dan Bisnis, 10(1), 2156-2166.

Zyma I, Tukaev S, Seleznov I, Kiyono K, Popov A, Chernykh M, Shpenkov O. Electroencephalograms during Mental Arithmetic Task Performance. Data. 2019; 4(1):14. https://doi.org/10.3390/data4010014