JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

Deteksi Ucapan yang Dihasilkan oleh Artificial Intelligence Menggunakan Metode Convolutional Neural Network

Muhammad Sofyan[1], Mhd. Gilang Suryanata[2], Meri Sri Wahyuni[3]
[1 2 3] Sistem Informasi, STMIK Triguna Dharma
Email: [1]sofyanmuhammad2003@gmail.com, [2]suryanatagilang@gmail.com, [3]meri.sriwahyuni@gmail.com
Email Penulis Korespondensi: sofyanmuhammad2003@gmail.com

Abstrak
Seiring berkembangnya teknologi Artificial Intelligence (AI), khususnya teknologi sintesis suara seperti Text to Speech dan Voice Conversion, media yang dihasilkan oleh teknologi ini semakin realistis, sehingga sulit dibedakan antara suara manusia dan suara yang dihasilkan oleh AI. Fenomena ini mengakibatkan peningkatan signifikan dalam kasus penyalahgunaan, sehingga menimbulkan tantangan serius di berbagai sektor, baik dalam ranah *financial technology* (fintech) maupun non-fintech. Pada penelitian ini, dalam upaya mendeteksi ucapan hasil manipulasi AI, atau deepfake, digunakan dataset Real, Fake, and Partially fake. Seluruh data pada dataset yang mengandung suara AI digabungkan dan diberi label 'Fake', sedangkan suara manusia diberi label 'Real'. Pendeteksian ini menggunakan pendekatan ekstraksi fitur audio *Mel-Frequency Cepstral Coefficients* (MFCC), diikuti dengan pemrosesan seperti normalisasi fitur, padding, labelling, dan reshape input. Selanjutnya, data yang telah diproses akan dianalisis menggunakan arsitektur *Convolutional Neural Networks* (CNN) yang diusulkan. Hasil penelitian ini menunjukkan bahwa model arsitektur CNN yang diusulkan dapat mencapai akurasi pelatihan sebesar 99% pada data training dan validation, serta akurasi pengujian pada data testing sebesar 92% serta untuk hasil evaluasi metrik (*precision, recall, F1-score, macro avg,* dan *weighted avg*) model pada data testing cukup memuaskan, yang mana untuk setiap metrik menghasilkan lebih dari 85%, *Equal Error Rate* (EER) bernilai 8% serta *Area Under Precision-Recall Curve* (AUPRC) pada kelas 'Real' bernilai 89% dan 'Fake' bernilai 99% dan hasil sistem yang dikembangkan dapat membedakan hasil suara AI (Fake) dengan suara manusia (Real).
Kata Kunci: *Audio Deepfake Detection, Partial Fake, Mel-Frequency Cepstral Coefficients, Convolutional Neural Networks*

Abstract
With the advancement of Artificial Intelligence (AI) technology, particularly in voice synthesis technologies like Text to Speech and Voice Conversion, media generated by these technologies have become increasingly realistic, making it difficult to differentiate between human voices and those produce AI. This phenomenon has led to a significant increase in cases of misuse, posing serious challenges across various sectors, both in the realm of financial technology (fintech) and non-fintech. In this research, to detect AI-generated or deepfake speech, the Real, Fake, and Partially fake dataset was utilized. All data in the dataset containing AI-generated voices were combined and labeled as 'Fake', while human voices were labeled as 'Real'. The detection process employed a Mel-Frequency Cepstral Coefficients (MFCC) audio feature extraction approach, followed by processing steps such as feature normalization, padding, labeling, and input reshaping. Subsequently, the processed data was analyzed using the proposed Convolutional Neural Networks (CNN) architecture. The results of this research demonstrate that the proposed CNN architecture model achieved a training accuracy of 99% on the training and validation data, and 92% on the testing data. Additionally, the evaluation metrics (precision, recall, F1-score, macro avg, and weighted avg) on the testing data were satisfactory, with each metric yielding more than 85%. Equal Error Rate (EER) was recorded at 8%, while the Area Under Precision-Recall Curve (AUPRC) for the 'Real' was 89% and for the 'Fake' was 99%. The developed system can effectively distinguish between AI-generated (Fake) and human speech (Real).
Keywords*: Audio Deepfake Detection, Partial Fake, Mel-Frequency Cepstral Coefficients, Convolutional Neural Network*

1. PENDAHULUAN

Dalam beberapa tahun terakhir, kemajuan pesat di bidang *Generative Adversarial Networks* (GANs) [1] dan *Variational Auto-encoders* (VAEs) [2], teknik *deep learning* terkemuka, media hasil *Artificial Intelligence* (AI) kini semakin realistis dan tidak dapat dibedakan secara visual dari media asli [3]. *Deepfake*, berasal dari singkatan *deep learning* dan *fake*, sebagai produk dari teknologi AI, adalah manipulasi media digital yang melibatkan penggantian identitas asli manusia (e.g. foto, video, atau rekaman suara) dengan media sintetis yang dihasilkan oleh model AI, terutama dengan teknik *deep learning* [4].

Seiring dengan pesatnya kemajuan dalam teknologi sintesis ucapan berbasis AI, seperti model *Text-to-Speech* (TTS) [5]-[7] dan *Voice Conversion* (VC) [8]-[11], telah membuat peningkatan yang signifikan dalam kasus penyalahgunaan dan menyebabkan tantangan serius dalam berbagai aspek. Dalam satu tahun terakhir, upaya verifikasi identitas palsu menggunakan audio *deepfake* pada institusi keuangan, khususnya bank, mengalami eskalasi sebesar 3000% [12]. Sebagai contoh, sebuah firma di Hongkong kehilangan USD 20 jt setelah tertipu dengan media yang dihasilkan oleh AI yang berkamuflase sebagai pejabat senior perusahaannya untuk mentransfer sejumlah uang [13]. Tidak hanya itu, penipuan hasil *voice cloning* juga menghampiri Mark Read, CEO WPP, dalam upaya mendapatkan uang dan informasi pribadi [14].

Selain berdampak pada institusi keuangan, penyalahgunaan teknologi sintesis ucapan juga merambah ke ranah non-*fintech*. Penipuan berbasis *Voice Conversion* (VC), atau yang dikenal juga sebagai *voice cloning*, telah dimanfaatkan untuk mengeksploitasi individu secara finansial (e.g. *phone scam*) [15]. Berdasarkan hasil survei yang dilakukan McAfee,

JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

secara global, sebanyak 70% responden tidak percaya diri membuktikan mereka bisa atau tidaknya membedakan antara suara hasil *deepfake* dan yang asli [16].

Dataset ASVspoof2015 [17] menjadi pelopor penilitian terhadap pendeteksian suara deepfake. Pada saat itu, *dataset* tersebut menjadi set data *state-of-the-art* dalam konteks pendektesian suara deepfake atau sintesis. ASVspoof merupakan gabungan *challenge* yang dibuat untuk pengembangan dalam mencegah *Automatic Speaker Verification* (ASV) dari serangan *spoofing*, seperti ucapan hasil sintetis, pengulangan, dan konversi [18]. Penelitian mengenai pendeteksian suara *deepfake* menjadi ranah penelitian yang aktif sejak rilisnya dataset tersebut, bahkan sampai rilisnya dataset terbaru mereka, ASVspoof2021 [19]. Tidak hanya ASVspoof, dataset yang lain seperti *Fake or Real Dataset* [20], FakeAVCeleb [21], dan H-Voice [22] terlibat juga dengan penelitian berbasis pendektesian ini.

Para peneliti telah mengeksplorasi berbagai pendekatan untuk mendeteksi audio deepfake, termasuk penggunaan model *deep learning* seperti *Convolutional Neural Networks* (CNN). Berdasarkan hasil *review* yang dilakukan Almutairi dan Elgibren [23], penelitian yang menggunakan metode *deep learning* khususnya CNN menjadi pilihan yang terbaik serta memiliki kinerja dan stabilitas yang mumpuni dibandingkan dengan berbasis *machine learning*, seperti *Support Vector Machine* (SVM).

Pada penelitian ini, dalam pendetektesian ucapan hasil AI, atau *deepfake*, penulis akan menggunakan dataset terbaru, yaitu Real Fake, and Partially fake (RFP) [24]. Berbeda dengan yang lainnya, dataset ini menarik karena menyediakan data ucapan yang mencakup *partial fake*, yang mana gabungan antara suara palsu dan asli. Selain *partial fake*, penelitian ini juga akan menggunakan sub-*dataset* yang lainnya, seperti Audio_With_Noice, Converted_Voices, Text_to_Speech, serta Real, dalam melakukan penelitiannya. Metode yang akan digunakan oleh penelitian ini dalam pendeteksian ucapan hasil *deepfake* adalah model arsitektur *Convolutional Neural Network* (CNN) yang dikembangkan sendiri, Penelitian ini juga akan melakukan ekstraksi fitur audio menggunakan fitur *Mel-Frequency Cepstral Coefficients* (MFCC), yang nantinya akan di-*feed* ke model CNN. Dengan menggunakan *dataset* RFP yang belum pernah dieksplorasi sebelumnya, harapannya, penelitian ini dapat memberikan kontribusi yang signifikan terhadap pengembangan model deteksi ucapan *deepfake* serta dalam menghadapi tantangan *partial fake*.

Adapun hasil yang diharapkan dari penelitian ini yaitu dapat mengetahui hasil performa model yang didapatkan berupa metrik akurasi serta mengetahui evaluasi model yang telah didapatkan dan membangun sistem berbasis web untuk mendeteksi ucapan yang dihasilkan oleh AI, atau *deepfake*

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Dalam penelitian ini, proses deteksi suara AI atau *deepfake* melibatkan beberapa tahapan utama (**Gambar 2.1**). Mulanya, kami memanfaatkan dataset publik RFP Dataset yang terdiri dari rekaman suara asli dan deepfake. Dataset ini telah dipartisi ke dalam tiga set: *training*, *validation*, dan *testing*. Tahap *audio preprocessing* sangat penting untuk mempersiapkan data mentah agar sesuai dengan model pembelajaran. Kami mengekstrak fitur Mel-Frequency Cepstral Coefficients (MFCC) yang menangkap karakteristik spektral suara, diikuti oleh normalisasi fitur menggunakan *Z-score normalization* untuk mengurangi dampak variasi skala. Selanjutnya, kami menerapkan *padding* untuk menyamakan panjang fitur audio, serta pemberian *labelling* kelas ke dalam format *one-hot encoding* dan *reshape input* audio menjadi saluran tunggal (mono) untuk kompatibilitas dengan arsitektur Convolutional Neural Network (CNN).

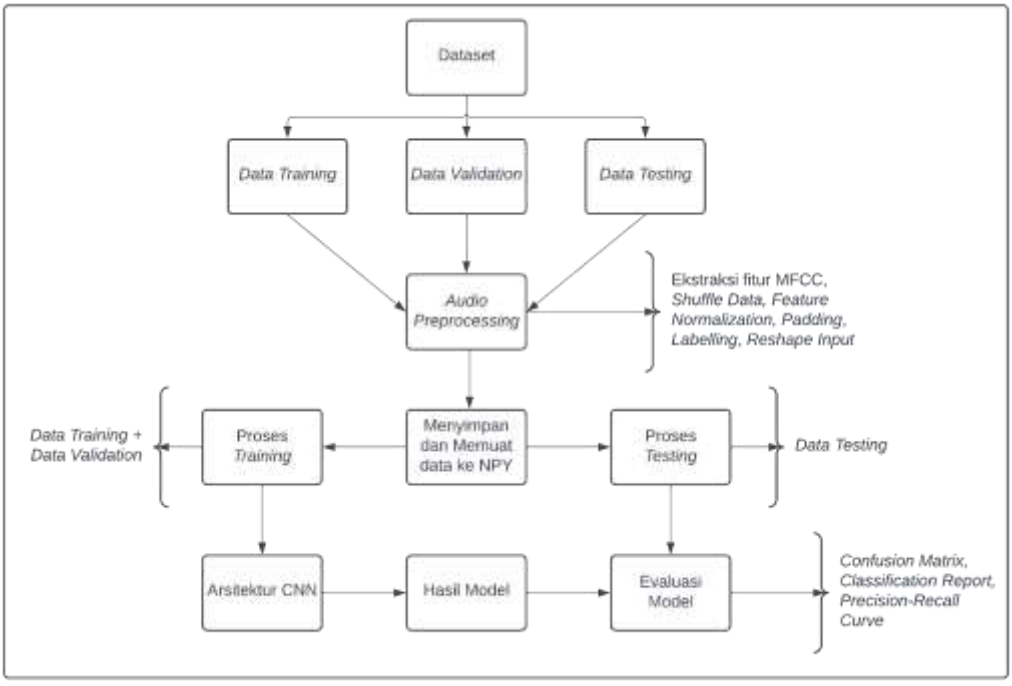
Setelah *preprocessing*, data disimpan dalam format NPY yang efisien untuk akses cepat selama *training* serta konfigurasi *training* diatur. Tahap berikutnya adalah pelatihan dan pengujian model. Proses *training* menggunakan data *training* dan *validation*, serta melibatkan penentuan arsitektur CNN yang diusulkan untuk pembelajaran pola dan menghasilkan model terbaik Terakhir, model yang dihasilkan dievaluasi menggunakan data *testing*. Evaluasi mencakup *confusion matrix* dan *classification report*, yang menyajikan metrik seperti *precision*, *recall*, *F1-score*, *Equal Error Eate* (EER), dan *Precision-Recall Curve* (PRC) untuk menilai kinerja model secara komprehensif.

2.2 *Dataset*

Dataset yang digunakan untuk mendeteksi suara hasil *artificial intelligence*, atau *deepfake*, adalah RFP Dataset atau lebih lengkapnya *Real, Fake, and Partially Fake Dataset* yang dikemukakan oleh AlAli dan Theodorakopoulos [24]. RFP Dataset merupakan salah satu dari sedikitnya dataset yang berbahasa inggris bebas terbuka yang mencakup *Partial Fake* (FP) dan dapat digunakan untuk mendeteksi ucapan palsu. Dataset ini mengandung lebih dari 127.000 *uttarances* dan terbagi menjadi dua versi, yaitu *original version* dan *normalized version*. Dataset tersebut memiliki lima tipe atau kelas audio berbeda, yaitu *real* (asli/manusia), *Text to Speech* (TTS), *Voice Conversion* (VC), *audio with noise*, dan *partial fake*. Sesuai dengan konteks penilitian ini, data yang digunakan adalah data yang sudah di tahap normalisasi (*normalised version*). *Normalised version* sendiri merupakan versi dengan data yang sudah dilakukan tahap *processing* dan siap untuk digunakan dengan *machine learning*. Semua audio pada versi ini yang sebelumnya berkarakteristik stereofonik dikonversi ke sinyal audio monofonik (mono), yang terdiri dari satu *channel*. Kemudian untuk semua *sample rate* pada audio juga dikonversi menjadi 16kHz dan hanya berformat WAV, dimana format yang umum digunakan untuk pemrosesan audio dalam *machine learning*. Versi ini memiliki distribusi gender dan total jumlah file perkategori yang seimbang, dimana

JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

penutur laki-laki dan perempuan berjumlah 115 dengan jumlah 108.385 *uttarances* yang seimbang. Rentang durasi dari file audio pada versi ini antara 3-20 detik.



Gambar 1. Alur Tahapan Penelitian

3. HASIL DAN PEMBAHASAN

3.1 Kombinasi dan Splitting Data

Pada penelitian ini, penggunaan data yang diterapkan adalah dengan menggabungkan semua data dari beberapa kelas untuk audio yang dihasilkan oleh AI menjadi satu kelas yang diberi label 'Fake', kelas yang diambil atau dikombinasikan untuk label ini adalah kelas Partial_Fake, Text_to_Speech, Converted_Voice, dan Audio_With_Noise, dengan hasil data *training* sebesar 57.374, *validation* sebesar 16.122, dan *testing* sebesar 7.448. selebihnya, untuk kelas yang tidak dilakukan kombinasi, yaitu kelas 'Real' akan tetap digunakan sebagai label 'Real' dengan jumlah data *training* sebesar 19.422, *validation* sebesar 5.489, dan *testing* sebesar 2.530.

Tabel 1. Kombinasi dan *Splitting Data*

Kelas	Deskripsi	Tipe	*Uttarance*	Ukuran
Fake	Kombinasi semua kelas yang memiliki data suara AI	*Training*	57.374	15.87 GB
		Validation	16.122	3.99 GB
		Testing	7.448	1.98 GB
Real	Suara manusia.	*Training*	19.422	8.99 GB
		Validation	5.489	2.59 GB
		Testing	2.530	1.24 GB
Total			**108.385**	**38.02 GB**

3.2 *Audio Preprocessing*

Prapemrosesan audio (*audio preprocessing*) merupakan tahapan krusial sebelum data audio digunakan sebagai masukan (*input*) untuk model CNN. Dalam penelitian ini, prapemrosesan audio melibatkan beberapa langkah, yaitu: (1) ekstraksi fitur MFCC (*Mel-Frequency Cepstral Coefficients*), (2) *Shuffle Data* (3) *Feature Normalization* (4) *padding* untuk menyeragamkan panjang sinyal audio, (5) pemberian label (*labelling*) pada data, dan (6) *reshape* data menjadi format yang sesuai untuk masukan model.

3.2.1 Ekstraksi Fitur *Mel-Frequency Cepstral Coefficients* (MFCC)

Proses pengektraksian MFCC ini dimulai meng-*import* pustaka Librosa. Setelah itu, Audio akan diubah menjadi format mono untuk menyederhanakan analisis dan memastikan kompatibilitas dengan langkah-langkah selanjutnya. Kemudian untuk *sample rate* telah disesuaikan menjadi 16kHz, yang mana merupakan nilai standar umum dalam pemrosesan audio, terutama untuk ucapan. Penyesuaian tersebut akan memastikan bahwa semua audio yang dianalisis

JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

memiliki representasi temporal yang konsisten. Setelah audio dimuat dan disiapkan, sinyal audio dinormalisasi. Proses ini, yang dilakukan oleh fungsi `librosa.util.normalize`, memastikan bahwa amplitudo sinyal audio berada dalam rentang yang konsisten, yang mana biasanya antara -1 dan 1. Normalisasi ini dianggap penting karena perbedaan amplitudo yang besar antar file audio dapat mempengaruhi perhitungan fitur MFCC secara signifikan.

Inti dari proses ini adalah ekstraksi fitur *Mel-Frequency Cepstral Coefficients* (MFCC). Hasil fitur ini dihitung menggunakan `librosa.feature.mfcc`, yang menerapkan serangkaian transformasi pada sinyal audio. Pertama, sinyal dibagi menjadi bingkai-bingkai kecil yang ditentukan oleh parameter `n_fft` (ukuran jendela *Fourier Transform*) dan `hop_length` (jumlah sampel antara pergeseran jendela berturut-turut). Parameter `num_mfcc` menentukan jumlah koefisien MFCC yang akan diekstraksi. Setelah koefisien MFCC dihitung, koefisien-koefisien tersebut menjalani proses *liftering*, yang ditentukan oleh parameter `lifter`. Proses *liftering* ini menekankan koefisien MFCC frekuensi rendah, yang sering kali lebih penting dalam pemodelan suara manusia.

Tabel 2. Parameter Librosa yang digunakan untuk ekstraksi fitur MFCC

Parameter	Deskripsi	Nilai
`num_mfcc`	Jumlah koefisien MFCC yang akan diekstrak.	13
`n_fft`	Panjang jendela *Fast Fourier Transform* (FFT) dalam sampel.	2048
`Hop_length`	Jumlah sampel yang akan dipindahkan antara jendela FFT yang berurutan (bisa juga disebut *hop size*).	256
`lifter`	Koefisien *liftering* yang digunakan untuk menekankan MFCC frekuensi rendah.	26 (`n_mfcc`*2)
`sample_rate`	Laju sampel target untuk audio.	16000 (16kHz)

3.2.2 *Shuffle Data*

Pada tahap ini, *shuffle* data diperlukan, dikarenakan jika tidak melakukan *shuffle* data model yang dihasilkan nanti akan terlalu fokus pada urutan data asli, yang mana model akan mempelajari pola spesifik dalam urutan tertentu, bukannya belajar konsep pola yang umum. Hal tersebut bisa menyebabkan model tidak bisa bekerja dengan baik dan benar pada data baru yang belum pernah dilihatnya. Proses pengacakan ini, atau *shuffle*, akan membantu mencegah masalah seperti *overfitting*. Proses ini akan membuat proses *training* lebih stabil dan hasil model lebih bisa digeneralisir dibandingkan dengan tidak digunakannya. Dengan bantuan dari *library* Scikit-learn, yang spesifiknya `sklearn.utils` dengan mengimpor fungsi `shuffle`, yang mana fungsi ini berguna untuk mengacak urutan elemen dalam *array* atau *list*. Proses ini diterapkan pada data *training*, *validation*, dan *testing*.

3.2.3 *Feature Normalization*

Setelah dilakukannya proses *shuffle*, langkah selanjutnya adalah melakukan tahap normalisasi. Pada peneliitian ini, penerapan normalisasi dilakukan secara manual dengan hanya menggunakan *library* Numpy yang bertujuan untuk menormalisasikan atau menstandarisasi fitur MFCC yang sebelumnya sudah diekstrak. Pertama-tama untuk mencari nilai *mean* dan standard deviasi global akan digunakannya data dari koefisien MFCC dalam data *training*, yang mana bertujuan untuk menghindari kebocoran data (*data leakage*). Mean global adalah nilai rata-rata dari semua nilai MFCC, dan sedangkan standar deviasi global adalah untuk mengukur seberapa tersebar nilai-nilai MFCC dari rata-rata tersebut. Selanjutnya setelah nilai *mean* dan standar deviasi sudah diinisialisasikan, setiap koefisien MFCC dalam data *training* dinormalisasi menggunakan dengan teknik *Z-score normalization*. Ini dilakukan dengan mengurangkan setiap nilai MFCC dengan *mean* global dan membaginya dengan standar deviasi global. Hasilnya adalah nilai-nilai baru yang memiliki rata-rata 0 dan standar deviasi 1. Terakhir, tahapan normalisasi yang sama diterapkan pada data *validation* dan data *testing*.

3.2.4 *Padding*

Setelah ekstraksi koefisien MFCC sudah dinormalisasikan maka diperolehlah matriks (*array*) fitur untuk setiap berkas audio. Dimensi matriks fitur ini bervariasi tergantung pada durasi audio, sehingga audio yang lebih panjang menghasilkan lebih banyak *frame* dan audio yang pendek menghasilkan sedikit *frame*. Oleh karena itu, normalisasi ukuran matriks setiap *frame* dan MFCC diperlukan sebelum menjadi masukan model CNN, dikarenakan *input*-an harus konsisten. Panjang maksimum (*max_length*) untuk normalisasi ditentukan dengan jumlah detik tertinggi pada dataset RFP, yang mana berdurasi 20 detik. Jadi, untuk mengetahui jumlah *frame* untuk durasi tersebut dapat ditentukan dengan perhitungan berikut ini:

$$\text{num\_frames} = \frac{\text{durasi audio} \times \text{sample rate}}{\text{hop\_length}}$$

$$\text{num\_frames} = \frac{20 \times 16000}{256} = \mathbf{1250}$$

JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

Setelah panjang maksimum ditentukan, semua urutan dalam dataset *training*, *validation*, dan *testing* dipadatkan (*padding*) agar memiliki panjang yang sama dengan *max_length*. Proses *padding* ini dilakukan dengan menambahkan nol di akhir urutan yang lebih pendek. Parameter `padding='post'` dalam fungsi `pad_sequences` dengan menggunakan *library* pada Keras, yang mana secara eksplisit menginstruksikan agar *padding* dilakukan di akhir urutan.

3.2.5 *Labelling*

Dalam tahap preprocessing data audio, `LabelEncoder` dari `sklearn.preprocessing` berperan penting dalam mengubah label kelas suara yang berupa teks (misalnya, nama folder) menjadi representasi numerik. Transformasi ini esensial karena dalam penelitian ini menggunakan CNN, yang mana beroperasi menggunakan angka, bukan teks. Metode `fit_transform` diterapkan pada label data *training* untuk menghitung label unik dan menghasilkan representasi numerik yang sesuai. Selanjutnya, transform digunakan pada label data *validation* dan *testing* untuk memastikan konsistensi pengkodean dengan menggunakan pemetaan yang sama yang dipelajari dari label data *training*.

Setelah label diubah menjadi numerik, langkah selanjutnya adalah melakukan *one-hot encoding* menggunakan fungsi `to_categorical` dari Keras. Proses ini mengubah label numerik menjadi format *one-hot encoding*, yaitu representasi biner yang krusial untuk output dalam skenario klasifikasi multikelas atau kelas biner. Dimana jika terdapat 2 kelas, label "0" akan diubah menjadi [1, 0], label "1" menjadi [0, 1]. Format *one-hot encoding* ini memungkinkan model pada penelitian ini dapat dilatih menggunakan fungsi loss seperti `binary_crossentropy` yang umum digunakan untuk klasifikasi kelas *binary*.

3.2.6 *Reshape Input*

Reshape Input ini berfungsi untuk mengubah bentuk data fitur MFCC menjadi format yang diharapkan oleh model CNN dengan menambahkan channel 1, yang mana juga mempresentasikan data saluran audio tunggal (mono). *Output* dari proses ini menjadi *input*-an dengan *shape* (1250,13,1), sesuai dengan 1250 *frame*, 13 koefisien MFCC per *frame*, dan mono.

3.3 Model Arsitektur *Convolutional Neural Network*

Bagian ini membahas arsitektur model yang diusulkan. Secara umum, arsitektur *Convolutional Neural Network* (CNN) terdiri dari lapisan *convolution* dan *pooling*. Dalam penelitian ini, digunakan dua *convolution block*, di mana setiap *block* terdiri dari dua lapisan *convolution* yang masing-masing diikuti oleh *Batch Normalization* untuk menstabilkan proses *training*, lapisan *pooling*, dan lapisan *Dropout* di akhir *block* untuk mengatasi *overfitting*.

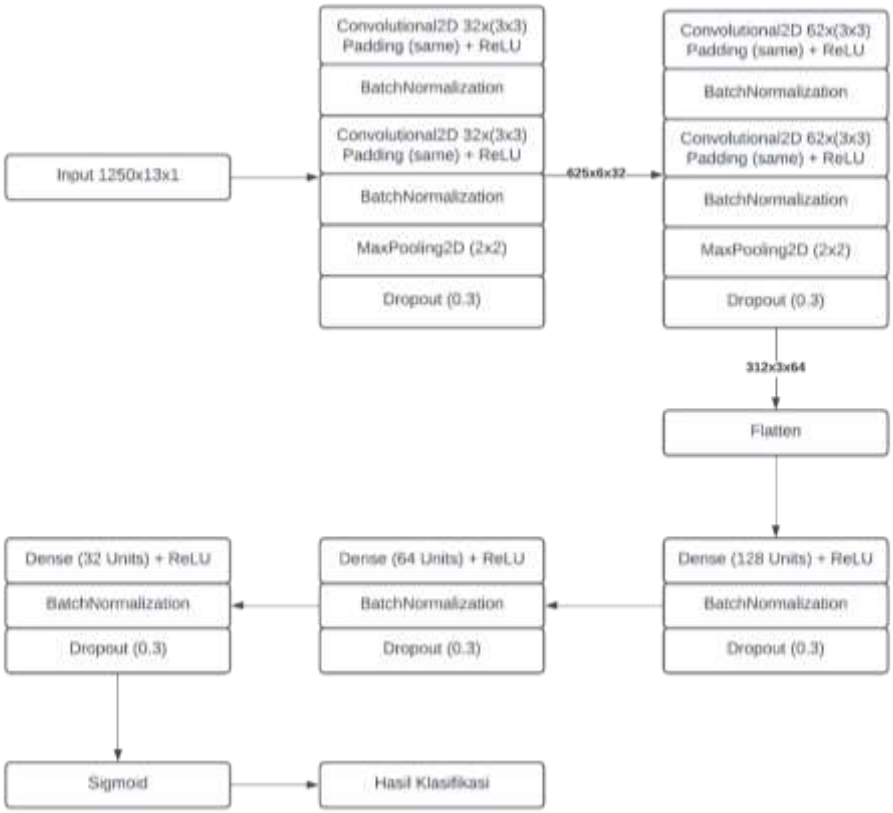
Lapisan *convolution* berfungsi untuk mengekstraksi fitur-fitur penting dari input, sedangkan lapisan *pooling* berperan mengurangi dimensi *feature map*. Setelah melewati lapisan-lapisan ini, serta lapisan *Dropout* yang berfungsi mencegah *overfitting*, *feature map* diubah menjadi *array* satu dimensi oleh lapisan *Flatten*. Lapisan *Flatten* ini bertindak sebagai jembatan antara lapisan *convolution* dan lapisan *Fully Connected*. Selanjutnya, lapisan *Fully Connected* akan memprediksi kelas berdasarkan input yang diterima. Pada tahap akhir, fungsi aktivasi *Sigmoid* digunakan untuk memprediksi probabilitas kelas *output*. Selain itu, *Batch Normalization* ditambahkan setelah lapisan tertentu untuk menstabilkan proses *training*, dan *Max Pooling* digunakan untuk mengurangi dimensi *feature map* pada lapisan berikutnya.

Arsitektur yang diusulkan menerima *input*-an dengan ukuran (1250,13,1) dengan tahapan data *audio preprocessing* yang diterapkan sebelumnya. *Flow diagram* dari arsitektur yang diusulkan dapat dilihat pada **Gambar 3.1**. Pada lapisan *convolutional block* pertama, operasi *convolution 2D* pada lapisan pertama dan yang kedua dilakukan dengan menggunakan ukuran filter (3,3) dan 32 filter yang berbeda. ReLU juga digunakan pada lapisan-lapisan ini sebagai fungsi aktivasi serta `Padding 'same'` agar menerima *output dimension* yang sama antar lapisan.

Pada setiap lapisan pada *convolutional block* pertama, terdapat lapisan *Batch Normalization* pada kedua lapisan *convolution*, dengan lapisan tersebut, kita telah mendapatkan *feature map* hasil dari kedua lapisan *convolution* tersebut yang diekstraksi dari fitur input. Namun, distribusi batch input bisa sangat bervariasi tergantung jenis fitur MFCC yang ada di dalamnya. Hal ini dapat menimbulkan masalah pada konvergensi algoritma pengoptimal, membuat proses *training* tidak stabil. Oleh karena itu, akan sangat membantu jika input untuk setiap layer adalah *unit gaussian*. Untuk mencapai hal ini, *feature map* ini dinormalisasi secara batch, yang menghasilkan proses *training* yang lebih cepat (*convergence* lebih cepat) dan mengurangi ketergantungan pada inisialisasi bobot.

Tensor hasil normalisasi *batch*, dengan ukuran (1250 x 13 x 32), selanjutnya diproses oleh lapisan *Max Pooling* berukuran (2,2) yang mereduksi ukuran input menjadi (625 x 6 x 32). Penggunaan lapisan *Max Pooling* ini didasari oleh penggunaan 32 filter pada model CNN yang lebih dalam untuk mengekstraksi fitur yang lebih kompleks. Namun, peningkatan jumlah filter ini juga meningkatkan dimensi *feature map*, yang berdampak pada peningkatan beban komputasi. Oleh karena itu, lapisan *pooling* digunakan untuk mengurangi dimensi *feature map*. Setelah mereduksi dimensi menjadi setengahnya, terdapat lapisan *Dropout* dengan nilai 0.3 yang secara acak menonaktifkan 30% dari unit/neuron *input*.

JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi



Gambar 2. *Flow Diagram* Arsitektur CNN

Setelah melewati *convolution block* pertama, yang menghasilkan *feature map* berukuran (625 x 6 x 32), data kemudian memasuki *convolution block* kedua. *Convolution block* kedua ini memiliki konfigurasi serupa dengan *block* pertama, terdiri dari dua lapisan *convolution* dengan aktivasi ReLU, masing-masing diikuti oleh lapisan *Batch Normalization*. Perbedaan utama antara kedua *block* terletak pada jumlah filter yang digunakan, yaitu 64 filter pada *block* kedua. Setelah lapisan *Batch Normalization* kedua, terdapat lapisan *Max Pooling* berukuran (2,2), dan lapisan *Dropout* dengan nilai 0.3 di akhir *block*. Hasil *feature map* dari *block* kedua ini berukuran (312 x 3 x 64).

Output dari *block* sebelumnya ditransformasikan ke dalam satu dimensional *array* menggunakan lapisan *Flatten*. Setelah itu, mengikuti yang dihasilkan *block* sebelumnya, terdapat sebuah lapisan *Fully Connected* dengan 128 neuron/unit dengan aktivasi ReLU, yang sekali lagi diikuti dengan lapisan *Batch Normalization* dan lapisan *Dropout* yang bernilai 0.3. Setelah melewati lapisan *Fully Connected* pertama, untuk yang kedua, lapisan yang digunakan yaitu lapisan *Fully Connected* yang memiliki 64 neuron/unit dengan aktivasi ReLU, yang diikuti dengan lapisan *Batch Normalization* dan lapisan *Dropout* yang bernilai 0.3. Setelah melewati lapisan *Fully Connected* pertama dan kedua, terdapat juga lapisan ketiga, lapisan yang digunakan yaitu lapisan *Fully Connected* yang memiliki 32 neuron/unit dengan aktivasi ReLU, yang diikuti dengan lapisan *Batch Normalization* dan lapisan *Dropout* yang bernilai 0.3. Akhirnya, terdapat sebuah lapisan *output* dengan dua neuron dan sebuah fungsi aktivasi *sigmoid*, yang mana akan memprediksi apakah audio fitur MFCC adalah AI (‘Fake’) atau bukan. Jika nilainya kurang dari 0,5 (50%), maka output yang diprediksi adalah ‘Fake’; jika tidak, itu adalah manusia (‘Real’).

Parameter dalam arsitektur yang dirancang memiliki total 7.744.898 parameter (**Tabel 3.3)**. Dari jumlah tersebut, 7.744.066 merupakan parameter yang dapat dilatih, artinya nilai-nilai parameter ini akan terus diperbarui selama proses *training* berlangsung. Tujuannya adalah untuk mengoptimalkan kinerja model agar mampu menghasilkan prediksi atau klasifikasi yang akurat. Di sisi lain, terdapat 832 parameter yang tidak dapat dilatih. Parameter-parameter ini memiliki nilai tetap yang tidak berubah selama proses *training*. Distribusi parameter pada setiap lapisan tidaklah seragam. Lapisan *Dense* memiliki jumlah parameter terbanyak. Hal ini disebabkan karena lapisan *Dense* bertanggung jawab melakukan transformasi linier pada data masukan yang berdimensi tinggi, yang mana dipengaruhi oleh masukan dari lapisan *Flatten*. Semakin tinggi dimensi data, semakin banyak pula parameter yang dibutuhkan untuk memetakan hubungan antara fitur-fitur input dengan target output.

JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

Tabel 3. *Output Dimension* dan Parameter untuk Setiap Lapisan

Layer	***Layer Type***	***Output Dimension***	***No. of Parameters***
1.	*Input, Convolution 2D*	(1250 x 13 x 32)	320
2	*Batch Normalization*	(1250 x 13 x 32)	128
3	*Convolution 2D*	(1250 x 13 x 32)	9248
4	*Batch Normalization*	(1250 x 13 x 32)	128
5	*Max Pooling 2D*	(625 x 6 x 32)	0
6	*Dropout*	(625 x 6 x 32)	0
7	*Convolution 2D*	(625 x 6 x 64)	18496
8	*Batch Normalization*	(625 x 6 x 64)	256
9	*Convolution 2D*	(625 x 6 x 64)	36928
10	*Batch Normalization*	(625 x 6 x 64)	256
11	*Max Pooling 2D*	(312 x 3 x 64)	0
12	*Dropout*	(312 x 3 x 64)	0
13	*Flatten*	(59904)	0
14	*Dense*	(128)	7667840
15	*Batch Normalization*	(128)	512
16	*Dropout*	(128)	0
17	*Dense*	(64)	8256
18	*Batch Normalization*	(64)	256
19	*Dropout*	(64)	0
20	*Dense*	(32)	2080
21	*Batch Normalization*	(32)	128
22	*Dropout*	(32)	0
23	*Dense*	(2)	66

Total Parameter	7.744.898
Trainable Parameter	7.744.066
Non-trainable Parameter	832

Lapisan *convolution* juga memiliki jumlah parameter yang signifikan. Lapisan ini berperan penting dalam mengekstraksi fitur-fitur penting dari data masukan. Setiap *filter* pada lapisan *convolution* memiliki bobot-bobot tersendiri yang akan dipelajari selama *training*. Semakin banyak *filter* yang digunakan, semakin banyak pula parameter yang dimiliki oleh lapisan ini. Berbeda dengan ketiga lapisan sebelumnya, lapisan *Batch Normalization*, lapisan *Max Pooling*, dan lapisan *Dropout* tidak memiliki parameter sama sekali. Lapisan-lapisan ini hanya melakukan operasi-operasi sederhana pada data, seperti normalisasi, pengurangan dimensi, dan penonaktifan neuron secara acak, tanpa melibatkan bobot atau bias yang perlu dilatih.

3.4 Konfigurasi Simulasi

Kami memanfaatkan penyimpanan data dalam format NPY untuk untuk menghemat waktu serta mengurangi proses komputasi. Format NPY merupakan format file standar biner yang digunakan oleh NumPy, *library* Python populer untuk komputasi numerik. Format ini dirancang untuk menyimpan secara efisien satu array NumPy tunggal ke dalam *storage disk*, dengan semua informasi yang diperlukan untuk merekonstruksi *array* tersebut dengan benar, termasuk bentuknya (dimensi), tipe data, dan data itu sendiri. Jadi, setelah dilakukannya proses *preprocessing*, data akan disimpan ke dalam format tersebut. Pada proses ini penyimpanan file *data training*, *validation*, dan *test* akan menggunakan Google Drive. Tujuan penyimpanan file di Google Drive agar dengan mudah diakses, yang mana selama melakukan *training* dan *testing*, kami menggunakan Google Colab. Setelah itu kami memanfaatkan kemampuan komputasi Google Colab Pro dengan GPU Tesla T4 serta NVIDIA L4 untuk mempercepat dan mempertahankan proses *training*.

3.5 Konfigurasi *Training*

Dalam proses *training*, *hyperparameter* yang akan digunakan pada penelitian ini adalah *optimizer* Adam dengan *learning rate* 0.0003 (3e-4) kemudian *batch size* berjumlah 64 serta jumlah iterasi atau *epoch* sebesar 100. Dikarenakan pemodelan yang akan dilakukan dengan *binary classification* (‘Fake’ dan ‘Real’), penelitian ini akan menggunakan *loss function* `binary_crossentropy`, yang khusus digunakan untuk kasus dengan konteks *binary class*. Setelah itu, *metric* yang digunakan adalah ‘accuracy’, yang merupakan proporsi prediksi yang benar di setiap kelasnya.

JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

Pada penelitian ini, sebelum melakukan proses *training*, konfigurasi *callback* akan dipakai. *Callback* sendiri merupakan utilitas atau fungsi khusus yang dijalankan selama proses *training* yang biasanya pada tahap-tahap tertentu dalam prosedur *training*. *Callback* juga merupakan alat canggih untuk mengatur perilaku model Keras selama *training*, evaluasi, atau inferensi. Dengan *callback*, dapat melakukan berbagai tindakan pada tahap-tahap tertentu, seperti menyimpan model terbaik, menghentikan *training* lebih awal jika performa tidak membaik, atau menyesuaikan parameter selama *training*. Ada banyak jenis *callback* yang disediakan oleh Tensorflow, akan tetapi untuk penelitian ini hanya menggunakan tiga jenis saja yaitu `ModelCheckpoint, EarlyStopping,` dan `ReduceLROnPlateu` dengan meng-*import* masing-masing jenis dari TensorFlow

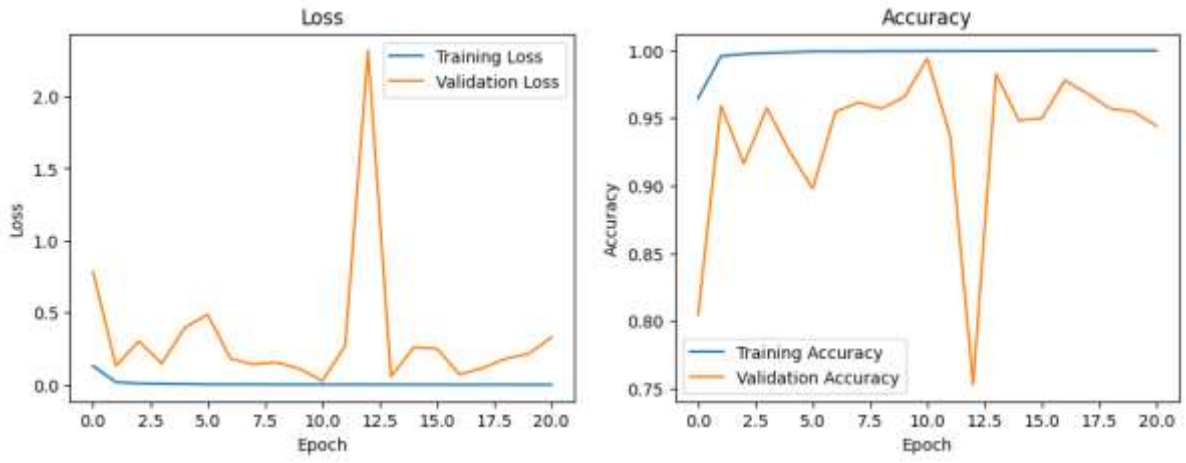
ModelCheckpoint merupakan sebuah *callback* yang berguna untuk menyimpan model terbaik kami. Setelah itu, ada data *validation* (val_accuracy) dengan parameter `monitor='val_accuracy'`. Kemudian, model terbaik dan model yang dianggap terbaik jika mencapai nilai `'val_accuracy'` tertinggi akan disimpan dengan parameter `save_best_only=True`. Kemudian, dikarenakan kita ingin memaksimalkan akurasi pada data *validation*, model diatur ke `'max'` dengan parameter `mode='max'`. Terakhir, *callback* ini akan mencetak pesan informastif setiap kali model akan disimpan dengan parameter `verbose=1.`

Setelah mengkonfigurasi ModelCheckpoint, *callback* selanjutnya adalah EarlyStopping. EarlyStopping merupakan salah satu *callback* yang paling banyak digunakan, yang mana biasanya digunakan untuk mencegah model mengalami *overfitting*. Pada penelitian ini, sama seperti ModelCheckpoint, EarlyStopping akan memantau akurasi pada data *validation* (val_accuracy) setelah setiap *epoch* dijalankan dengan parameter `monitor='val_accuracy'`. Setelah itu, perubahan minimum dalam akurasi pada data *validation* akan dipantau dengan parameter `min_delta=0` dan dalam hal ini, nilai 0 artinya setiap peningkatan, sekecil apapun, akan dianggap sebagai peningkatan. Kemudian, jumlah *epoch* yang diperbolehkan tanpa peningkatan dalam akurasi pada data *validation* adalah 10 dengan parameter `patience=10`, dan jika akurasi tidak meningkat selama 10 *epoch* berturut-turut, proses *training* akan dihentikan lebih awal secara otomatis. Sama dengan ModelCheckpoint, mode diatur ke `'max'` dan *callback* EarlyStopping akan mencetak pesan jika saat `early stopping` diaktifkan dengan parameter `verbose=1`.

Setelah mengkonfigurasi ModelCheckpoint dan EarlyStopping, *callback* yang digunakan selanjutnya, serta yang terakhir, adalah ReduceLROnPlateau. ReduceLROnPlateau merupakan jenis callback yang digunakan untuk mengubah laju pembelajaran (*learning rate*) ketika metrik atau monitor yang kita tuju berhenti meningkat. Seperti ModelCheckpoint dan EarlyStopping, ReduceLROnPlateau akan memantau akurasi pada data *validation* setelah setiap *epoch*. Factor yang mana digunakan untuk menurunkan *learning rate* (*learning rate* baru = *learning rate* lama * *factor*), dalam penelitian ini, learning rate akan dikalikan dengan 0.5, atau dibagi 2, setiap kali kondisi patience atau akurasi pada data *validation* tidak mengalami peningkatan dengan parameter `factor=0.5` dan `patience=3`. Sama seperti sebelumnya, untuk mode yang digunakan adalah `'max'` dan `verbose=1` untuk mencetak pesan ketika setiap kali *learning rate* dikurangi.

3.6 Hasil *Training*

Hasil proses *training* model dapat dilihat pada **Gambar 3.2**. Proses *training* berhenti pada *epoch* ke-21, meskipun awalnya diinisialisasi untuk berjalan hingga 100 *epoch*. Hal ini disebabkan oleh *callback* EarlyStopping, yang mana menghentikan proses *training* karena akurasi pada data *validation* tidak mengalami peningkatan selama 10 *epoch* terakhir. Penghentian dini ini bertujuan untuk mencegah terjadinya over*fitting* pada model. Model yang disimpan untuk proses *testing* terletak pada *epoch* ke-10 dengan *learning rate* yang sudah di-*reduce* dengan menggunakan *callback* ReduceLROnPlateau dengan berjumlah 0.00015 (15e-4) yang berhasil menghasilkan akurasi sebesar 0.99 **(99%)** pada data *training* serta 0.99 (**99%**) pada data *validation*.



JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

Gambar 3. *Accuracy* dan *Loss* selama masa *training* (*epoch*)

Terlihat pada grafik *loss* dan *accuracy* pada data *validation* menunjukkan kestabilan yang cukup mumpuni, yang mana untuk *loss*-nya rata-rata hanya direntang dibawah 1 dan untuk akurasinya direntang 0.90-1. Walaupun *trend*-nya terdapat fluktuasi disetiap *epoch*nya akan tetapi model ini sudah cukup bagus. Alasan yang paling mungkin menyebabkan fluktuasi ini adalah penggunaan *batch normalization* dan *dropout* layer pada arstitektur yang digunakan. Meskipun *accuracy* dan nilai kerugian (*loss*) pada *validation* model sedikit berfluktuasi, namun nilainya mengikuti pola *loss* pada data *training*. Ini menunjukkan bahwa model tidak mengalami over*fitting*. Dari **Gambar 3.2**, terlihat bahwa tidak ada perbedaan besar antara nilai *loss* data *training* dan data *validation*. Hal ini merupakan indikasi baik bahwa model bersifat umum dan tidak over*fitting*.

3.7 Metrik Evaluasi

Model yang kami usulkan berhasil mendapatkan *accuracy* sebesar 92% pada data *testing*, yang terdiri dari 7.448 audio 'Fake' dan 2.530 audio 'Real' yang sudah dilakukan tahap *preprocessing*. Seperti yang sudah terlihat pada sesi sebelumnya bahwa selama proses *training* pada data *validation* kinerja model relatif baik dan tidak ada tanda-tanda *overfitting*. Selain *accuracy*, nilai *precision*, *recall*, *F1-score*, *Equal Error Rate* (EER), dan *Precision-Recall Curve* (PRC) juga digunakan untuk mengevaluasi performa model. Bersamaan dengan metrik tersebut, kami menyajikan *confusion matrix* untuk memahami kemampuan klasifikasi model.

a. *Precision*: Metrik ini merupakan rasio *true positive* terhadap kelas positif yang diprediksi model. Secara harfiah metrik ini menunjukkan berapa banyak audio yang benar-benar 'Real' dari semua audio yang diprediksi sebagai 'Real' oleh model yang diusulkan. *True Positive* (TP) dalam konteks penelitian ini mengacu pada banyaknya jumlah data yang benar-benar 'Real' dan diprediksi sebagai 'Real'. Sebaliknya, *False Positive* (FP) mengacu pada banyaknya jumlah data yang sebenarnya 'Fake' tetapi diprediksi sebagai 'Real'.

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Hasil *precision* yang kami dapatkan pada data *testing* adalah 1.00 pada audio 'Fake' sedangkan untuk *precision* pada audio 'Real' adalah 0.76.

b. *Recall*: Metrik ini merupakan metrik ini merupakan rasio *true positive* terhadap kelas positif dalam *ground truth*. Metrik ini menunjukkan seberapa banyak audio yang diklasifikasikan sebagai 'Real' dari semua audio yang benar-benar 'Real'. *True Positive* (TP) mengacu pada banyaknya jumlah data yang benar-benar 'Real' dan diprediksi sebagai 'Real', sedangkan untuk *False Negative* (FN) mengacu pada banyaknya jumlah data yang sebenarnya 'Real' tetapi diprediksi sebagai 'Fake'.

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Hasil *recall* yang kami dapatkan pada data *testing* adalah 0.89 pada audio 'Fake' sedangkan untuk *precision* pada audio 'Real' adalah 1.00.

c. *F1-score*: metrik penting lain yang akan ditemui dalam tugas klasifikasi adalah F1 score. F1 score menggabungkan recall dan presisi, dan cara mendapatkannya dengan menghitung rata-rata harmonik dari presisi dan recall. Metrik ini mempertimbangkan *False Positives* dan *False Negative* sebagai pertimbangan.

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Hasil *F1-score* yang kami dapatkan pada data *testing* adalah 0.94 pada audio 'Fake' sedangkan untuk *precision* pada audio 'Real' adalah 0.86.

d. *Equal Error Rate* (EER): Metrik ini pada dasarnya menghitung proporsi prediksi yang salah terhadap total keseluruhan prediksi. Metrik ini menjumlahkan semua prediksi yang salah (FP + FN) setelah itu dibagi dengan penjumlahan semua prediksi (TP + TN + FN + FP).

$$ERR = \frac{FP + FN}{TP + TN + FN + FP}$$

Hasil EER yang kami dapatkan pada data *testing* adalah 0.08.

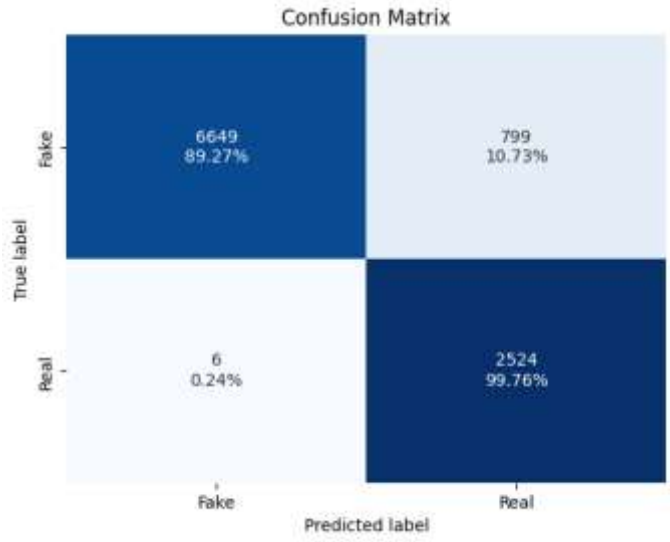
Tabel 4. *Classification Report*

(*100%)	Precision	Recall	F1-score	Support
Fake	1.00`	0.89	0.94	7448
Real	0.76	1.00	0.86	2530
Macro avg	0.88	0.95	0.90	9978
Weighted avg	0.94	0.92	0.92	9978
Accuracy			0.92	9978

JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

EER			0.08	9978

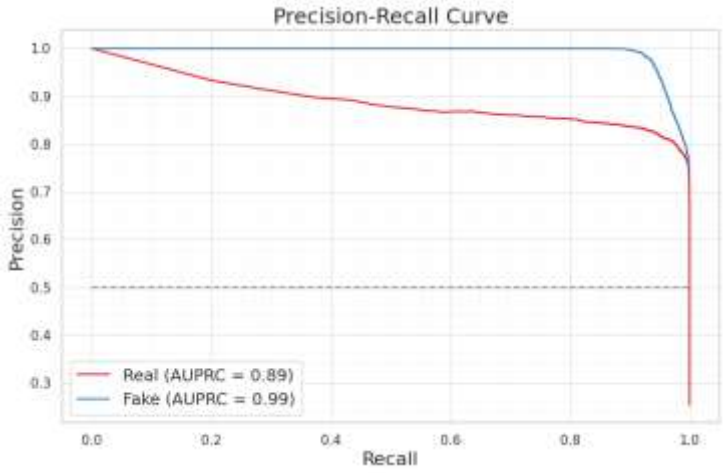
Classification report dari kinerja model kami pada data *testing* dapat dilihat pada **Tabel 3.3**. *accuracy*, *precision*, *recall*, *F1-score*, *macro average*, *weighted average*, dan *equal error rate* (EER) dapat dilihat pada tabel tersebut. Untuk mengevaluasi lebih lanjut kemampuan klasifikasi model yang diusulkan, kami menyajikan *confusion matrix* pada **Gambar 3.3.** *True label* mengacu pada label yang diberikan pada data yang sebenarnya, sementara itu untuk *predicted label* adalah hasil klasifikasi dari model kami. Kombinasi ini menghasilkan empat kategori: *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). TP dalam konteks penelitian ini mengacu pada banyaknya jumlah data yang benar-benar 'Real' dan diprediksi sebagai 'Real'. Setelah itu untuk TN mengacu pada banyaknya jumlah data yang benar-benar 'Fake' dan diprediksi sebagai 'Fake'. Selanjutnya untuk FP mengacu pada banyaknya jumlah data yang sebenarnya 'Fake' tetapi diprediksi sebagai 'Real'. Terakhir, untuk FN mengacu pada banyaknya jumlah data yang sebenarnya 'Real' tetapi diprediksi sebagai 'Fake'.



Gambar 4. *Confusion Matrix*

Dikarenakan data yang kami gunakan termasuk kedalam *class imbalance*, maka kami akan menggunakan metrik evaluasi yang bekerja baik dengan kondisi tersebut yang mana menggunakan metrik *Precision-Recall Curve* (PRC). Hasil PRC dapat dilihat pada **Gambar 4.** Hasil AUPRC pada kelas 'Real' menunjukkan kinerja yang cukup baik dengan nilai 0.89 **(89%)**. Model mampu mencapai keseimbangan yang layak antara *precision* dan *recall* untuk kelas tersebut. Bentuk kurva menunjukkan bahwa awalnya, *precision* tinggi ketika *recall* rendah. Hal tersebut menginterpretasikan bahwa ketika model sangat yakin tentang prediksi positifnya, prediksi tersebut cenderung benar. Saat *recall* mengalami peningkatan (model mencoba mengidentifikasi lebih banyak sampel positif), *precision* mulai menurun. Ini menunjukkan adanya *trade-off*: untuk menemukan lebih banyak sampel positif, model harus mengorbankan beberapa akurasi prediksi positifnya.

JURNAL SISTEM INFORMASI TGD
Volume 5, Nomor 3, Mei 2026, Hal 821-833
P-ISSN : 2828-1004 ; E-ISSN : 2828-2566
https://ojs.trigunadharma.ac.id/index.php/jsi

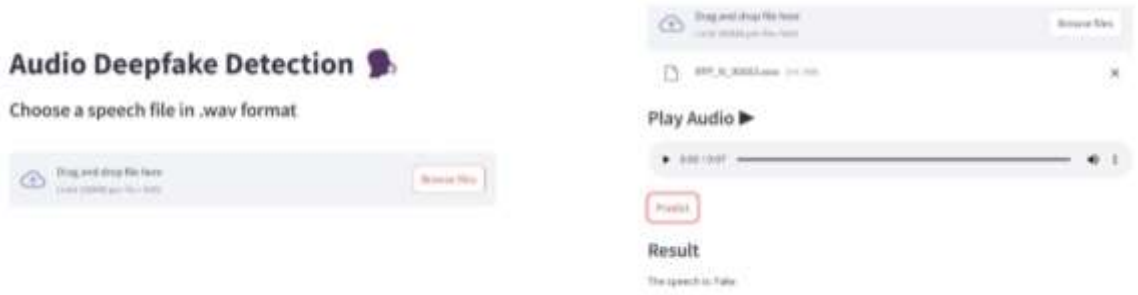


Gambar 5. *Precision-Recall Curve*

Sedangkan, hasil AUPRC pada kelas 'Fake' menunjukkan kinerja model yang sangat baik dengan nilai 0.99 **(99%)**. model sangat akurat dalam memprediksi atau mengidentifikasi sampel pada kelas tersebut. Bentuk kurva pada kelas tersebut menunjukkan bahwa kurva menetapkan ketinggian untuk sebagian rentang *recall*. Hal ini menginterpretasikan bahwa model mampu mempertahankan *precision* tinggi bahkan ketika mencoba mengidentifikasi lebih banyak sampel 'Fake', Adapun penurunan *precision* yang tajam hanya terjadi pada tingkat *recall* yang sangat tinggi, menunjukkan bahwa hanya ketika model mencoba untuk menjadi sangat inklusif dalam mengidentifikasi sampel 'Fake', ia mulai membuat beberapa kesalahan. Akan tetapi, secara keseluruhan sudah baik dan bekerja dengan maksimal dalam memprediksi audio yang dihasilkan AI.

3.8 Hasil Implementasi Sistem

Implementasi sistem yang dibuat menggunakan *library* Streamlit. Sistem yang dibuat dalam mendeteksi ucapan AI memiliki kelebihan dengan tampilan antarmuka pengguna (AI) yang sangat sederhana dan mudah dipahami serta digunakan. *User* hanya *User* hanya perlu mengunggah file audio WAV dan menekan tombol "Predict" untuk mendapatkan hasil prediksi. Hal ini membuat sistem mudah digunakan oleh pengguna yang tidak memiliki latar belakang teknis. Setelah itu, sistem menyediakan pemutar audio dan *progress bar* yang memungkinkan pengguna untuk mendengarkan audio yang diunggah. Fitur ini berguna untuk verifikasi dan analisis lebih lanjut. Selanjutnya, sistem yang dibuat ini menggunakan integrasi model *deep learning*, yang dihasilkan pada proses *training*¸ untuk mendeteksi ucapan yang dihasilkan oleh AI. Disamping kelebihannya, sistem ini memiliki kekurangan, yang mana hanya menerima file audio dalam format WAV. Hal ini dapat menjadi batasan jika pengguna memiliki file audio dalam format lain (MP3 dan FLAC).



(a) (b)

Gambar 6. (a) Tampilan *Upload*; (b) Tampilan Deteksi dan Hasilnya

4. KESIMPULAN

Bagian Hasil menunjukkan bahwa model arsitektur *Convolutional Neural Network* yang telah dibuat dapat menghasilkan akurasi untuk data *training* dan *validation* sebesar 99% dan untuk data *testing* sebesar 92%. Setelah itu, hasil evaluasi metrik (*precision*, *recall*, *F1-score*, *macro avg*, dan *weighted avg*) model pada data testing cukup memuaskan, yang mana untuk setiap metrik menghasilkan lebih dari 85%, akan tetapi untuk metrik *precision* pada kelas ‘Real’ hanya menghasilkan 76%, hal ini terjadi dikarenakan data yang cukup *imbalance* pada kelas ‘Real’ dan untuk

metrik EER berhasil mendapatkan 8% serta nilai AUPRC pada kelas 'Real' berhasil mendapatkan nilai 89% dan pada kelas 'Fake' berhasil mendapatkan nilai yang sangat memuaskan yaitu 99%. Terakhir, sistem yang dikembangkan dapat membedakan hasil suara *artificial intelligence* (*Fake*) dengan suara manusia (*Real*) yang berfungsi sebagai alat bantu untuk meningkatkan kemampuan membedakan ucapan yang dihasilkan AI. Sebagai saran untuk penelitian ke depannya, kami menyarankan agar dapat menggunakan metode *machine learning* atau *deep learning* serta ekstraksi fitur audio lainnya agar performanya dapat dibandingkan. Kemudian, dikarenakan data yang kami gunakan termasuk *imbalance* pada kelas 'Real', harapannya penelitian selanjutnya dapat menerapkan teknik *oversampling* maupun *undersampling* serta dapat menambahkan jumlah data agar lebih banyak lagi untuk setiap kelas dan dalam bahasa yang berbeda.

DAFTAR PUSTAKA

[1] I. J. Goodfellow *dkk.*, "Generative Adversarial Networks," 10 Juni 2014, *arXiv*: arXiv:1406.2661. Diakses: 4 Juni 2024. [Daring]. Tersedia pada: http://arxiv.org/abs/1406.2661

[2] D. P. Kingma dan M. Welling, "Auto-Encoding Variational Bayes," 10 Desember 2022, *arXiv*: arXiv:1312.6114. doi: 10.48550/arXiv.1312.6114.

[3] D. Dagar dan D. K. Vishwakarma, "A literature review and perspectives in deepfakes: generation, detection, and applications," *Int. J. Multimed. Inf. Retr.*, vol. 11, no. 3, hlm. 219–289, Sep 2022, doi: 10.1007/s13735-022-00241-w.

[4] C. Herke, "Deepfake," *Futuro*, 2023, [Daring]. Tersedia pada: https://api.semanticscholar.org/CorpusID:264316779

[5] A. Łańcucki, "FastPitch: Parallel Text-to-speech with Pitch Prediction," 16 Februari 2021, *arXiv*: arXiv:2006.06873. doi: 10.48550/arXiv.2006.06873.

[6] J. Kim, S. Kim, J. Kong, dan S. Yoon, "Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search," 22 Oktober 2020, *arXiv*: arXiv:2005.11129. doi: 10.48550/arXiv.2005.11129.

[7] J. Shen *dkk.*, "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions," 15 Februari 2018, *arXiv*: arXiv:1712.05884. doi: 10.48550/arXiv.1712.05884.

[8] B. van Niekerk, M.-A. Carbonneau, J. Zaïdi, M. Baas, H. Seuté, dan H. Kamper, "A Comparison of Discrete and Soft Speech Units for Improved Voice Conversion," 8 Juni 2022. doi: 10.1109/ICASSP43922.2022.9746484.

[9] LiuSongxiang, CaoYuewen, WangDisong, WuXixin, LiuXunying, dan MengHelen, "Any-to-Many Voice Conversion With Location-Relative Sequence-to-Sequence Modeling," *IEEEACM Trans. Audio Speech Lang. Process.*, 2021, doi: 10.1109/TASLP.2021.3076867.

[10] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, M. Kudinov, dan J. Wei, "Diffusion-Based Voice Conversion with Fast Maximum Likelihood Sampling Scheme," 4 Agustus 2022, *arXiv*: arXiv:2109.13821. doi: 10.48550/arXiv.2109.13821.

[11] J. Li, W. Tu, dan L. Xiao, "Freevc: Towards High-Quality Text-Free One-Shot Voice Conversion," dalam *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece: IEEE, Jun 2023, hlm. 1–5. doi: 10.1109/ICASSP49357.2023.10095191.

[12] D. Katanich, "It's a scam! How deepfakes and voice cloning taps into your cash," euronews. Diakses: 4 Juni 2024. [Daring]. Tersedia pada: https://www.euronews.com/business/2024/04/10/its-a-scam-how-deepfakes-and-voice-cloning-taps-into-your-cash

[13] D. Milmo, "Company worker in Hong Kong pays out £20m in deepfake video call scam," *The Guardian*, 5 Februari 2024. Diakses: 5 Juni 2024. [Daring]. Tersedia pada: https://www.theguardian.com/world/2024/feb/05/hong-kong-company-deepfake-video-conference-call-scam

[14] "CEO of world's biggest ad firm targeted by deepfake scam | Technology | The Guardian." Diakses: 5 Juni 2024. [Daring]. Tersedia pada: https://www.theguardian.com/technology/article/2024/may/10/ceo-wpp-deepfake-scam

[15] M. Cerullo, "AI scams mimicking voices are on the rise," CBS News. Diakses: 4 Juni 2024. [Daring]. Tersedia pada: https://www.cbsnews.com/news/ai-scam-voice-cloning-rising/

[16] A. Bunn, "Artificial Imposters—Cybercriminals Turn to AI Voice Cloning for a New Breed of Scam," McAfee Blog. Diakses: 4 Juni 2024. [Daring]. Tersedia pada: https://www.mcafee.com/blogs/privacy-identity-protection/artificial-imposters-cybercriminals-turn-to-ai-voice-cloning-for-a-new-breed-of-scam/

[17] Z. Wu *dkk.*, "ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," dalam *Interspeech 2015*, ISCA, Sep 2015, hlm. 2037–2041. doi: 10.21437/Interspeech.2015-462.

[18] X. Wang *dkk.*, "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech," 14 Juli 2020, *arXiv*: arXiv:1911.01601. doi: 10.48550/arXiv.1911.01601.

[19] J. Yamagishi *dkk.*, "ASVspoof 2021: accelerating progress in spoofed and deepfake speech detection," 1 September 2021, *arXiv*: arXiv:2109.00537. doi: 10.48550/arXiv.2109.00537.

[20] R. Reimao dan V. Tzerpos, "FoR: A Dataset for Synthetic Speech Detection," dalam *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, Timisoara, Romania: IEEE, Okt 2019, hlm. 1–10. doi: 10.1109/SPED.2019.8906599.

[21] H. Khalid, S. Tariq, M. Kim, dan S. S. Woo, "FakeAVCeleb: A Novel Audio-Video Multimodal Deepfake Dataset," 1 Maret 2022, *arXiv*: arXiv:2108.05080. doi: 10.48550/arXiv.2108.05080.
[22] "A dataset of histograms of original and fake voice recordings (H-Voice) - ScienceDirect." Diakses: 7 Juni 2024. [Daring]. Tersedia pada: https://www.sciencedirect.com/science/article/pii/S2352340920302250?via%3Dihub
[23] Z. Almutairi dan H. Elgibreen, "A Review of Modern Audio Deepfake Detection Methods: Challenges and Future Directions," *Algorithms*, vol. 15, no. 5, hlm. 155, Mei 2022, doi: 10.3390/a15050155.
[24] A. AlAli dan G. Theodorakopoulos, "An RFP dataset for Real, Fake, and Partially fake audio detection," 26 April 2024, *arXiv*: arXiv:2404.17721. doi: 10.48550/arXiv.2404.17721.