(2026), 4 (2): 300-321 **Scientica** 3021-8209

Jurnal Ilmiah Sain dan Teknologi

PENERAPAN CNN UNTUK KLASIFIKASI TINGKAT TUTUR TEKS BAHASA JAWA

Dina Kamelia [1], Herry Sujaini [2], Rina Septiriana [3]
Fakultas Teknik
Universitas Tanjungpura
Email : dinakamelia1973@student.untan.ac.id, hs@untan.ac.id, rinaseptiriana@informatika.untan.ac.id

Abstract
This study applies a Convolutional Neural Network (CNN) model to classify speech levels in Javanese texts, namely ngoko, madya, and krama. The objective is to investigate the application of the CNN model in classifying Javanese text speech levels and to evaluate its performance using accuracy, precision, recall, and F1-score metrics. The dataset consists of 1200 labeled texts, processed through preprocessing stages (cleaning, case folding, tokenization, padding, and label encoding) and evaluated across three experimental scenarios: Scenario 1 (original data), Scenario 2 (data with Synthetic Minority Over-sampling Technique/SMOTE), and Scenario 3 (combined original and synthetic madya data). The CNN model, with a simple architecture (embedding, convolution, pooling, and dense layers), Adam optimizer, and 100 epochs of training, was evaluated using 10-fold cross-validation. The SMOTE oversampling technique was employed to address class imbalance in the dataset. Scenario 1 achieved an average accuracy, precision, recall, and F1-score of 0.96 but faced challenges in classifying the madya class due to data imbalance. Scenario 2 yielded an average metric of 0.92, with significant improvement in madya class classification. Scenario 3 achieved an average metric of 0.89, with reduced performance due to variations in the characteristics of additional madya data. In conclusion, the CNN model is effective for classifying Javanese text speech levels, with Scenario 2 providing the best balance between accuracy and minority class classification. This study contributes to the preservation of Javanese culture through structured digital text documentation, supporting the development of local language learning applications.

Article History
Submitted: 21 April 2026
Accepted: 24 April 2026
Published: 25 April 2026

Key Words
CNN, Text Classification, Javanese Language, SMOTE

Abstrak
Penelitian ini menerapkan model *Convolutional Neural Network* (CNN) untuk mengklasifikasikan tingkat tutur dalam teks bahasa Jawa, yaitu ngoko, madya, dan krama. Tujuan penelitian adalah untuk mengetahui penerapan model Convolutional Neural Network (CNN) dalam mengklasifikasikan tingkat tutur teks bahasa Jawa dan mengevaluasi performa model melalui metrik akurasi, presisi, recall, dan F1-score. Dengan menggunakan *dataset* berisi 1200 teks yang dilabeli sesuai tingkat tutur, yang diproses melalui tahap *preprocessing* (*cleaning*, *case folding*, tokenisasi, *padding*, *label encoding*) dan tiga skenario eksperimen: Skenario 1 (data asli), Skenario 2 (data dengan Synthetic Minority Over-sampling Technique/SMOTE), dan Skenario 3 (gabungan data asli dan data sintetis madya). Model CNN dengan arsitektur sederhana (embedding, konvolusi, pooling, dan dense layer), optimizer Adam, dan pelatihan selama 100 epochs dievaluasi menggunakan 10-fold cross-validation. Selain itu, teknik *oversampling* SMOTE digunakan untuk mengatasi ketidakseimbangan kelas dalam dataset. Tiga skenario eksperimen diuji: data asli, data dengan SMOTE, dan gabungan tambahan data sintetis SMOTE dengan data asli. Evaluasi dilakukan dengan *10-fold cross-validation*, menghasilkan rata-rata akurasi, presisi, recall, dan F1-score sebesar 0,96 pada skenario 1 (data asli), namun kesulitan mengklasifikasikan kelas madya karena ketidakseimbangan data. Skenario 2 menghasilkan rata-rata metrik 0,92 dengan peningkatan signifikan pada klasifikasi kelas madya. Skenario 3 mencapai rata-rata metrik 0,89, dengan penurunan performa akibat

Sejarah Artikel
Submitted: 21 April 2026
Accepted: 24 April 2026
Published: 25 April 2026

Kata Kunci
CNN, Klasifikasi Teks, Bahasa Jawa, SMOTE

variasi karakteristik data tambahan madya. Kesimpulan penelitian ini adalah model CNN efektif untuk mengklasifikasikan tingkat tutur teks bahasa Jawa

1. PENDAHULUAN

Dalam era perkembangan teknologi yang pesat saat ini, berbagai jenis data terus dihasilkan, terutama dalam bentuk teks. Data teks ini muncul dari berbagai sumber, seperti media sosial, situs web, dokumen digital, dan sumber lainnya, menciptakan kekayaan informasi yang perlu diolah. Data teks yang dihasilkan bukan hanya bahasa yang umum digunakan secara internasional, tetapi juga termasuk bahasa-bahasa daerah yang memiliki kekayaan budaya dan sejarah, seperti bahasa Jawa. Bahasa Jawa merupakan salah satu bahasa daerah di Indonesia yang memiliki nilai budaya yang tinggi dan digunakan oleh jutaan penutur. Secara tradisional, bahasa Jawa dikenal memiliki tiga tingkat tutur, yaitu ngoko, madya, dan krama, yang mencerminkan tingkatan kesopanan dalam interaksi sosial (Poedjosoedarmo et al., 2013).

Namun, seiring dengan perubahan zaman dan arus globalisasi, penggunaan bahasa Jawa, terutama di kalangan generasi muda, semakin berkurang (Wahyuni et al., 2018). Hal ini dapat berakibat pada penurunan pemahaman mengenai tingkat tutur yang tepat serta berpotensi mengancam kelestarian budaya dan bahasa tersebut. Oleh karena itu, perlunya penerapan solusi teknologi yang mampu membantu dalam pengenalan dan pemahaman perbedaan tutur bahasa Jawa agar tetap lestari di tengah arus modernisasi dan era digital ini.

Teknologi *deep learning*, khususnya *Convolutional Neural Network* (CNN), telah terbukti berhasil dalam pengklasifikasian data teks dalam berbagai penelitian. CNN memiliki kemampuan ekstraksi fitur yang tinggi, yang memungkinkan model ini untuk menangani data teks yang kompleks dan berstruktur (Listyarini & Anggoro, 2021). Penerapan CNN dalam klasifikasi tingkat tutur bahasa Jawa dapat menjadi solusi efektif dalam memetakan dan memahami perbedaan tingkat tutur dengan tepat. Dengan mengembangkan model yang dapat mengklasifikasikan teks bahasa Jawa berdasarkan tingkat tutur, diharapkan dapat memudahkan proses pembelajaran bahasa.

Selain itu, klasifikasi ini juga dapat mendukung upaya pelestarian budaya dengan memudahkan dokumentasi digital teks berbahasa Jawa secara terstruktur. Hasil klasifikasi tersebut selanjutnya dapat digunakan untuk berbagai keperluan, seperti pembelajaran interaktif, pengembangan aplikasi berbasis bahasa daerah, serta dokumentasi budaya yang dapat diakses oleh generasi mendatang. Dengan memanfaatkan teknologi deep learning, diharapkan hasil penelitian ini dapat berkontribusi pada pengembangan metode yang lebih akurat dan efisien dalam pengelolaan bahasa Jawa, serta meningkatkan kesadaran akan pentingnya pelestarian bahasa dan budaya lokal di era modern.

Penelitian ini menggunakan *dataset* berupa korpus teks bahasa Jawa yang disusun secara manual dengan merujuk pada buku 'Tingkat Tutur Bahasa Jawa' karya Poedjosoedarmo et al. (2013). *Dataset* ini dikelompokkan ke dalam tiga tingkat tutur: ngoko, madya, dan krama. Proses pengumpulan data dilakukan dengan menerjemahkan kalimat-kalimat berbahasa Indonesia yang memiliki sintaksis sederhana dan jelas makna ke dalam tiga tingkat tutur bahasa Jawa, sehingga menghasilkan representasi yang akurat dan sesuai dengan kaidah linguistik bahasa Jawa. Sumber data ini memberikan kontribusi penting dalam penelitian ini, karena menyediakan korpus teks yang terstruktur dan terlabeli dengan baik, yang memungkinkan model CNN untuk mempelajari pola-pola linguistik khas dari masing-masing tingkat tutur.

Berdasarkan uraian diatas, penelitian ini akan berfokus pada penerapan algoritma *Convolutional Neural Network* (CNN) beserta arsitekturnya dalam klasifikasi tingkat tutur teks bahasa jawa. Sistem yang akan dibangun menggunakan input berupa korpus bahasa Jawa, outputnya adalah hasil klasifikasi teks bahasa Jawa sesuai dengan jenis tingkat tuturnya.

Adapun jenis tingkat tutur yang akan diklasifikasi pada penelitian ini ada 3 jenis yaitu Ngoko, Madya, Krama.

2. METODE PENELITIAN

Alat dan Data Penelitian

Dalam sebuah penelitian, diperlukan alat untuk membantu pengerjaan penelitian serta data penelitian yang akan digunakan untuk mendukung hasil dari penelitian. Berikut adalah alat dan data yang akan digunakan pada penelitian ini.

Perangkat Keras

Perangkat keras yang digunakan berupa laptop dengan spesifikasi sebagai berikut.

a. *Processor* : *Intel(R) Core(TM) i5-5200U CPU @2.20GHz 2.19 GHz*,
b. *Memory* : 8GB RAM.

Perangkat Lunak

Perangkat lunak yang digunakan dalam penelitian ini yaitu.

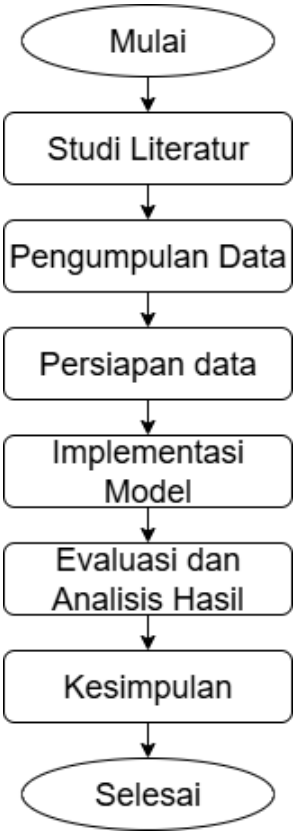
a. Sistem Operasi *Windows* 10 *Pro*
b. *Web Browser* (*Google Chrome*)
c. *Python*
d. *Google Colaboratory*

Data Penelitian

Data yang akan digunakan pada penelitian ini adalah kumpulan data berupa teks bahasa Jawa. Adapun jenis tingkat tutur yang akan diklasifikasi pada penelitian ini ada 3 (tiga) jenis tingkat tutur yaitu Ngoko, Madya, Krama.

Diagram Alir Penelitian

Diagram alir penelitian akan memberikan penjelasan mengenai tahapan-tahapan yang dilakukan dalam penelitian ini, dapat dilihat pada **Gambar 2.2**.



Gambar 2. 1 Diagram alir penelitian

3. Hasil dan pembahasan

Persiapan Data

Inisiasi Data

Sebelum menggunakan data untuk pelatihan model, *dataset* awal dari *file excel* (dataset bahasa jawa.xlsx) perlu diproses terlebih dahulu agar memiliki format dan distribusi kelas yang sesuai. *File* tersebut berisi tiga kolom, yaitu Ngoko, Madya, dan Krama, yang masing-masing merepresentasikan tingkat tutur dalam bahasa Jawa

.

Kode Program 3. 1 Tahap Inisiasi Data

```
# Inisiasi Data 1
file_path = '/content/drive/MyDrive/dataset bahasa jawa.xlsx'
# Check if the file exists before attempting to read
if not os.path.exists(file_path):
print(f"Error: File not found at {file_path}")
else:
df = pd.read_excel(file_path)
# Transformasi dataset
data = []
for index, row in df.iterrows():
        data.append({'kalimat': row['Ngoko'], 'kelas': 'ngoko'})
        data.append({'kalimat': row['Madya'], 'kelas': 'madya'})
        data.append({'kalimat': row['Krama'], 'kelas': 'krama'})
df_transformed = pd.DataFrame(data)
# Menyesuaikan jumlah data per kelas
madya_data1 = df_transformed[df_transformed['kelas'] == 'madya'].sample(n=200,
random_state=seed)
ngoko_data1 = df_transformed[df_transformed['kelas'] == 'ngoko']
krama_data1 = df_transformed[df_transformed['kelas'] == 'krama']
df_data1 = pd.concat([ngoko_data1, madya_data1, krama_data1],
ignore_index=True)
# Inisiasi Data 2
file_path_data2 = '/content/drive/MyDrive/madyajawa.xlsx'
if not os.path.exists(file_path_data2):
print(f"Error: File not found at {file_path_data2}")
# Handle the case where the file is not found, maybe skip data 2
data2_available = False
else:
data2_available = True
df_data2 = pd.read_excel(file_path_data2)
# Transformasi data 2 (sesuai format data 1)
data2 = []
for index, row in df.iterrows():
        data.append({'kalimat': row['Ngoko'], 'kelas': 'ngoko'})
        data.append({'kalimat': row['Madya'], 'kelas': 'madya'})
        data.append({'kalimat': row['Krama'], 'kelas': 'krama'})
df_data2 = pd.DataFrame(data)
# Menyesuaikan jumlah data per kelas
  madya_data2 = df_transformed[df_transformed['kelas'] == 'madya'].sample(n=50,
  random_state=seed)
  ngoko_data2 = df_transformed[df_transformed['kelas'] == 'ngoko']
```

```
krama_data2 = df_transformed[df_transformed['kelas'] == 'krama']
df_data2 = pd.concat([ngoko_data2, madya_data2, krama_data2],
ignore_index=True)
```

Untuk inisiasi data 1, file path ditentukan dan diperiksa keberadaannya dengan os.path.exists sebelum diproses menggunakan pd.read_excel. Data diubah menjadi list dengan loop yang mengambil pasangan "kalimat" dan "kelas" berdasarkan kolom Ngoko, Madya, dan Krama, lalu dikonversi ke dataframe bernama df_transformed. Sampel madya sebanyak 200 diambil dengan sample, sementara ngoko dan krama diambil semua, kemudian digabungkan menjadi df_data1 menggunakan pd.concat. Proses serupa dilakukan untuk data 2 dengan file "madyajawa.xlsx". Verifikasi dilakukan dengan mencetak nama kolom dan distribusi kelas menggunakan print.

Terakhir, kode mencetak nama kolom dataframe dan distribusi data per kelas berdasarkan kolom "*kelas*" untuk memverifikasi hasilnya. Proses ini bertujuan untuk menyeimbangkan dataset bahasa Jawa agar siap digunakan dalam analisis lebih lanjut. Adapun hasil *outputnya* adalah jumlah data per kelas yang dapat dilihat pada **Tabel 3.1** sebagai berikut.

Tabel 3. 1 Hasil Inisiasi Data 1

Kelas	Jumlah
Ngoko	500
Madya	200
Krama	500

Tabel 3. 2 Hasil Inisiasi Data 1

Kelas	Jumlah
Ngoko	500
Madya	50
Krama	500

Menampilkan Data

Setelah data berhasil dimuat maka selanjutnya adalah menampilkan data. Langkah ini dilakukan untuk memastikan bahwa proses transformasi data dari format *Excel* ke format dua kolom telah berhasil dan sesuai dengan yang diharapkan. Pada tahap ini, program mencetak judul *"Contoh data per kelas"* sebagai penanda awal, kemudian menampilkan beberapa contoh kalimat dari setiap kelas tingkat tutur dalam Bahasa Jawa, yaitu Ngoko, Madya, dan Krama. Output dari proses ini berupa cuplikan kalimat dari masing-masing kelas, seperti pada Gambar 3.3 berikut.

Kode Program 3. 2 Menampilkan Data

```
print("\nContoh data per kelas data 1:")
print("\nNgoko (5 contoh pertama):")
print(df_data1[df_data1['kelas'] == 'ngoko']['kalimat'].head().values)
print("\nMadya (5 contoh pertama):")
print(df_data1[df_data1['kelas'] == 'madya']['kalimat'].head().values)
print("\nKrama (5 contoh pertama):")
print(df_data1[df_data1['kelas'] == 'krama']['kalimat'].head().values)
```

```
Contoh data per kelas data 1:

Ngoko (5 contoh pertama):
['Awak dhewe njupuk pit rusak ing wayah sore.'
 'Kancaku nggawe roti soklat ing pawon.'
 'Kancaku ajar bal-balan ing taman cedhak griya.'
 'Aku tuku tas sekolah ing pasar.'
 'Wong-wong padha nonton filem ing biyoskup cerak kantor.']

Madya (5 contoh pertama):
[nan 'para tiyang sami dolanan layangan teng sawah.' nan
 'dinten niki, kula kesah piknik.' 'sampeyan saweg pados napa ?']

Krama (5 contoh pertama):
['Kula panjenengan sami mendhet sepedhah risak wanci sonten.'
 'rencang kawula ndamel roti soklat wonten ing pawon.'
 'Kanca kula ajar bal-balan wonten ing taman celak griya.'
 'Kula mundhut tas sekolah wonten ing peken.'
 'Para tiyang sami mriksani filem wonten ing biyoskup celak kantor.']
```

Gambar 3. 1 Tampilan Data Tiap Kelas

Dengan menampilkan data seperti ini dapat memperoleh gambaran awal mengenai isi dataset setelah diproses. Hal ini penting untuk memverifikasi bahwa setiap kelas telah terisi dengan benar dan seimbang, sehingga dataset siap digunakan untuk tahap selanjutnya seperti pelatihan model.

Cleaning Data* dan *Case Folding

Tahap *cleaning data* merupakan tahap membersihkan teks dalam kolom kalimat agar siap digunakan dalam proses pengolahan teks selanjutnya.

Kode Program 3. 3 Tahap *Cleaning* Data dan *Case Folding*

```
# Cleaning Data (Fungsi)
def clean_text(text):
text = re.sub(r'[^a-zA-Z\s]', '', str(text))
# Menghapus spasi berlebih
text = re.sub(r'\s+', ' ', text).strip()
return text
# Sel 4: Case Folding (Fungsi)
def case_folding(text):
return text.lower()
# Cleaning dan Case Folding Data 1 ---
df_data1['kalimat_clean'] =
df_data1['kalimat'].apply(clean_text).apply(case_folding)
print("\nData 1 setelah cleaning dan case folding:")
print(df_data1[['kalimat', 'kalimat_clean']].head())
```

Pertama, digunakan fungsi *clean_text()* yang bertugas menghapus karakter yang tidak diperlukan, seperti angka, simbol, dan tanda baca, sehingga hanya huruf dan spasi yang tersisa. Setelah itu juga merapikan teks dengan menghapus spasi berlebih, termasuk spasi di awal dan akhir kalimat. Kemudian, fungsi ini diterapkan ke seluruh isi kolom kalimat dan hasilnya disimpan dalam kolom baru bernama *kalimat_clean*.

Semua teks diubah ke huruf kecil menggunakan fungsi case_folding untuk menyamakan representasi kata dan mengurangi variasi yang tidak perlu dalam proses analisis. Fungsi case_folding didefinisikan untuk mengubah teks menjadi huruf kecil dengan return text.lower().

Sebagai langkah akhir, program menampilkan lima contoh pertama dari kolom *kalimat* (asli) dan *kalimat_clean* (yang sudah dibersihkan dan di *case folding*) untuk memperlihatkan hasil perubahannya. Pembersihan ini penting agar teks menjadi lebih rapi dan seragam, sehingga lebih mudah dipahami oleh model saat dianalisis atau dilatih. Adapun hasil outpun pada tahap *cleaning data* dan *case folding* dapat dilihat pada **Tabel 3.3** sebagai berikut.

Tabel 3. 3 Hasil *Cleaning* Data dan *Case Folding*

Kalimat	Kalimat_clean
Awak dhewe njupuk pit rusa king wayah sore.	awak dhewe njupuk pit rusa king wayah sore
Kancaku nggawe roti soklat ing pawon.	kancaku nggawe roti soklat ing pawon
Kancaku ajar bal-balan ing taman cedhak griya.	kancaku ajar bal-balan ing taman cedhak griya
Aku tuku tas sekolah ing pasar.	aku tuku tas sekolah ing pasar
Wong-wong padha nonton filem ing biyoskup cera…	wong-wong padha nonton filem ing biyoskup cerak…

Tokenisasi

Tokenisasi untuk mengubah teks yang telah dibersihkan menjadi representasi numerik agar dapat digunakan. Proses ini dilakukan dengan menggunakan *Tokenizer* dari pustaka *tensorflow.keras.preprocessing.text*, yang pertama-tama diinisialisasi tanpa parameter tambahan sehingga secara otomatis akan mengumpulkan semua kata unik dalam dataset. *Tokenizer* kemudian dilatih menggunakan fungsi *fit_on_texts()* untuk membentuk kamus kata (*word index*), yaitu pemetaan setiap kata ke indeks numeriknya berdasarkan frekuensi kemunculan. Setelah kamus terbentuk, teks dalam kolom tersebut dikonversi menjadi urutan angka. Sehingga setiap kata digantikan oleh indeks yang sesuai. Selanjutnya, panjang maksimum dari seluruh urutan dihitung menggunakan *max()* untuk menentukan nilai *max_length*, yang nantinya diperlukan dalam proses *padding* agar semua urutan memiliki panjang yang sama. Langkah ini penting sebagai tahap awal dalam mempersiapkan data teks agar bisa diproses oleh model.

Kode Program 3. 4 Tahap Tokenisasi

```
1  tokenizer = Tokenizer()
2  all_kalimat_clean = pd.concat([df_data1['kalimat_clean'], df_data2['kalimat_clean']],
   ignore_index=True)
3  tokenizer.fit_on_texts(all_kalimat_clean)
4  vocab_size = len(tokenizer.word_index) + 1
5  # Konversi teks ke sequence
6  sequences_data1 = tokenizer.texts_to_sequences(df_data1['kalimat_clean'])
7  # Menentukan panjang maksimum sequence untuk data 1
8  max_length_data1 = max([len(seq) for seq in sequences_data1])
9  print("Contoh hasil tokenisasi (sebelum padding) untuk Data 1:")
10 # Display the first 5 tokenized sequences for Data 1
11 for i in range(min(5, len(sequences_data1))):
12       print(f"Original sentence: {df_data1['kalimat_clean'].iloc[i]}")
13       print(f"Tokenized sequence: {sequences_data1[i]}")
```

Tabel 3. 4 Hasil Tokenisasi

Original sentencen	Tokenized sequencen
awak dhewe njupuk pit rusak ing wayah sore	[83, 25, 265, 316, 460, 1, 261, 441]
kancaku nggawe roti soklat ing pawon	[206, 181, 75, 182, 1, 161]
kancaku ajar balbalan ing tama cedhak griya	[206, 233, 234, 1, 110, 277, 62]
aku tuku tas sekolah ing pasar	[5, 56, 266, 117, 1, 118]
wongwong padha nonton film ing biyoskop cerak kantor	[90, 317, 130, 128, 1, 157, 1259, 175]

Padding

Proses *padding* dilakukan setelah tokenisasi untuk memastikan bahwa semua urutan angka yang mewakili kalimat memiliki panjang yang seragam. Karena model pembelajaran mesin memerlukan input dengan ukuran yang konsisten, maka panjang setiap urutan angka yang dihasilkan dari tokenisasi dihitung. Blok kode ini melakukan padding pada sequence yang dihasilkan dari tokenisasi agar semua sequence memiliki panjang yang sama. Dengan menggunakan pad_sequences dari TensorFlow, kode ini menambahkan nol di akhir setiap sequence (dengan parameter padding='post') hingga panjangnya sesuai dengan *max_length* yang telah ditentukan sebelumnya. Proses ini memastikan bahwa semua *input* teks memiliki dimensi yang konsisten, yang diperlukan untuk pelatihan model CNN karena model memerlukan *input* dengan ukuran seragam. Kode kemudian menampilkan lima contoh *sequence* setelah *padding* untuk memverifikasi bahwa proses telah dilakukan dengan benar, menunjukkan representasi numerik dari kalimat dalam bentuk array dengan panjang yang sama.

Kode Program 3. 5 Tahap *Padding*

```
1  padded_sequences_data1 = pad_sequences(sequences_data1,
   maxlen=max_length_data1, padding='post')
2  print("\nContoh data 1 setelah padding:")
3  print(padded_sequences_data1[:5])
```

```
Contoh data 1 setelah padding:
[[  83   25  265  316  460    1  261  441    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0]
 [ 206  181   75  182    1  161    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0]
 [ 206  233  234    1  110  377   62    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0]
 [   5   56  266  117    1  118    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0]
 [  90  317  130  128    1  157 1259  175    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0]]
```

Gambar 3. 2 Hasil *Padding*

Label Encoding

Label kelas, yaitu ngoko, madya, dan krama, diubah menjadi bentuk numerik menggunakan LabelEncoder dan kemudian dikonversi ke one-hot encoding menggunakan to_categorical untuk keperluan pelatihan model.

Kode Program 3. 6 Tahap *Encoding* Label

```
# Encoding Label untuk data 1
label_encoder = LabelEncoder()
labels_data1_int = label_encoder.fit_transform(df_data1['kelas'])
labels_data1 = to_categorical(labels_data1_int)
# Encode label untuk data 2
label_encoder_data2 = LabelEncoder()
labels_data2_int = label_encoder_data2.fit_transform(df_data2['kelas'])
labels_data2 = to_categorical(labels_data2_int)
```

Label Encoder diinisialisasi dengan label_encoder = LabelEncoder(), label dikonversi ke integer dengan labels_data1_int = label_encoder.fit_transform(df_data1['kelas']), lalu diubah ke one-hot encoding dengan labels_data1 = to_categorical(labels_data1_int). Contoh encoding ditampilkan untuk verifikasi.

Tabel 3. 5 Hasil *Encoding* Label

Original Labels	Integer Encoded	One-Hot Encoded
Ngoko	2	[0. 0. 1.]
Madya	1	[0. 1. 0.]
Krama	0	[1. 0. 0.]

***Synthetic Minority Over-sampling Technique* (SMOTE)**

Pada Skenario 1, data digunakan secara langsung tanpa modifikasi. Variabel *padded_sequences_s1* berisi data fitur hasil padding dari proses preprocessing, sedangkan labels_s1 merupakan label dalam bentuk *one-hot encoding*. Skenario ini merepresentasikan kondisi awal dataset sebelum dilakukan penyeimbangan kelas.

Sementara itu, Skenario 2 menerapkan metode SMOTE (Synthetic Minority Oversampling Technique) untuk mengatasi ketidakseimbangan kelas. Teknik ini digunakan pada seluruh dataset untuk menghasilkan data pelatihan yang lebih seimbang. Dalam implementasinya, digunakan parameter *sampling_strategy='auto'* yang berarti semua kelas diseimbangkan ke jumlah sampel pada kelas mayoritas. Proses ini dilakukan dengan objek SMOTE dari pustaka imblearn, menggunakan *k_neighbors=5* dan *random_state* untuk memastikan replikasi hasil. Karena metode SMOTE memerlukan label dalam bentuk numerik, label one-hot (labels) dikonversi sementara menjadi bentuk integer dengan np.argmax. Setelah proses fit_resample, hasil label yang diperoleh (labels_s2_int) dikonversi kembali ke format *one-hot encoding* menggunakan fungsi to_categorical, sehingga konsisten dengan format input model.

Kode Program 3. 7 Tahap SMOTE Data 1

```
# Skenario 1: Data 1
X_data_s1 = padded_sequences_data1
y_data_s1 = labels_data1
# Skenario 2: Data 1 dengan SMOTE (menggunakan data Skenario 1 sebagai basis)
smote_s2 = SMOTE(sampling_strategy='auto', random_state=seed, k_neighbors=5)
# SMOTE requires integer labels, so convert back temporarily
X_data_s2, labels_s2_int = smote_s2.fit_resample(X_data_s1, np.argmax(y_data_s1,
axis=1))
# Convert back to one-hot encoding
y_data_s2 = to_categorical(labels_s2_int)
```

Tabel 3. 6 Hasil SMOTE Data 1

Label	Jumlah
2	500
1	500
0	500

Penerapan SMOTE pada Data 2 untuk Skenario 3 Proses SMOTE diterapkan pada data 2 untuk menambahkan data sintetis pada kelas 'madya' yang awalnya hanya memiliki 50 sampel. Teknik ini digunakan untuk menghasilkan 300 sampel sintetis kelas 'madya' dengan parameter sampling_strategy yang ditentukan berdasarkan target jumlah sampel.

Kode Program 3. 8 Tahap SMOTE Data 2 Madya

```
# Pastikan X dan y untuk data2 sesuai dengan format yang dibutuhkan SMOTE (numerik)
X_data2 = padded_sequences_data2
y_data2_int = np.argmax(labels_data2, axis=1) # SMOTE butuh label integer
# Menentukan sampling_strategy SMOTE untuk kelas 'madya' di data 2
# Temukan indeks untuk kelas 'madya'
madya_class_index_data2 = label_encoder_data2.transform(['madya'])[0]
current_madya_count_data2 = np.sum(y_data2_int == madya_class_index_data2)
target_madya_count = 300
if target_madya_count > current_madya_count_data2:
    smote_strategy_data2 = {madya_class_index_data2: target_madya_count}
else:
    smote_strategy_data2 = {} # Tidak perlu augmentasi jika target <= jumlah saat ini
if smote_strategy_data2:
    print(f"\nMenerapkan SMOTE ke dataset Data 2 untuk kelas 'madya' (dari {current_madya_count_data2} menjadi  {target_madya_count})...")
    smote_data2 = SMOTE(sampling_strategy=smote_strategy_data2, random_state=seed, k_neighbors=5)
# Terapkan SMOTE pada Data 2
    X_data2_smote, y_data2_int_smote = smote_data2.fit_resample(X_data2, y_data2_int)
# Konversi kembali label ke one-hot encoding
    y_data2_smote = to_categorical(y_data2_int_smote, num_classes=len(label_encoder_data2.classes_))
    print("\nUkuran dataset Data 2 setelah SMOTE:")
    print(f"Features: {X_data2_smote.shape}, Labels: {y_data2_smote.shape}")
# Tampilkan distribusi kelas setelah SMOTE pada Data 2
    print(f"Distribusi kelas Data 2 setelah SMOTE:\n{pd.Series(np.argmax(y_data2_smote, axis=1)).value_counts().sort_index()}")
# Variabel yang bisa Anda gunakan selanjutnya adalah X_data2_smote dan y_data2_smote
else:
    print("\nJumlah sampel 'madya' di Data 2 sudah mencapai atau melebihi target (300). SMOTE tidak diterapkan untuk kelas ini.")
    X_data2_smote = X_data2
```

```
28        y_data2_smote = labels_data2
```

Kode Program 3. 9 Hasil SMOTE Data 2 Madya

Label	Jumlah
2	500
1	300
0	500

Setelah penerapan SMOTE, ukuran dataset data 2 meningkat menjadi 1300 sampel dengan distribusi kelas sebagai berikut: ngoko sebanyak 500, madya sebanyak 300, dan krama sebanyak 500. Hasil ini menunjukkan bahwa SMOTE berhasil menambahkan sampel sintetis untuk kelas minoritas, siap untuk digabungkan dengan data lain.

Selanjutnya untuk data skenario 3, menggabungkan Data 1 dengan data *madya* dari Data 2 yang telah dilakukan penambahan data sintetis dengan SMOTE untuk menciptakan dataset yang lebih seimbang dengan total 1500 sampel. Pertama memfilter data *madya* dari Data 2 setelah SMOTE menggunakan indeks kelas *madya* dari *label_encoder_data2*. Kemudian, menggabungkan fitur dan label dari Data 1 dengan data *madya* ini menggunakan *np.concatenate*. Karena kedua dataset sudah diproses dengan *tokenizer* dan *max_length* yang sama, penggabungan ini aman. Periksa konsistensi *label encoder* untuk memastikan kelas seperti *ngoko*, *madya*, dan *krama* terpetakan dengan benar.

Kode Program 3. 10 Pembentukan data untuk skenario 3

```
1   # Skenario 3: Gabungan Data 1 dan Data 2 (Madya yang sudah di-SMOTE)
2   madya_index_in_y2 = label_encoder_data2.transform(['madya'])[0]
3   # Filter X_data2_smote dan y_data2_smote hanya untuk kelas 'madya'
4   X_madya_data2_smote = X_data2_smote[np.argmax(y_data2_smote, axis=1) ==
    madya_index_in_y2]
5   y_madya_data2_smote = y_data2_smote[np.argmax(y_data2_smote, axis=1) ==
    madya_index_in_y2]
6   print(f"\nUkuran data 'madya' dari Data 2 setelah SMOTE:")
7   print(f"Features: {X_madya_data2_smote.shape}, Labels:
    {y_madya_data2_smote.shape}")
8   # Ambil data dari Data 1 (semua kelas)
9   X_all_data1 = X_data_s1
10  y_all_data1 = y_data_s1
11  # Gabungkan X dan y dari Data 1 dan data 'madya' dari Data 2 (setelah SMOTE)
12  X_data_s3 = np.concatenate((X_all_data1, X_madya_data2_smote), axis=0)
13  y_data_s3 = np.concatenate((y_all_data1, y_madya_data2_smote), axis=0)
14  print("\nUkuran dataset Skenario 3 (Gabungan Data 1 dan Madya Data 2 setelah
    SMOTE):")
15  print(f"Features: {X_data_s3.shape}, Labels: {y_data_s3.shape}")
16  y_data_s3_int = np.argmax(y_data_s3, axis=1)
17  # Cek urutan kelas di kedua encoder
18  print("\nUrutan kelas di label_encoder (dari Data 1):", label_encoder.classes_)
19  print("Urutan kelas di label_encoder_data2 (dari Data 2):",
    label_encoder_data2.classes_)
20  # Cek indeks madya:
21  madya_idx_enc1 = label_encoder.transform(['madya'])[0]
```

```
madya_idx_enc2 = label_encoder_data2.transform(['madya'])[0]
if madya_idx_enc1 != madya_idx_enc2:
    print("Warning: Indeks kelas 'madya' berbeda antara label_encoder dan label_encoder_data2.")
# Buat Series dari integer labels dan map ke nama kelas
combined_labels_s3 = pd.Series(y_data_s3_int).map(lambda x: label_encoder_data2.inverse_transform([x])[0])
print(f"\nDistribusi kelas dataset Skenario 3:")
print(combined_labels_s3.value_counts().sort_index())
# Variabel X_data_s3 dan y_data_s3 sekarang berisi data gabungan untuk Skenario 3.
```

Tabel 3. 7 Hasil Akhir Data untuk Skenario 3

Ngoko	500
Madya	500
Krama	500

```
Ukuran data 'madya' dari Data 2 setelah SMOTE:
Features: (300, 37), Labels: (300, 3)

Ukuran dataset Skenario 3 (Gabungan Data 1 dan Madya Data 2 setelah SMOTE):
Features: (1500, 37), Labels: (1500, 3)

Urutan kelas di label_encoder (dari Data 1): ['krama' 'madya' 'ngoko']
Urutan kelas di label_encoder_data2 (dari Data 2): ['krama' 'madya' 'ngoko']

Distribusi kelas dataset Skenario 3:
krama    500
madya    500
ngoko    500
Name: count, dtype: int64
```

Gambar 3. 3 Hasil Akhir Data untuk Skenario 3

Implementasi Model CNN

Model CNN diterapkan dengan arsitektur yang mencakup lapisan embedding, konvolusi, pooling, dan dense untuk mengklasifikasikan tingkat tutur bahasa Jawa.

Membangun Model CNN

Model ini diimplementasikan menggunakan framework Keras dengan arsitektur Sequential. Fungsi build_cnn_model(vocab_size, max_length) digunakan untuk membangun dan mengembalikan model CNN dengan parameter sesuai dataset. Adapun lapisan model CNN sebagai berikut.

Kode Program 3. 11 Model CNN

```
def build_cnn_model(vocab_size, max_length):
    model = Sequential([
        Embedding(vocab_size, 100, input_length=max_length),
        Conv1D(filters=128, kernel_size=5, activation='relu'),
        MaxPooling1D(pool_size=2),
        Flatten(),
        Dense(64, activation='relu'),
        Dense(3, activation='softmax')  # 3 kelas: ngoko, madya, krama
])
```

```
    model.compile(optimizer='adam', loss='categorical_crossentropy',
    metrics=['accuracy'])
    return model
```

a. *Embedding Layer*

Lapisan ini digunakan untuk mengubah kata-kata yang ada dalam data teks menjadi representasi vektor berdimensi tetap, memungkinkan model menangkap hubungan semantik antar kata. Parameter vocab_size menentukan jumlah kata unik dalam dataset, 100 menentukan dimensi vektor *embedding* yang dipilih secara empiris untuk menangkap banyak informasi, dan max_length menentukan panjang maksimal *input* teks yang diproses, penting untuk *padding* agar semua *input* teks memiliki panjang seragam.

b. *Convolutional Layer*

Lapisan konvolusi ini digunakan untuk data sekuensial seperti teks, memanfaatkan filter untuk mengekstraksi fitur lokal dalam urutan kata. Parameter filters=128 menentukan jumlah filter untuk mengekstrak fitur, kernel_size=5 menentukan ukuran jendela konvolusi untuk menangkap n-gram lokal, dan activation='relu' memperkenalkan non-linearitas untuk pembelajaran yang lebih kompleks.

c. *Max Pooling Layer*

Lapisan ini mengurangi dimensi fitur yang dihasilkan oleh Conv1D sambil mempertahankan informasi paling penting dengan mengambil nilai maksimum dari tiap jendela, membantu model menjadi invariant terhadap pergeseran kecil. Parameter pool_size=2 adalah ukuran jendela pooling yang umum digunakan untuk mengurangi dimensi output setengahnya.

d. *Flatten layer*

Lapisan ini mengubah data multi-dimensi hasil konvolusi dan pooling menjadi vektor panjang yang dapat diteruskan ke lapisan dense, karena lapisan fully connected hanya menerima input dalam bentuk vektor satu dimensi.

e. *Dense Layer*

Lapisan fully connected pertama dengan 64 unit dan ReLU sebagai fungsi aktivasi memungkinkan model menangkap hubungan kompleks dan memperkenalkan non-linearitas, sedangkan lapisan output dengan 3 unit dan *softmax* digunakan untuk klasifikasi multi-kelas, memberikan probabilitas untuk setiap kelas (ngoko, madya, krama) dengan kelas berprobabilitas tertinggi dipilih sebagai prediksi.

f. *Compilation :* Model dikompilasi dengan optimizer Adam yang efisien dalam melatih jaringan saraf karena menggabungkan kelebihan momentum dan RMSProp, loss categorical crossentropy untuk menghitung selisih antara distribusi probabilitas yang diprediksi dan label target dalam klasifikasi multi-kelas, dan metrik akurasi untuk memberikan gambaran umum seberapa baik model mengklasifikasikan data.

Pelatihan dan Evaluasi dengan *K-Fold* Setiap Skenario

Pada tahap ini, kita melakukan pelatihan dan evaluasi model menggunakan *K-Fold Cross Validation* untuk setiap skenario yang telah dijelaskan sebelumnya, yaitu skenario 1 (Data Asli), skenario 2 (Data SMOTE) dan skenario 3. *K-Fold Cross Validation* membagi data menjadi beberapa fold, di mana setiap fold digunakan sebagai data uji secara bergantian, sementara data lainnya digunakan untuk pelatihan model. Teknik ini memastikan evaluasi yang lebih objektif dan menyeluruh.

Proses dimulai dengan inisialisasi matriks kebingungannya untuk setiap skenario, yaitu skenario 1 (Data Asli), kkenario 2 (Data SMOTE) dan skenario 3. Matriks kebingungannya digunakan untuk memvisualisasikan kesalahan klasifikasi yang terjadi pada setiap kelas dalam dataset. Kemudian, ketiga skenario didefinisikan dengan memilih dataset yang sesuai: Skenario 1 menggunakan data 1, skenario 2 menggunakan data 1 yang telah

diseimbangkan menggunakan SMOTE, skenario 3 menggunakan gabungan data skenario 1 dan data 2 madya yang telah dibuat tambahan data sintetis dengan smote.

Kode Program 3. 12 Mempersiapkan skenario dan matriks kebingungan

```
results = []
confusion_matrices = {
    'Skenario 1 (Data Asli)': np.zeros((3, 3)),
    'Skenario 2 (SMOTE dari Skenario 1)': np.zeros((3, 3)),
}
# Use data2_available to check if Skenario 3 data can be processed
if data2_available:
    confusion_matrices['Skenario 3 (Skenario 1 + SMOTED Madya Data 2)'] =
    np.zeros((3, 3))
scenarios = {
    'Skenario 1 (Data Asli)': (X_data_s1, y_data_s1),
    'Skenario 2 (SMOTE dari Skenario 1)': (X_data_s2, y_data_s2),
}
# Use data2_available to include Skenario 3 if data is available
if data2_available:
# Note: The code for creating X_data_s3 and y_data_s3 (combined data1 and smoted
data2 madya)
    scenarios['Skenario 3 (Skenario 1 + SMOTED Madya Data 2)'] = (X_data_s3,
    y_data_s3)
```

Setelah itu, untuk masing-masing skenario, dilakukan *K-Fold Cross Validation* dengan menggunakan *Stratified K-Fold* untuk memastikan distribusi kelas yang proporsional di setiap fold. Pada setiap fold, data dibagi menjadi dua bagian: data latih (*training*) dan data uji (*validation*). Model CNN kemudian dilatih menggunakan data latih dan diuji dengan data uji. Proses ini diulang untuk setiap fold, yang memungkinkan setiap bagian data diuji tepat satu kali. Selama pelatihan, model CNN dilatih selama 100 epoch dengan ukuran *batch* 32.

Kode Program 3. 13 Pelatihan dan Evaluasi

```
for scenario_name, (X_data, y_data) in scenarios.items():
    print(f"\n--- Menjalankan K-Fold untuk {scenario_name} ---")
    # Reset aggregated confusion matrix for this scenario
    scenario_cm_key = scenario_name
    # Ensure the key exists if not already initialized
    if scenario_cm_key not in confusion_matrices:
    confusion_matrices[scenario_cm_key] = np.zeros((3, 3))
    # Loop melalui setiap fold untuk skenario ini
    y_data_int = np.argmax(y_data, axis=1)
for fold, (train_index, val_index) in enumerate(kfold.split(X_data, y_data_int), 1):
    print(f"\n--- Fold {fold}/{kfold.n_splits} - {scenario_name} ---")
    # Data untuk fold ini
    X_train_fold, X_val_fold = X_data[train_index], X_data[val_index]
    y_train_fold, y_val_fold = y_data[train_index], y_data[val_index]
    print(f"Fold {fold} - Train fold: {X_train_fold.shape}, Val fold:
    {X_val_fold.shape}")
    # Bangun dan latih model
    model = build_cnn_model(vocab_size, max_length)
    # Latih model pada data latih fold dari skenario saat ini
```

```
model.fit(X_train_fold, y_train_fold, epochs=100, batch_size=32, verbose=0)
# Evaluasi pada data validasi fold dari skenario saat ini
y_val_pred = model.predict(X_val_fold)
y_val_pred_classes = np.argmax(y_val_pred, axis=1)
y_val_true_classes = np.argmax(y_val_fold, axis=1)
# Simpan hasil evaluasi
results.append(evaluate_model(y_val_true_classes, y_val_pred_classes,
scenario_name, fold))
# Update confusion matrix
confusion_matrices[scenario_cm_key] += confusion_matrix(y_val_true_classes,
y_val_pred_classes)
```

Hasil Evaluasi dan Analisis Hasil

1. Skenario 1

Pada skenario pertama ini model CNN akan dilatih langsung menggunakan *dataset* asli tanpa mempertimbangkan ketidakseimbangan kelas. Pelatihan dan evaluasi model menggunakan 10 *fold cross validation*.

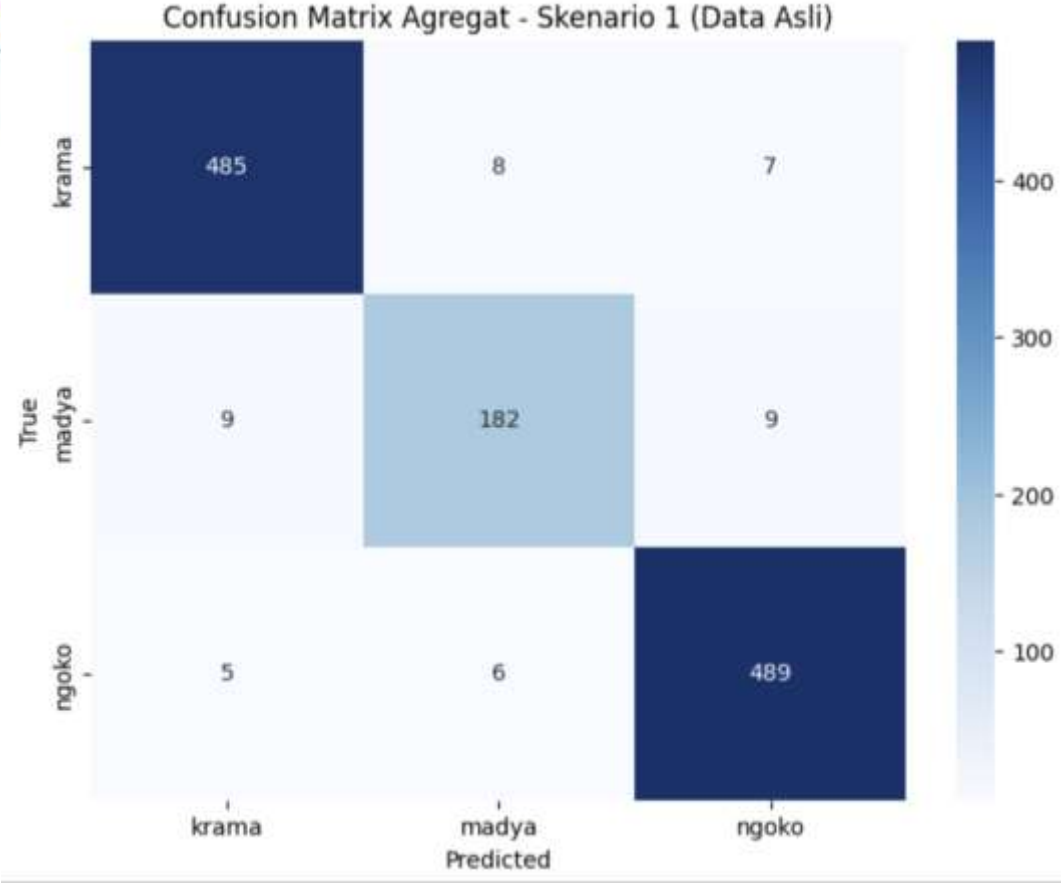
Berikut adalah nilai akurasi, presisi, *recall*, dan skor f1 tiap *fold* yang didapatkan pada skenario 1.

Tabel 3. 8 Hasil Metrik Evaluasi Skenario 1

Fold	Akurasi	Presisi	Recall	F1 Score
1	0.99	0.99	0.99	0.99
2	0.94	0.94	0.94	0.94
3	0.96	0.96	0.96	0.96
4	0.97	0.97	0.97	0.97
5	0.94	0.94	0.94	0.94
6	0.99	0.99	0.99	0.99
7	0.97	0.98	0.97	0.98
8	0.95	0.95	0.95	0.95
9	0.94	0.94	0.94	0.94
10	0.97	0.98	0.97	0.97
Rata - Rata	0.96	0.96	0.96	0.96

Hasil evaluasi pada table 4.1 menunjukkan bahwa pada Skenario 1, model klasifikasi yang dilatih dengan data asli tanpa penyeimbangan menghasilkan performa yang sangat baik. Secara rata-rata model memperoleh akurasi sebesar 0,96, presisi 0,96, *recall* 0,96, dan *F1 score* sebesar 0,96. Hal ini mengindikasikan bahwa model mampu mengenali dan mengklasifikasikan tingkat tutur bahasa Jawa (ngoko, madya, krama) secara konsisten dan akurat. Hasil tersebut menunjukkan bahwa model memiliki generalisasi yang baik tanpa *overfitting* yang signifikan.

Berikut adalah hasil *confusion matrix* dari seluruh lipatan *fold* pada skenario 1.



Gambar 3. 4 Hasil *confusion matrix* pada skenario 1

Pada Gambar 3.7 memperlihatkan hasil *confusion matrix* dari seluruh lipatan fold pada skenario 1, Sebagian besar prediksi berada pada posisi diagonal, yaitu sel-sel yang menunjukkan jumlah prediksi yang benar untuk masing-masing kelas. Hal ini menunjukkan bahwa model berhasil mengklasifikasikan tingkat tutur untuk ketiga kelas

2. Skenario 2

Pada skenario ini model melakukan pelatihan dan evaluasi menggunakan data yang telah melalui proses SMOTE (*Synthetic Minority Over-sampling Technique*) untuk menyeimbangkan distribusi kelas pada *dataset*, khususnya agar kelas minoritas memiliki jumlah data yang setara dengan kelas mayoritas..Pelatihan menggunakan *10-fold cross validation* di mana *dataset* telah diimbangi menggunakan teknik SMOTE untuk menangani ketidakseimbangan kelas.

Berikut adalah nilai akurasi, presisi, *recall*, dan skor f1 tiap *fold* yang didapatkan pada skenario 2.

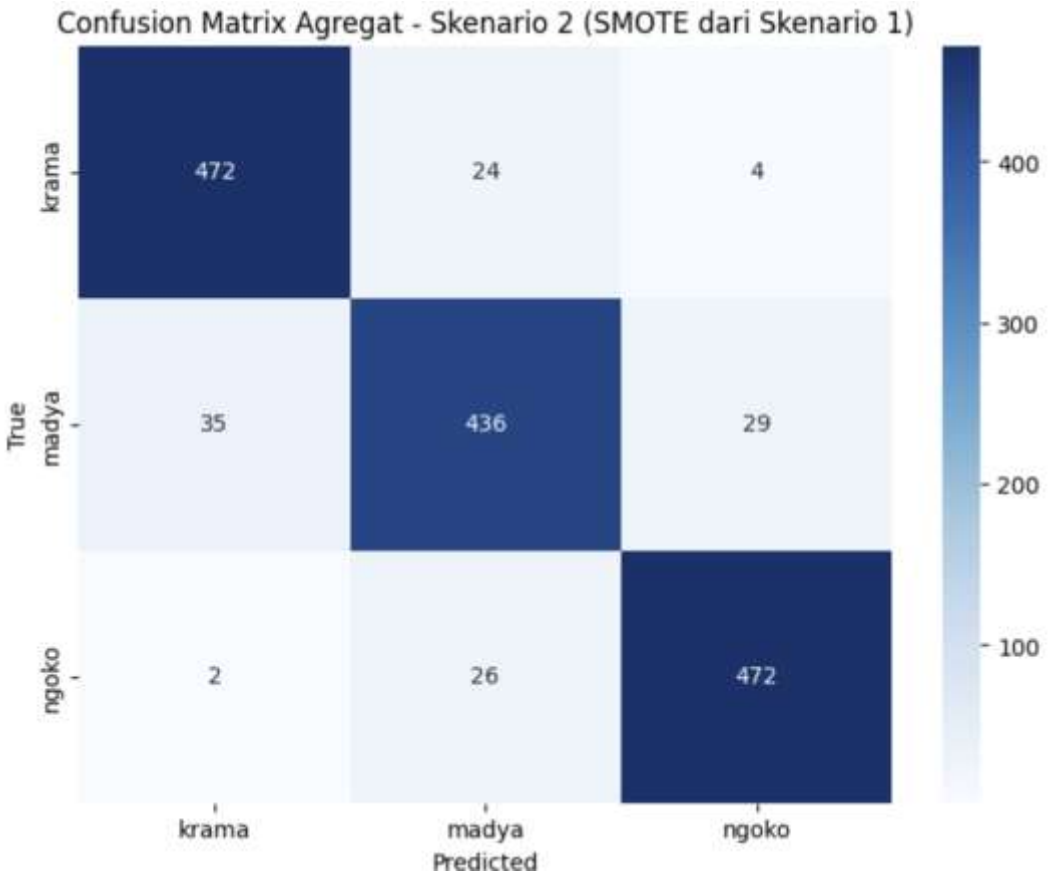
Tabel 3. 9 Hasil Metrik Evaluasi Skenario 2

Fold	Akurasi	Presisi	Recall	F1 Score
1	0.95	0.95	0.95	0.95
2	0.87	0.87	0.87	0.87
3	0.93	0.93	0.93	0.93
4	0.96	0.96	0.96	0.96
5	0.93	0.93	0.93	0.93
6	0.93	0.93	0.93	0.93
7	0.93	0.93	0.93	0.93

8	0.93	0.93	0.93	0.93
9	0.89	0.90	0.89	0.89
10	0.87	0.87	0.87	0.86
Rata - Rata	0.92	0.92	0.92	0.92

Hasil yang diperoleh pada skenario 2 model dilatih dengan data yang telah diseimbangkan menggunakan teknik SMOTE. Tabel 3.2 menunjukkan hasil metrik evaluasi pada setiap *fold*. Rata-rata akurasi, presisi, *recall*, dan *F1 score* masing-masing berada pada angka 0,92. Meskipun menunjukkan penurunan performa secara keseluruhan dibandingkan skenario 1, model tetap menunjukkan performa yang stabil dan mampu memberikan prediksi yang relevan.

Berikut adalah hasil *confusion matrix* dari seluruh lipatan *fold* pada skenario 2.



Gambar 3. 5 Hasil *confusion matrix* skenario 2

Pada Gambar 3.8 menampilkan *confusion matrix* hasil dari skenario 2 menggunakan SMOTE, Sebagian besar prediksi masih berada pada posisi diagonal matrix, yang menandakan bahwa banyak data berhasil diklasifikasikan dengan benar. Namun terlihat adanya sedikit penyebaran prediksi, terutama pada kelas madya. Hal ini menunjukkan bahwa penerapan teknik SMOTE membantu meningkatkan kemampuan model dalam mengenali kelas minoritas dengan lebih adil, meskipun disertai dengan sedikit penurunan ketepatan prediksi pada kelas mayoritas.

3. Skenario 3

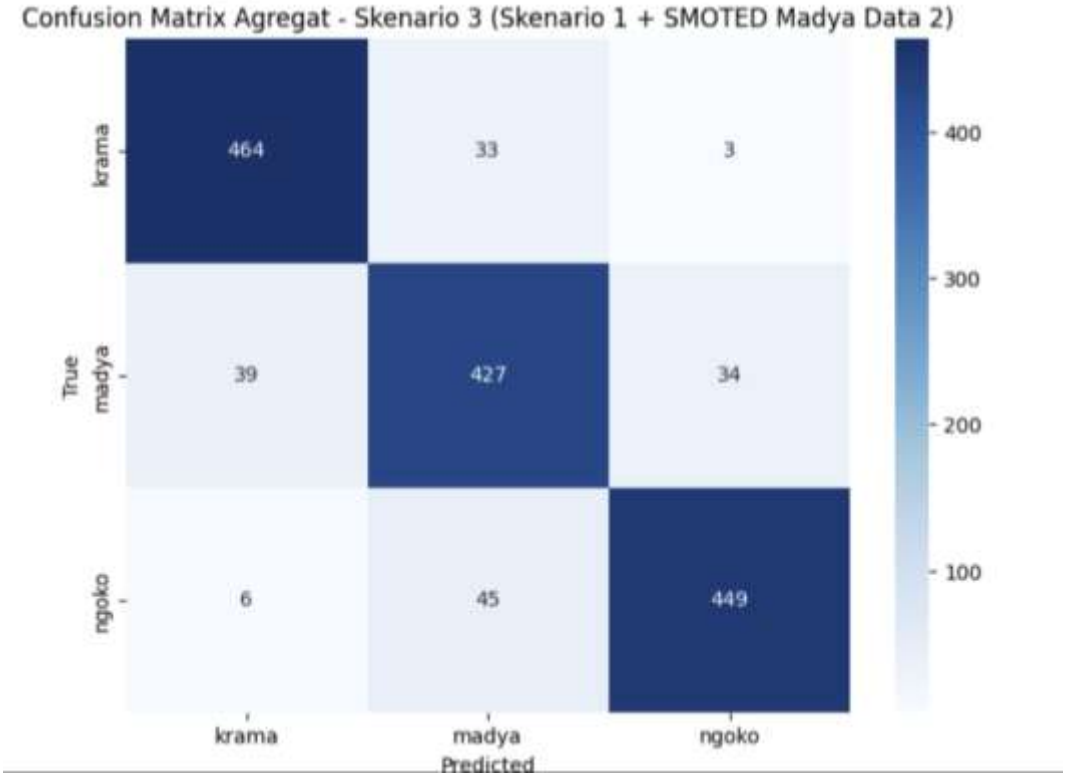
Berikut adalah nilai akurasi, presisi, *recall*, dan skor f1 tiap *fold* yang didapatkan pada skenario 3.

Tabel 3. 10 Hasil Metrik Evaluasi Skenario 3

Fold	Akurasi	Presisi	Recall	F1 Score
1	0.93	0.93	0.93	0.93
2	0.87	0.88	0.87	0.87
3	0.90	0.90	0.90	0.90
4	0.91	0.91	0.91	0.91
5	0.88	0.88	0.88	0.88
6	0.89	0.90	0.89	0.89
7	0.84	0.84	0.84	0.83
8	0.89	0.89	0.89	0.89
9	0.90	0.90	0.90	0.90
10	0.92	0.92	0.92	0.92
Rata - Rata	0.89	0.89	0.89	0.89

Hasil yang diperoleh pada skenario 3 model dilatih dengan data untuk scenario 3 yang telah dijelaskan sebelumnya. Tabel 3.3 menunjukkan hasil metrik evaluasi pada setiap *fold*. Rata-rata akurasi, presisi, *recall*, dan *F1 score* masing-masing berada pada angka 0,89.

Berikut adalah hasil *confusion matrix* dari seluruh lipatan *fold* pada skenario 2.



Gambar 3. 6 Hasil *confusion matrix* skenario 3

Pada Gambar 3.9 menampilkan *confusion matrix* hasil dari skenario 3, sebagian besar prediksi masih berada pada posisi diagonal matrix, yang menunjukkan bahwa model berhasil mengklasifikasikan banyak data dengan benar. Namun, terlihat adanya sedikit penyebaran prediksi, terutama pada kelas madya. Skenario 3 menunjukkan prediksi yang cukup akurat

untuk kelas krama dan ngoko, tetapi beberapa data pada kelas madya masih diprediksi sebagai kelas krama dan ngoko, meskipun jumlahnya tidak terlalu signifikan

Dari penjabaran hasil evaluasi pada ketiga skenario sebelumnya, berikut adalah **Tabel 3.3** yang berisi hasil rata-rata metrik evaluasi tiap *fold* dari setiap skenario. Hasil rata rata ini memberikan gambaran menyeluruh tentang kinerja model CNN setiap skenario.

Tabel 3. 11 Hasil Rata - Rata Metrik Evaluasi Setiap Skenario

Skenario	Akurasi	Precision	Recall	F1 Score
1	0.96	0.96	0.96	0.96
2	0.92	0.92	0.92	0.92
3	0.89	0.89	0.89	0.89

1. Analisis Berdasarkan K-Fold Cross-Validation

Pada penelitian ini, digunakan *10-Fold Stratified Cross-Validation* untuk mengevaluasi performa model CNN pada masing-masing skenario. Metode ini dipilih untuk memastikan bahwa setiap lipatan (*fold*) data uji memiliki distribusi kelas yang proporsional, merefleksikan distribusi kelas pada keseluruhan dataset skenario. Hasil evaluasi per *fold* memberikan gambaran mengenai stabilitas performa model terhadap variasi subset data pelatihan dan pengujian.

Skenario 1 (Data Asli): Terlihat bahwa metrik evaluasi untuk skenario 1 menunjukkan nilai yang konsisten tinggi di sebagian besar *fold* (berkisar antara 0.95 hingga 0.99). Ini mengindikasikan bahwa model memiliki performa yang stabil ketika dilatih dan diuji pada dataset skenario 1. Variasi antar *fold* relatif kecil, menunjukkan bahwa model tidak terlalu sensitif terhadap pembagian data pada skenario ini.

Skenario 2 (SMOTE dari data Skenario 1): Setelah penerapan SMOTE pada data Skenario 1, performa metrik per fold masih menunjukkan konsistensi yang baik, meskipun rata-ratanya sedikit menurun dibandingkan skenario 1. Metrik per *fold* berkisar antara 0.93 hingga 0.97. Hal ini menunjukkan bahwa SMOTE berhasil membantu model mempertahankan performa yang stabil meskipun dengan data sintetis, dan proses validasi silang tetap efektif.

Skenario 3: Pada Skenario 3, variasi metrik per *fold* cenderung lebih terlihat dibandingkan dua skenario sebelumnya, dengan rentang nilai yang lebih lebar (akurasi berkisar antara 0.79 hingga 0.90). Ini bisa mengindikasikan bahwa penambahan data madya yang baru, meskipun sudah diseimbangkan, mungkin memperkenalkan variabilitas baru atau karakteristik data yang sedikit berbeda, sehingga performa model lebih bervariasi di setiap *fold*.

Analisis per *fold* ini dilakukan untuk menunjukkan bahwa hasil rata-rata metrik yang disajikan di bagian sebelumnya bukanlah kebetulan dari satu kali pembagian data, melainkan representasi dari performa model yang dievaluasi secara robust di berbagai subset data.

2. Analisis Berdasarkan *Confusion Matrix*

Analisis *confusion matrix* memberikan wawasan yang mendalam mengenai jenis kesalahan prediksi yang dilakukan oleh model pada setiap skenario.

Skenario 1 (Data Asli): Confusion matrix menunjukkan bahwa model Skenario 1 memiliki performa yang sangat baik dalam mengklasifikasikan kelas 'ngoko' dan 'krama', dengan sedikit kesalahan prediksi ke kelas lain. Namun, kesalahan prediksi yang paling sering terjadi adalah salah mengklasifikasikan kelas 'madya' sebagai 'ngoko' atau 'krama'. Hal ini konsisten dengan distribusi data yang tidak seimbang di Skenario 1, di mana kelas 'madya' memiliki jumlah sampel yang jauh lebih sedikit, membuat model kesulitan untuk mempelajarinya dengan baik.

Skenario 2: *confusion matrix* skenario 2 menunjukkan peningkatan dalam identifikasi kelas 'madya'. Jumlah sampel 'madya' yang salah diklasifikasikan menurun drastis dibandingkan skenario 1, dan distribusi kesalahan prediksi menjadi lebih merata di antara

kelas-kelas. Ini membuktikan efektivitas SMOTE dalam menangani ketidakseimbangan kelas, memungkinkan model untuk memprediksi kelas minoritas dengan lebih akurat. Namun, masih ada beberapa kesalahan silang antar kelas, terutama antara 'madya' dengan 'ngoko' dan 'krama'.

Skenario 3: *Confusion matrix* Skenario 3 menunjukkan bahwa model masih memiliki kesulitan dalam membedakan kelas 'madya' dari 'ngoko' dan 'krama'. Jumlah kesalahan prediksi untuk kelas 'madya' (baik True Madya diprediksi salah, maupun kelas lain diprediksi Madya) lebih tinggi dibandingkan Skenario 2. Hal ini mungkin disebabkan oleh karakteristik linguistik pada data tambahan data madya yang, meskipun dilabeli sebagai 'madya', mungkin memiliki kemiripan dengan 'ngoko' atau 'krama' dari dataset awal, atau sebaliknya. Analisis lebih lanjut pada sampel yang salah diklasifikasikan di skenario ini diperlukan untuk mengidentifikasi pola spesifik kesalahan tersebut.

PENUTUP

Kesimpulan

Berdasarkan keseluruhan hasil penelitian yang telah dilakukan, maka dapat diperoleh beberapa kesimpulan sebagai berikut.

a. Dalam penelitian ini, penerapan model CNN telah berhasil dilakukan untuk mengklasifikasikan tingkat tutur teks bahasa Jawa, yaitu ngoko, madya, dan krama. Model CNN dengan arsitektur sederhana (*embedding*, konvolusi, *pooling*, dan *dense layer*), optimizer Adam, dan pelatihan selama 100 epochs mampu mengenali dan mengklasifikasikan teks pada dataset melalui tiga skenario eksperimen: data asli, data dengan SMOTE, dan gabungan data asli dengan data sintetis madya. Penerapan ini menunjukkan bahwa CNN merupakan metode yang efektif untuk tugas klasifikasi tingkat tutur teks bahasa Jawa
b. Evaluasi menggunakan 10-fold cross-validation menunjukkan performa model yang konsisten. Pada Skenario 1, rata-rata akurasi, presisi, recall, dan F1-score mencapai 0,96, menunjukkan kemampuan model yang sangat baik pada data asli, meskipun bias terhadap kelas mayoritas (ngoko dan krama). Skenario 2, dengan data yang diseimbangkan menggunakan SMOTE, menghasilkan rata-rata metrik sebesar 0,92, dengan peningkatan signifikan pada klasifikasi kelas madya. Skenario 3 menghasilkan rata-rata metrik sebesar 0,89, dengan performa yang sedikit menurun karena variasi karakteristik data tambahan madya. Skenario 2 memberikan keseimbangan terbaik antara akurasi dan kemampuan mengklasifikasikan semua kelas secara merata.

Saran

Penelitian ini tentu memiliki keterbatasan, sehingga saran berikut diharapkan dapat menjadi masukan untuk penelitian selanjutnya.

a. Pengembangan dataset dapat dilakukan dengan menggunakan dataset yang lebih besar dan beragam, mencakup berbagai konteks kalimat serta variasi dialek, sehingga model menjadi lebih general dan robust.
b. Mengeksplorasi penggunaan arsitektur deep learning lain yang lebih kompleks atau berbeda seperti LSTM, GRU, atau model berbasis Transformer, yang mungkin lebih efektif dalam menangkap konteks dan hubungan sekuensial dalam teks.

DAFTAR PUSTAKA

Amrozi, Y., Yuliati, D., Susilo, A., Novianto, N., & Ramadhan, R. (2022). Klasifikasi Jenis Buah Pisang Berdasarkan Citra Warna dengan Metode SVM. *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, *11*(3), 394–399. https://doi.org/10.32736/sisfokom.v11i3.1502

Atmawati, D. (2021). Language Politeness in the Javanese Verb Speech Level. *Lingua Cultura*, *15*(1), 51–57. https://doi.org/10.21512/lc.v15i1.7109

Bej, S., Davtyan, N., Wolfien, M., Nassar, M., & Wolkenhauer, O. (2021). LoRAS: an oversampling approach for imbalanced datasets. *Machine Learning*, *110*(2), 279–301. https://doi.org/10.1007/s10994-020-05913-4

Damuri, A., Riyanto, U., Rusdianto, H., & Aminudin, M. (2021). Implementasi Data Mining dengan Algoritma Naïve Bayes Untuk Klasifikasi Kelayakan Penerima Bantuan Sembako. *JURIKOM (Jurnal Riset Komputer)*, *8*(6), 219. https://doi.org/10.30865/jurikom.v8i6.3655

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). A Survey on Text Classification Algorithms: From Text to Predictions. *Information (Switzerland)*, *13*(2). https://doi.org/10.3390/info13020083

Imron, S., Setiawan, E. I., & Santoso, J. (2023). Deteksi Aspek Review E-Commerce Menggunakan IndoBERT Embedding dan CNN. *Journal of Intelligent System and Computation*, *5*(1), 10–16. https://doi.org/10.52985/insyst.v5i1.267

Kim, H., & Jeong, Y. S. (2019). Sentiment classification using Convolutional Neural Networks. *Applied Sciences (Switzerland)*, *9*(11), 1–14. https://doi.org/10.3390/app9112347

Kostopoulos, A., Garefalakis, T., Michelaraki, E., Katrakazas, C., & Yannis, G. (2024). Modeling and Sustainability Implications of Harsh Driving Events: A Predictive Machine Learning Approach. *Sustainability (Switzerland)*, *16*(14). https://doi.org/10.3390/su16146151

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information (Switzerland)*, *10*(4), 1–68. https://doi.org/10.3390/info10040150

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Listyarini, S. N., & Anggoro, D. A. (2021). Analisis Sentimen Pilkada di Tengah Pandemi Covid-19 Menggunakan Convolution Neural Network (CNN). *Jurnal Pendidikan Dan Teknologi Indonesia*, *1*(7), 261–268. https://doi.org/10.52436/1.jpti.60

Poedjosoedarmo, S., Kundjana, T., Soepomo, G., & Suharso, A. (2013). *Tingkat Tutur Bahasa Jawa*. Balai Bahasa Provinsi Daerah Istimewa Yogyakarta.

Rahman, M. A., Budianto, H., & Setiawan, E. I. (2019). Aspect Based Sentimen Analysis Opini Publik Pada Instagram dengan Convolutional Neural Network. *Journal of Intelligent System and Computation*, *1*(2), 50–57. https://doi.org/10.52985/insyst.v1i2.83

Ramdhani, M. A., Ramdhani, M. A., Maylawati, D. S. adillah, & Mantoro, T. (2020). Indonesian news classification using convolutional neural network. *Indonesian Journal of Electrical Engineering and Computer Science*, *19*(2), 1000–1009. https://doi.org/10.11591/ijeecs.v19.i2.pp1000-1009

Sutoyo, E., & Fadlurrahman, M. A. (2020). Penerapan SMOTE untuk Mengatasi Imbalance Class dalam Klasifikasi Television Advertisement Performance Rating Menggunakan Artificial Neural Network. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, *6*(3), 379. https://doi.org/10.26418/jp.v6i3.42896

Simanjuntak, S. P., Berutu, S. S., & Setyawan, G. C. (2024). Implementasi metode CNN pada klasifikasi sentimen terhadap penyelenggaraan Piala Dunia U-17. Journal of Engineering and Emerging Technology, 02(01).

Terrada, O., Cherradi, B., Raihani, A., & Bouattane, O. (2019). Classification and Prediction of atherosclerosis diseases using machine learning algorithms. *2019 International Conference on Optimization and Applications, ICOA 2019, May 2020*, 1–5. https://doi.org/10.1109/ICOA.2019.8727688

Wahyuni, S., Subiyantoro, S., & Fadhilah, S. (2018). *Obstacles Level of Learning Javanese Speech: A Phenomenology Study in Elementary School*. *173*(Icei 2017), 298–301.

https://doi.org/10.2991/icei-17.2018.78
Wijayanti, N. P. Y. T., N. Kencana, E., & Sumarjaya, I. W. (2021). Smote: Potensi Dan Kekurangannya Pada Survei. *E-Jurnal Matematika*, *10*(4), 235. https://doi.org/10.24843/mtk.2021.v10.i04.p348
Wijiyanto, W., Pradana, A. I., Sopingi, S., & Atina, V. (2024). Teknik K-Fold Cross Validation untuk Mengevaluasi Kinerja Mahasiswa. *Jurnal Algoritma*, *21*(1), 239–248. https://doi.org/10.33364/algoritma/v.21-1.1618