ANALISIS FORENSIK *DEEPFAKE* BERBASIS *CONVOLUTIONAL NEURAL NETWORK* (CNN) UNTUK DETEKSI INKONSISTENSI TEKSTUR DAN POLA PADA CITRA WAJAH

Toto Raharjo, Fikri Irfan Adristi, Frendi Yusroni Romadhona, Rosi Rahmadi Syahputra, Muhammad Yusuf Halim, Mikhail Ashshidiqie Rachman, Rohsan Nur Marjianto, Desylo Santicho, Permadi Kusuma, Erika Ramadhani
Informatika, Universitas Islam Indonesia
Jl. Kaliurang KM. 14,5 Kabupaten Sleman, Provinsi Daerah Istimewa Yogyakarta, Indonesia
fikri.adristi@students.uii.ac.id

ABSTRAK

Perkembangan teknologi *artificial intelligence* (AI) yang semakin pesat memunculkan teknologi *deepfake* yang menggunakan basis algoritma seperti *generative adversarial networks* (GANs) untuk memanipulasi citra wajah dengan tingkat realisme yang tinggi. Fenomena ini memunculkan tantangan dalam keamanan digital karena potensi penyalahgunaan, disinformasi, pelanggaran privasi, dan kejahatan dunia maya. Penelitian ini bertujuan mengembangkan model deteksi *deepfake* berbasis *convolutional neural networks* (CNN) yang mampu mengidentifikasi inkonsistensi tekstur dan pola biologis pada citra wajah secara efisien secara *resource hadware* dan efektif dalam proses deteksi *image*. Metode penelitian meliputi pengumpulan data dari *dataset* FaceForensics++ dan Celeb-DF, pra-proses data menggunakan augmentasi dan normalisasi, serta pengembangan model dengan arsitektur EfficientNetB0 yang dilatih menggunakan *transfer learning* pada *dataset* besar dengan menggunakan TensorFlow dan GPU. Hasil pengujian menunjukkan bahwa Model cenderung mengklasifikasikan gambar asli dengan nilai prediksi dari model *neural network* di atas 0,5, sementara gambar palsu justru dideteksi sebagai asli karena nilainya juga melebihi 0,5. Evaluasi menunjukkan bahwa model ini mencapai tingkat akurasi tinggi dalam mendeteksi manipulasi citra wajah, dengan teknik augmentasi yang meningkatkan kehandalan model terhadap berbagai skenario. Penelitian ini menyimpulkan bahwa pendekatan berbasis CNN efektif untuk deteksi deepfake, namun perlu pengembangan lebih lanjut untuk mengatasi keterbatasan pada data yang kompleks.

Kata kunci : *deepfake, convolutional neural networks, keamanan digital, generative adversarial networks, transfer learning.*

1. PENDAHULUAN

Perkembangan teknologi *artificial intelligence* (AI) telah membawa kemajuan signifikan dalam manipulasi citra digital khususnya teknologi *deepfake* yang mampu memanipulasi atau mengganti wajah seseorang dalam gambar atau video dengan tingkat realisme yang tinggi [1].

Peningkatan jumlah video *deepfake* yang beredar setiap tahunnya semakin mengkhawatirkan seperti sumsub, penyedia verifikasi siklus penuh global, mendeteksi peningkatan deepfake hingga 245% YoY di seluruh dunia [2].

Teknologi manipulasi *image* dan *video* deepfake menggunakan arsitektur *deep learning* terutama *generative adversarial network* (GANs) yang mampu menghasilkan gambar sintesis dengan kualitas yang semakin mendekati gambar asli [3].

Kemampuan GANs menghasilkan citra *image* palsu telah berkembang dengan pesat sehingga mata manusia kesulitan membedakan antara konten asli dan manipulasi. Penelitian yang dilakukan oleh Bray et al. [4] menunjukan bahwa akurasi deteksi manusia terhadap *deepfake* wajah hanya 62%, dengan variasi signifikan per gambar antara 85% hingga 30%, menunjukkan ketidakcukupan kemampuan deteksi meski keyakinan peserta tinggi.

Meskipun berbagai metode deteksi *deepfake* telah dikembangkan, tantangan utama masih terletak pada kemampuan untuk mengidentifikasi manipulasi yang semakin canggih [5].

Mengidentifikasi bahwa metode deteksi konvensional seringkali gagal dalam menganalisis inkonsistensi tekstur mikro dan pola biologis yang tidak natural, yang merupakan indikator kunci dalam identifikasi *deepfake* [5].

Karakteristik biologis manusia seperti pola kedipan mata, tekstur kulit, dan sinkronisasi bibir sering menunjukkan anomali yang dapat dideteksi melalui analisis mendalam.

Pendekatan berbasis *deep learning*, khususnya *convolutional neural networks* (CNN), menawarkan potensi besar dalam mendeteksi pola-pola manipulasi pada citra wajah. Seperti penelitian Wang et al. [6] menunjukkan bahwa arsitektur CNN yang dioptimalkan mampu mendeteksi artefak manipulasi dengan tingkat akurasi hingga 95% pada *dataset* standar. Sehingga dapat diidentifikasi masih terdapat kesenjangan signifikan dalam integrasi analisis forensik tradisional dengan pendekatan *deep learning*. Pentingnya pengembangan metode deteksi yang handal semakin dipertegas oleh meningkatnya kasus penyalahgunaan *deepfake* dalam konteks keamanan nasional dan privasi individual.

Disarikan dari Miller [7] bahwa perusahaan lambat dalam bertindak dalam menghadapi aksi *cybercrime* seperti *deepfake* yang mana persentasenya

46%-nya tidak mempunyai langkah dan perencanaan, 29%-nya telah mengambil langkah, dan 25%-nya baru merencanakan dalam mengambil langkah. Dalam konteks kriminologi, pencurian identitas berbasis *deepfake* mencerminkan ancaman baru dalam kejahatan dunia maya yang semakin sulit diatasi. Kejahatan ini umumnya melibatkan manipulasi teknologi untuk menciptakan konten palsu, seperti video atau audio yang menyerupai wajah, suara, atau data pribadi korban, dengan tujuan untuk menipu pihak lain [8].

Hal ini menunjukkan urgensi untuk mengembangkan sistem deteksi yang yang tidak hanya akurat, tetapi juga dapat diimplementasikan secara praktis dalam sistem keamanan digital.

Tujuan penelitian ini adalah mengembangkan sistem deteksi *deepfake* berbasis algoritma *convolutional neural networks* (CNN) yang akurat dan praktis untuk diimplementasikan dalam sistem keamanan digital, guna meningkatkan perlindungan terhadap ancaman siber.

2. TINJAUAN PUSTAKA

Deepfake adalah teknologi berbasis *artificial intelligence* (AI) yang memanfaatkan algoritma *deep learning*, seperti *generative adversarial networks* (GAN), untuk menciptakan manipulasi visual dan audio yang sangat realistis [9].

Teknologi *deepfake* juga memanfaatkan data berupa wajah dari individu yang merupakan bagian dari data pribadi dan berpotensi untuk disalah gunakan, baik itu untuk tindakan kejahatan seperti, propaganda, pornografi, pencurian identitas ataupun isu privasi terkait lainnya [10].

Tantangan dalam deteksi *deepfake* terletak pada tingkat realisme yang semakin tinggi, minimnya dataset untuk pelatihan model deteksi, dan evolusi cepat teknologi manipulasi ini. Oleh karena itu, diperlukan pendekatan forensik digital yang mampu mengatasi tantangan tersebut.

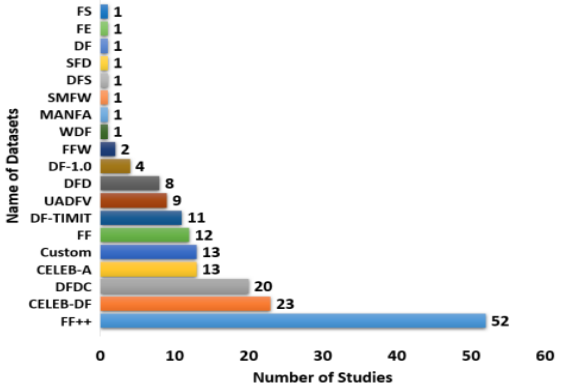
Salah satu pendekatan yang telah banyak dikembangkan adalah penggunaan *convolutional neural networks* (CNN), yang terbukti efektif dalam menganalisis data visual. Pada tahun 2022, metode ini digunakan dalam 77% penelitian terkait dibandingkan dengan metode lainnya [11].

CNN dapat mengekstraksi fitur spasial dari gambar dan mengidentifikasi artefak kecil yang ditinggalkan oleh teknologi *deepfake*, seperti ketidaksesuaian tekstur atau pola pencahayaan. Berbagai arsitektur CNN, seperti MesoNet dan model berbasis *transfer learning*, telah dikembangkan untuk meningkatkan akurasi deteksi [12].

Selain itu, dataset berkualitas tinggi, seperti FaceForensics++ [13] dan Celeb-DF [14], memainkan peran penting dalam melatih model untuk mengenali manipulasi *deepfake*.

Penelitian sebelumnya telah menggunakan berbagai jenis model dataset. Dalam beberapa kasus, penelitian dilakukan dengan memanfaatkan dataset yang dikembangkan secara mandiri, yang mana manipulasi dilakukan pada video atau gambar untuk menciptakan variasi data *deepfake* yang lebih beragam. Diketahui dari sekian banyak dataset yang telah dibuat dan diuji coba, terdapat beberapa dataset paling populer, yaitu FaceForensics++, Celeb-DF, dan DFDC [14].

Dataset ini menjadi populer karena memiliki keragaman data yang cukup baik dalam pengujian berbagai metode deteksi deepfake [11].



Gambar 1. Data penggunaan dataset dalam penelitian *deepfake* [11]

Penelitian terdahulu juga menjelaskan tentang *deepfake*, yakni teknologi berbasis algoritma *deep learning* yang digunakan untuk menciptakan wajah manusia yang sangat realistis melalui manipulasi video atau citra. Teknologi ini semakin berkembang dengan hadirnya metode seperti *autoencoder* dan *generative adversarial networks* (GANs) yang memungkinkan manipulasi konten wajah secara efektif. Kemampuan untuk menghasilkan video palsu dengan sedikit artefak visual menjadikan *deepfake* sebagai tantangan utama dalam forensik digital, khususnya untuk mencegah penyebaran konten palsu yang dapat digunakan dalam tindakan kriminal seperti pemerasan atau disinformasi politik [15].

Dalam upaya mendeteksi *deepfake*, terdapat penelitian terdahulu yang mengembangkan metode berbasis jaringan saraf seperti *VGG convolutional neural networks* yang telah ditingkatkan dengan penambahan filter SRM untuk mendeteksi *noise* gambar yang sulit terlihat. Model yang dikenal sebagai NA-VGG menggunakan fitur *noise* ini untuk melatih sistem dalam mengenali gambar manipulasi, bahkan dalam kondisi di mana fitur wajah sulit diidentifikasi secara langsung. *Dataset* seperti Celeb-DF digunakan untuk menguji efektivitas metode ini dan menunjukkan peningkatan deteksi yang signifikan dibandingkan metode lain [16].

Pada penelitian terkait seperti Tamtama & Senapartha [17] juga membahas mengenai *deepfake* yaitu teknologi dibangun menggunakan arsitektur MobileNets untuk mendeteksi wajah palsu. Dengan adanya metode verifikasi wajah asli, maka keamanan sistem bisa ditingkatkan dan meminimalisir penyalahgunaan.

3. METODE PENELITIAN

Penelitian ini bertujuan untuk mengembangkan dan mengevaluasi model deteksi deepfake berbasis deep learning dengan pendekatan sistematis. Metod penelitian yang digunakan mencakup beberapa tahap utama: pengumpulan data, pra-proses data, pengembangan model, pelatihan, dan evaluasi.

a. Pengumpulan data: Data yang digunakan dalam penelitian ini berasal dari direktori prepared_dataset di repositori DeepFake-Detect pada akun Github aaronchong888 [18].
b. Pra-proses data mencakup: (1) Ekstraksi *frame*: Video dipecah menjadi frame individu untuk analisis berbasis citra; (2) deteksi wajah: Area wajah pada setiap *frame* diidentifikasi menggunakan algoritma seperti MTCNN *(multi-task cascaded convolutional networks)*.; (3) normalisasi ukuran: Gambar wajah diseragamkan ke dimensi tertentu, misalnya 224x224 piksel; dan (4) Augmentasi data: Teknik augmentasi seperti rotasi, *flipping*, penyesuaian kecerahan, dan penambahan noise diterapkan untuk meningkatkan variasi data pelatihan dan mengurangi *overfitting*.
c. Pengembangan model: Model deteksi *deepfake* menggunakan arsitektur *convolutional neural networks* (CNN), dengan konfigurasi sebagai berikut: (1) Arsitektur model: menggunakan model berbasis EfficientNet atau XceptionNet yang telah terbukti efektif dalam deteksi manipulasi citra; (2) *Transfer learning*: Model *pre-training* pada *dataset* ImageNet dimodifikasi untuk tugas deteksi *deepfake*. Metode *convolutional neural networks* (CNN) adalah algoritma *deep learning* yang umum digunakan dalam berbagai aplikasi. CNN sering digunakan untuk klasifikasi gambar, segmentasi, deteksi objek, pemrosesan video, pemrosesan bahasa alami, dan pengenalan suara. CNN memiliki empat *layer*: *convolution layer, pooling layer, fully connected layer*, dan *non-linear layer*. *Convolution layer* menggunakan filter kernel untuk menghitung *convolution* gambar input dengan mengekstraksi fitur-fitur mendasar [19].
d. Pelatihan model *(training model)*: (1) Dataset dibagi menjadi tiga subset: pelatihan (70%), validasi (15%), dan pengujian (15%); (2) Proses *training* dilakukan menggunakan *framework* seperti TensorFlow atau PyTorch [20] dengan bantuan GPU untuk mempercepat komputasi: serta (3) penggunaan *callback* seperti EarlyStopping dan ModelCheckpoint untuk mencegah *overfitting* dan menyimpan model terbaik.
e. Evaluasi Model: Model yang telah dilatih dievaluasi menggunakan dataset uji yang mencakup berbagai kondisi data. Metrik evaluasi meliputi: (1) Akurasi: Tingkat keberhasilan model dalam mengklasifikasikan gambar sebagai real atau fake; (2) *precision, recall, dan F1-Score:* Guna mengukur performa model dalam mendeteksi *deepfake*; (3) ROC-AUC *(receiver operating characteristic - area under curve)*: Guna mengevaluasi kemampuan model membedakan antara gambar autentik dan manipulasi.
f. Eksperimen Tambahan: Guna meningkatkan akurasi deteksi, dilakukan eksperimen tambahan seperti: (1) *Ensemble learning:* Menggabungkan prediksi dari beberapa model untuk meningkatkan kinerja; (2) analisis kinerja pada variasi data: Menguji model pada data dengan resolusi rendah atau pencahayaan ekstrem untuk mengevaluasi robustitasnya.
g. Analisis hasil: Hasil evaluasi model dianalisis untuk memahami kelebihan dan kekurangannya. Performa model dibandingkan dengan penelitian sebelumnya untuk menilai kontribusi penelitian ini terhadap deteksi *deepfake*.

4. HASIL DAN PEMBAHASAN

Analisis ini mengimplementasikan model klasifikasi gambar menggunakan arsitektur EfficientNetB0, bagian dari keluarga CNN yang dikembangkan oleh Google AI. Keunggulan utama EfficientNetB0 terletak pada efisiensi komputasi dan akurasi melalui teknik *"compound scaling"*, yang menyesuaikan kedalaman, lebar, dan resolusi model secara proporsional. Arsitektur ini lebih kecil dibandingkan model CNN lain namun tetap memberikan performa tinggi berkat penggunaan MBConv dan penerapan *transfer learning* dengan dataset besar seperti ImageNet. Model dikembangkan dengan membagi dataset menjadi training, validasi, dan pengujian, serta menerapkan augmentasi data pada set pelatihan. Proses klasifikasi dilakukan dengan menambahkan lapisan *dense* dan *dropout*, dikompilasi menggunakan *Adam optimizer* dan *binary crossentropy loss*, serta memanfaatkan *EarlyStopping* dan *ModelCheckpoint* untuk mencegah *overfitting*. Setelah *training*, model terbaik digunakan untuk membuat prediksi pada data uji, dengan hasil yang disimpan dalam *dataframe* untuk analisis lebih lanjut.

4.1. *Import Library*

Kode dimulai dengan mengimpor library yang diperlukan, termasuk *library* untuk pengolahan data seperti json, os, dan pandas, serta *library* tensorflow dan keras untuk pengembangan model *deep learning*. library ini menyediakan fungsi dan kelas yang diperlukan untuk membangun dan melatih model klasifikasi gambar.

```
import                                  json
import                                    os
from distutils.dir_util import copy_tree
import                                 shutil
import         pandas         as          pd
import       tensorflow        as        tf
from tensorflow.keras import backend as K
```

4.2. Pengaturan *Workspace*

Setelah mengimpor *library*, kode mencetak versi TensorFlow yang digunakan. Langkah ini penting untuk memastikan bahwa *workspace* pengembangan telah diatur dengan benar dan kompatibel dengan kode yang akan dijalankan.

```
print('Versi TensorFlow: ', tf.__version__)
```

4.3. Pengaturan *Dataset*

Jalur ke dataset yang telah dibagi menjadi tiga bagian—pelatihan *(training)*, validasi, dan pengujian—ditentukan. Kode juga membuat direktori sementara untuk keperluan *debugging*, yang memungkinkan pengguna untuk menyimpan dan mengelola *file* sementara selama proses pengembangan.

```
dataset_path = 'split_dataset'
tmp_debug_path = 'tmp_debug'
print('Membuat Direktori: ' + tmp_debug_path)
os.makedirs(tmp_debug_path, exist_ok=True)
```

4.4. Fungsi untuk Mengambil Nama *File*

Sebuah fungsi didefinisikan untuk mengambil nama *file* tanpa ekstensi. Fungsi ini berguna untuk analisis hasil prediksi, memungkinkan pengguna untuk mengidentifikasi gambar berdasarkan nama *file* yang relevan.

```
def get_filename_only(file_path):
    file_basename = os.path.basename(file_path)
    filename_only = file_basename.split('.')[0]
    return filename_only
```

4.5. Augmentasi Data

Augmentasi data dilakukan dengan menggunakan *ImageDataGenerator*, yang memungkinkan penerapan berbagai teknik augmentasi pada gambar pelatihan. Teknik ini mencakup rotasi, pergeseran, pemotongan, dan pembesaran gambar, yang bertujuan untuk meningkatkan keragaman dataset pelatihan dan membantu model belajar lebih baik.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

4.6. Pengaturan *Generator Data*

Generator data dibuat untuk masing-masing set data (pelatihan, validasi, dan pengujian). Pada data pelatihan *(training)*, augmentasi diterapkan, sedangkan untuk data validasi dan pengujian, hanya *rescaling* dilakukan untuk mengubah nilai piksel ke rentang [0, 1].

```
train_datagen = ImageDataGenerator(
    rescale=1/255,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
```

4.7. Membangun Model CNN

Model CNN dibangun dengan memanfaatkan arsitektur EfficientNetB0 yang telah dilatih sebelumnya pada dataset ImageNet. Model ini tidak menyertakan lapisan atas *(fully connected layers)* dan menggunakan *pooling* maksimum untuk mengurangi dimensi fitur yang dihasilkan.

```
train_generator = train_datagen.flow_from_directory(
    directory=train_path,
    target_size=(input_size, input_size),
    color_mode="rgb",
    class_mode="binary",
    batch_size=batch_size_num,
    shuffle=True)
```

4.8. Menambahkan Lapisan ke Model

Setelah memuat EfficientNetB0, beberapa lapisan dense dan *dropout* ditambahkan untuk klasifikasi biner. Lapisan *output* menggunakan fungsi aktivasi sigmoid untuk menghasilkan probabilitas yang menunjukkan kelas gambar.

```
val_datagen = ImageDataGenerator(rescale=1/255)
val_generator = val_datagen.flow_from_directory(
    directory=val_path,
    target_size=(input_size, input_size),
    color_mode="rgb",
    class_mode="binary",
    batch_size=batch_size_num,
    shuffle=True
```

4.9. Kompilasi Model

Model dikompilasi dengan menggunakan *Adam optimizer* dan fungsi *loss binary crossentropy*, serta metrik akurasi untuk evaluasi. Langkah ini penting untuk mempersiapkan model sebelum proses pelatihan dimulai.

```
efficient_net = EfficientNetB0(
    weights='imagenet',
    input_shape=(input_size, input_size, 3),
    include_top=False,
    pooling='max'
)
```

4.10. Pengaturan *Callback* untuk *Training*

Callback EarlyStopping dan *ModelCheckpoint* diatur untuk menghentikan pelatihan jika tidak ada perbaikan dalam validasi *loss* dan untuk menyimpan

model terbaik berdasarkan kinerja pada data validasi. Ini membantu dalam mencegah *overfitting* dan memastikan bahwa model yang disimpan adalah yang paling optimal.

```
    model = Sequential()
    model.add(efficient_net)
    model.add(Dense(units=512,
activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(units=128,
activation='relu'))
    model.add(Dense(units=1,
activation='sigmoid'))
    model.summary()
```

4.11. *Model Training*

Model di-*training* menggunakan data pelatihan dan validasi selama sejumlah *epoch* yang ditentukan. Proses pelatihan ini melibatkan pembaruan bobot model berdasarkan data yang diberikan dan evaluasi kinerja model pada data validasi.

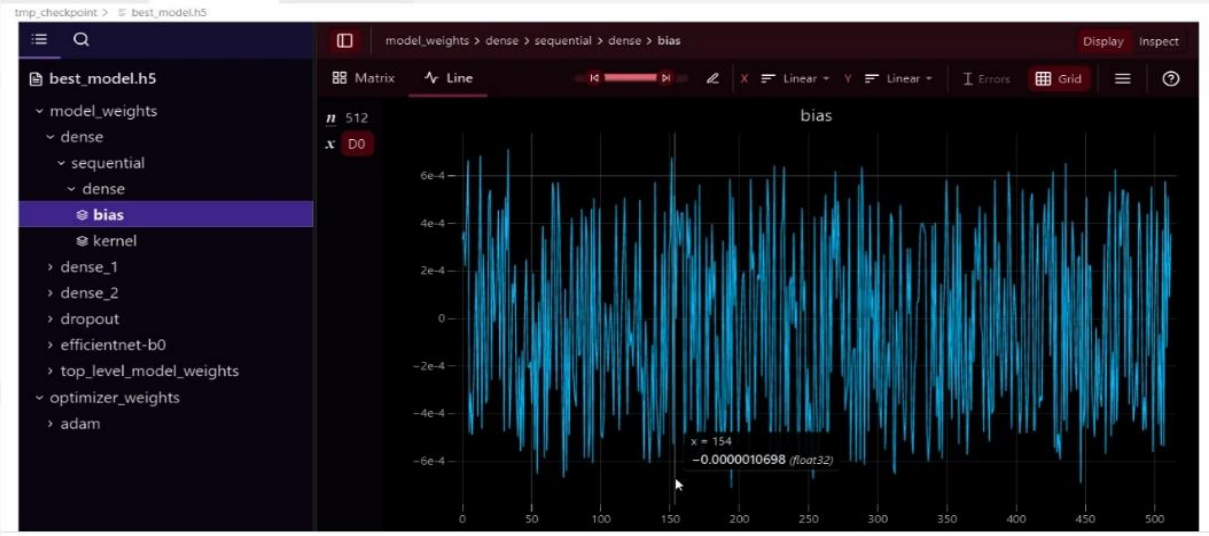
```
    model.compile(optimizer=Adam(learning_r
ate=0.0001), loss='binary_crossentropy',
metrics=['accuracy'])
```

4.12. Memuat Model Terbaik

Setelah pelatihan selesai, model terbaik yang disimpan selama proses *training* dimuat untuk digunakan dalam menghasilkan prediksi pada *dataset* pengujian. Hal ini memastikan bahwa model yang digunakan untuk evaluasi adalah yang memiliki kinerja terbaik.

```
custom_callbacks = [
        EarlyStopping(monitor='val_loss',
mode='min', patience=5, verbose=1),

ModelCheckpoint(filepath=os.path.join(check
point_filepath, 'best_model.h5'),
monitor='val_loss', mode='min', verbose=1,
save_best_only=True)
    ]
```



Gambar 4. Best model h.5

File best_model.h5 menyimpan elemen-elemen penting, seperti model_weights (bobot terlatih), lapisan *dense (fully connected)*, *dropout* (mencegah *overfitting*), arsitektur EfficientNet-B0, optimizer_weights, serta algoritma optimasi Adam untuk meningkatkan performa model.

4.13. Menghasilkan Prediksi

Model yang telah dimuat digunakan untuk menghasilkan prediksi pada data pengujian. Proses ini melibatkan pemrosesan gambar pengujian dan penerapan model untuk menentukan kelas gambar.

```
    history = model.fit(
        train_generator,
        epochs=num_epochs,

steps_per_epoch=len(train_generator),
        validation_data=val_generator,

validation_steps=len(val_generator),
        callbacks=custom_callbacks
    )
```

4.14. Menyimpan Hasil Prediksi

Hasil prediksi disimpan dalam *DataFrame* yang mencakup nama *file* gambar dan nilai prediksi. *DataFrame* ini kemudian dicetak untuk analisis lebih lanjut, memungkinkan pengguna untuk melihat hasil klasifikasi yang dihasilkan oleh model.

```
 best_model =
load_model(os.path.join(checkpoint_filepath
, 'best_model.h5'))
```

Model: "sequential"

Layer (type)	Output Shape	Param #
efficientnet-b0 (Functional)	(None, 1280)	4,049,564
dense (Dense)	(None, 512)	655,872
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 128)	65,664
dense_2 (Dense)	(None, 1)	129

Total params: 4,771,229 (18.20 MB)
Trainable params: 4,729,213 (18.04 MB)
Non-trainable params: 42,016 (164.12 KB)

Gambar 5. *Data training*

Model yang ditampilkan adalah arsitektur *neural network* bertipe *"sequential"*, yang terdiri dari beberapa lapisan yang ditambahkan secara berurutan. Lapisan utama adalah EfficientNet-b0, yang menghasilkan 1280 fitur, diikuti oleh lapisan *dense* yang menghasilkan 512 fitur. Terdapat juga lapisan *dropout* untuk mengurangi *overfitting*, serta dua lapisan *dense* tambahan yang masing-masing menghasilkan 128 dan 1 fitur. Total parameter dalam model ini mencapai 4,771,229, dengan 4,729,213 parameter dapat dilatih dan 42,016 parameter *non-trainable*.

Gambar di atas menunjukkan hasil prediksi dari model yang telah di-*training*, dengan kolom yang mencakup nama file gambar dan nilai prediksi untuk masing-masing gambar. Setiap entri mencakup dua kategori: *"real"* dan *"fake"*, dengan nilai prediksi yang menunjukkan probabilitas bahwa gambar tersebut termasuk dalam kategori tertentu. Nilai prediksi berkisar antara 0 hingga 1, dimana nilai yang lebih tinggi menunjukkan keyakinan model yang lebih besar terhadap klasifikasi tersebut. Hasil ini penting untuk evaluasi kinerja model dalam membedakan antara gambar asli dan palsu.

Hasil prediksi dari model *neural network* menunjukkan nilai Pada terhadap klasifikasi gambar yang dianalisis. Pada gambar pertama, yaitu "real/abarnvbtwb-000-00.png", model memberikan prediksi dengan nilai 0,663287; menunjukkan tingkat kepercayaan yang cukup tinggi. Gambar kedua, "real/abarnvbtwb-002-00.png", memiliki prediksi sebesar 0,465302; yang menunjukkan ketidakpastian lebih besar dalam klasifikasi. Selanjutnya, untuk gambar "fake/aagfhtgpmv-009-00.png", model memberikan prediksi 0,650603; menunjukkan keyakinan yang moderat. Terakhir, gambar "fake/aapnvogymq-004-01.png" memperoleh prediksi 0,674296; yang menunjukkan kepercayaan yang lebih tinggi terhadap klasifikasi tersebut.

Nilai di atas 0,5 menunjukkan bahwa model cenderung mengklasifikasikan gambar tersebut sebagai "positif" atau "benar" (misalnya, gambar asli). Nilai di bawah 0,5 Menunjukkan bahwa model cenderung mengklasifikasikan gambar tersebut sebagai "negatif" atau "salah" (misalnya, gambar palsu). Berdasarkan hasil yang ujicoba:

a. real/abarnvbtwb-000-00.png: 0,663287 (di atas 0,5; kemungkinan besar asli)
b. real/abarnvbtwb-002-00.png: 0,465302 (di bawah 0,5; kemungkinan besar palsu)
c. fake/aagfhtgpmv-009-00.png: 0,650603 (di atas 0,5; kemungkinan besar asli)
d. fake/aapnvogymq-004-01.png: 0,674296 (di atas 0,5; kemungkinan besar asli)

4.15. Kendala dalam Menjalankan Skrip *Machine Learning* pada Perangkat dengan Spesifikasi Rendah

Kendala utama yang dihadapi saat menjalankan skrip *machine learning* pada perangkat dengan spesifikasi rendah, seperti RAM 1GB dan CPU 1 core, mencakup beberapa aspek kritis. Pertama, keterbatasan memori sering kali menyebabkan kehabisan memori, di mana model dan *dataset* yang besar tidak dapat dimuat sepenuhnya, sehingga mengakibatkan program *crash* atau kesulitan dalam memproses data. Selain itu, penggunaan *swap memory* yang lebih lambat daripada RAM dapat memperlambat pengolahan data. Kedua, keterbatasan pada prosesor dengan hanya satu *core* mengakibatkan waktu pelatihan yang sangat lama, terutama untuk algoritma yang kompleks, serta mengurangi kemampuan untuk memanfaatkan paralelisasi yang dapat mempercepat proses. Ketiga, ukuran *dataset* yang besar menjadi tantangan tersendiri, karena kesulitan dalam memuat dan memproses data secara efisien dapat menciptakan *bottleneck* dalam alur kerja.

Keempat, keterbatasan dalam penggunaan algoritma yang lebih canggih memaksa pengguna untuk beralih ke metode yang lebih sederhana, yang mungkin kurang efektif. Selain itu, proses validasi model yang memerlukan beberapa iterasi pelatihan menjadi sulit dilakukan, sehingga mengurangi akurasi evaluasi model. Keterbatasan ini juga berdampak pada kemampuan untuk melakukan eksperimen yang diperlukan untuk mengoptimalkan model, seperti

tuning hyperparameter, serta menyulitkan analisis hasil melalui visualisasi data yang memerlukan sumber daya lebih besar. Secara keseluruhan, tantangan-tantangan ini menuntut pendekatan yang cermat dan penggunaan teknik yang dapat mengoptimalkan pemanfaatan sumber daya yang tersedia.

4.16. Solusi untuk Mengatasi Kendala Sumber Daya

Guna mengatasi kendala sumber daya saat menjalankan skrip *machine learning* pada perangkat dengan spesifikasi rendah, beberapa solusi dapat diterapkan. Pertama, penggunaan model yang lebih ringan, seperti regresi linier atau *random forest*, dapat mengurangi kebutuhan memori dan waktu pemrosesan.

Selain itu, pengurangan ukuran dataset melalui teknik sampling atau pengurangan dimensi seperti PCA dapat membantu mengelola beban memori. Implementasi pemrosesan *batch*, di mana data dibagi menjadi subset yang lebih kecil, juga dapat meningkatkan efisiensi. Mengoptimalkan kode skrip untuk meningkatkan efisiensi, serta memanfaatkan layanan *cloud computing* untuk akses ke sumber daya yang lebih besar, merupakan langkah-langkah yang efektif.

Teknik *transfer learning*, dimana model yang sudah dilatih sebelumnya digunakan sebagai dasar, dapat mengurangi waktu dan sumber daya yang diperlukan untuk pelatihan. Selain itu, memilih *framework machine learning* yang dirancang untuk efisiensi sumber daya, seperti TensorFlow Lite [20], dan melakukan *monitoring* serta *profiling* pada penggunaan sumber daya selama eksekusi skrip dapat membantu mengidentifikasi *bottleneck* dan mengoptimalkan bagian-bagian yang memerlukan perbaikan.

4.17. Penerapan *Error Level Analysis* (ELA) untuk Deteksi Manipulasi Gambar

Adapun *error level analysis* (ELA) dapat menjadi teknik alternatif yang efektif untuk mendeteksi manipulasi gambar pada perangkat dengan spesifikasi rendah. ELA bekerja dengan menganalisis tingkat kompresi pada berbagai bagian gambar untuk mengidentifikasi perbedaan yang mungkin menunjukkan adanya modifikasi. Teknik ini tidak memerlukan model *machine learning* yang besar atau kompleks, sehingga lebih ringan dari segi kebutuhan memori dan prosesor [21].

Dengan menggunakan alat seperti yang disediakan oleh Fake Image Detector [22], ELA dapat diterapkan pada gambar dalam format yang didukung, seperti JPEG, untuk mengevaluasi keasliannya. Analisis ini menghasilkan visualisasi tingkat kesalahan dalam kompresi, yang dapat digunakan untuk mendeteksi pola-pola anomali yang menunjukkan manipulasi. ELA sangat cocok untuk perangkat dengan spesifikasi rendah karena prosesnya bersifat langsung dan dapat dilakukan tanpa memerlukan pelatihan model tambahan, memungkinkan pengguna untuk tetap mendapatkan hasil yang akurat tanpa memerlukan sumber daya komputasi besar.

5. KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa model *convolutional neural networks* (CNN), khususnya EfficientNetB0, mampu mendeteksi *deepfake* dengan akurasi tinggi dalam mengidentifikasi gambar asli, dengan contoh prediksi seperti real/abarnvbtwb-000-00.png sebesar 0,663287 (kemungkinan besar asli). Namun, model masih mendeteksi beberapa gambar palsu sebagai asli, seperti fake/aagfhtgpmv-009-00.png yang diprediksi 0,650603 (kemungkinan besar asli). Penggunaan teknik augmentasi data, normalisasi, *transfer learning*, dan pengurangan dimensi terbukti meningkatkan performa model, meskipun masih ada tantangan dalam mendeteksi *deepfake* pada video dengan resolusi rendah atau pencahayaan ekstrem.

Pada pengembangan lebih lanjut, disarankan mengintegrasikan *ensemble learning* untuk meningkatkan ketahanan model terhadap manipulasi yang lebih kompleks, serta penambahan variasi dataset dengan *noise*, *blur*, dan pencahayaan ekstrem agar lebih adaptif terhadap kondisi nyata. Penggunaan model multimodal yang mengombinasikan analisis visual, audio, dan gerakan wajah juga perlu dieksplorasi untuk meningkatkan deteksi *deepfake* dalam format video yang lebih dinamis, sementara optimalisasi efisiensi komputasi penting agar model dapat diterapkan dalam sistem keamanan digital dengan sumber daya terbatas.

DAFTAR PUSTAKA

[1] M. Sharma and M. Kaur, “A Review of Deepfake Technology: An Emerging AI Threat,” in *Soft Computing for Security Applications: Proceedings of ICSCS 2021*, G. Ranganathan, X. Fernando, F. Shi, and Y. El Allioui, Eds., Singapore: Springer Singapore, 2022, pp. 605–619. doi: 10.1007/978-981-16-5301-8_44.

[2] Sum and Substance Ltd, “Deepfake Cases Surge in Countries Holding 2024 Elections, Sumsub Research Shows,” Sumsub. Accessed: Jan. 26, 2025. [Online]. Available: https://sumsub.com/newsroom/deepfake-cases-surge-in-countries-holding-2024-elections-sumsub-research-shows/

[3] Y. Mirsky and W. Lee, “The Creation and Detection of Deepfakes: A Survey,” *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–41, Jan. 2021, doi: 10.1145/3425780.

[4] S. D. Bray, S. D. Johnson, and B. Kleinberg, “Testing human ability to detect ‘deepfake’ images of human faces,” *J. Cybersecurity*, vol. 9, no. 1, p. tyad011, Jan. 2023, doi: 10.1093/cybsec/tyad011.

[5] T. T. Nguyen *et al.*, “Deep learning for deepfakes

creation and detection: A survey," *Comput. Vis. Image Underst.*, vol. 223, p. 103525, 2022, doi: 10.1016/j.cviu.2022.103525.

[6] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "CNN-generated images are surprisingly easy to spot... for now," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, IEEE, 2020, pp. 8695–8704. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Wang_CNN-Generated_Images_Are_Surprisingly_Easy_to_Spot..._for_Now_CVPR_2020_paper.html

[7] M. Miller, "Deepfakes: Real Threat," 2023. [Online]. Available: https://kpmg.com/kpmg-us/content/dam/kpmg/pdf/2023/deepfakes-real-threat.pdf

[8] A. Herdian and U. Sumarwan, "Analisis Kriminologi Deepfake Melalui Media Sosial Berdasarkan Teori Rational Choice," *IKRA-ITH Hum. J. Sos. dan Hum.*, vol. 9, no. 1, pp. 323–331, 2025, [Online]. Available: https://ojs.upi-yai.ac.id/index.php/ikraith-humaniora/article/view/4496

[9] Z. Xue, X. Jiang, Q. Liu, and Z. Wei, "Global–Local Facial Fusion Based GAN Generated Fake Face Detection," *Sensors*, vol. 23, no. 2, p. 616, 2023, doi: 10.3390/s23020616.

[10] M. A. A. Jufri and A. K. Putra, "Aspek Hukum Internasional Dalam Pemanfaatan Deepfake Technology Terhadap Perlindungan Data Pribadi," *Uti Possidetis J. Int. Law*, vol. 2, no. 1, pp. 31–57, 2021, doi: 10.22437/up.v2i1.11093.

[11] M. S. Rana, M. N. Nobi, B. Murali, and A. H. Sung, "Deepfake Detection: A Systematic Literature Review," *IEEE Access*, vol. 10, pp. 25494–25513, 2022, doi: 10.1109/ACCESS.2022.3154404.

[12] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, "Deepfakes and beyond: A Survey of face manipulation and fake detection," *Inf. Fusion*, vol. 64, pp. 131–148, 2020, doi: 10.1016/j.inffus.2020.06.014.

[13] J. Liu, L. Wang, R. Wang, J. Ke, X. Ye, and Y. Wu, "Exposing the Forgery Clues of DeepFakes via Exploring the Inconsistent Expression Cues," *Int. J. Intell. Syst.*, vol. 2025, no. 1, p. 7945646, Jan. 2025, doi: 10.1155/int/7945646.

[14] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, "Celeb-DF: A Large-Scale Challenging Dataset for DeepFake Forensics," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, IEEE, 2020, pp. 3207–3216. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Li_Celeb-DF_A_Large-Scale_Challenging_Dataset_for_DeepFake_Forensics_CVPR_2020_paper.html

[15] P. M. G. I. Reis and R. O. Ribeiro, "A forensic evaluation method for DeepFake detection using DCNN-based facial similarity scores," *Forensic Sci. Int.*, vol. 358, p. 111747, 2024, doi: 10.1016/j.forsciint.2023.111747.

[16] X. Chang, J. Wu, T. Yang, and G. Feng, "DeepFake Face Image Detection based on Improved VGG Convolutional Neural Network," in *2020 39th Chinese Control Conference (CCC)*, Shenyang: IEEE, 2020, pp. 7252–7256. doi: 10.23919/CCC50068.2020.9189596.

[17] G. I. W. Tamtama and I. K. D. Senapartha, "Fake Face Detection System Using MobileNets Architecture," *J. Comput. Eng. Syst. Sci.*, vol. 8, no. 2, pp. 329–338, 2023, [Online]. Available: https://jurnal.unimed.ac.id/2012/index.php/cess/article/view/43762

[18] A. Chong and H. Ng, "DeepFake-Detect," Github. Accessed: Jan. 15, 2025. [Online]. Available: https://github.com/aaronchong888/DeepFake-Detect

[19] Purwono, A. Ma'arif, W. Rahmaniar, H. I. K. Fathurrahman, A. K. Frisky, and Q. M. ul Haq, "Understanding of Convolutional Neural Network (CNN): A Review," *Int. J. Robot. Control Syst.*, vol. 2, no. 4, pp. 739–748, 2022, [Online]. Available: https://www.pubs2.ascee.org/index.php/IJRCS/article/view/888

[20] TensorFlow Developers, "TensorFlow v2.16.1." TensorFlow, 2024. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/lite

[21] S. Chandana, C. R. Nagarathna, A. Amrutha, and A. Jayasri, "Detection Of Image Forgery Using Error Level Analysis," in *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Bangalore: IEEE, 2024, pp. 1–5. doi: 10.1109/IITCEE59897.2024.10467523.

[22] Fake Image Detector, "Fake Image Detector," Fake Image Detector. Accessed: Jan. 15, 2025. [Online]. Available: https://www.fakeimagedetector.com/