JURNAL INSY PRO Information System and Processing
Volume 9, Nomor 2, November 2024 P–ISSN 2597-419X, E-ISSN 2579-468X
Ojs : http://journal.uin-alauddin.ac.id/index.php/insypro
Email : insypro@uin-alauddin.ac.id

Analisis Performa *Convolutional Neural Network* dengan *Hyperparameter Tuning* dalam Mendeteksi Gambar *Deepfake*

Performance Analysis of Convolutional Neural Networks with Hyperparameter Tuning in Detecting Deepfake Images

Darmatasia[1)], Abdur Rahman Ramli[2)], Azizah Salsabila[3)] Fhatiah Adiba[4)]

[1,2,3]Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Alauddin Makassar
Jl. H. M. Yasin Limpo No.36 Kel. Romang Polong, Kec. Somba Opu, Kab. Gowa, Kode Pos 92118
[4]Program Studi Teknik Komputer, Fakultas Teknik, Universitas Negeri Makassar
Jl. Daeng Tata Raya Parang Tambung, Kec. Tamalate, Kota Makassar, Sulawesi Selatan, Kode Pos 90224

E-mail: darmatasia@uin-alauddin.ac.id[-1)], rahman.ramli@uin-alauddin.ac.id [2)], 60200120062@uin-alauddin.ac.id[3)], fhatiah.adiba@unm.ac.id[4)]

Abstrak – Penelitian ini menganalisis performa *Convolutional Neural Network* (CNN) dalam mendeteksi gambar *deepfake* dengan fokus pada *tuning hyperparameter*. Penelitian ini terdiri dari dua kelas yaitu gambar *fake* dan gambar *real* dengan jumlah data masing-masing 5000 untuk setiap kelas. *Tuning hyperparameter* dilakukan dengan menggunakan *library* keras-tuner, sebuah *framework* yang digunakan untuk *tuning hyperparameter* secara otomatis pada model yang dibangun menggunakan Keras sehingga tidak perlu melakukan *tuning hyperparameter* secara manual melalui *trial* and *error*. Adapun strategi pencarian *hyperparameter* menggunakan *random search*. Hasil penelitian menunjukkan bahwa *tuning hyperparameter* secara signifikan meningkatkan akurasi deteksi model. Berbagai eksperimen dilakukan untuk mengevaluasi dampak pengaturan *hyperparameter*, seperti jumlah dan ukuran filter, *learning rate*, dan *optimizer*. Analisis terhadap berbagai *optimizer* menghasilkan variasi performa yang signifikan, Adam *optimizer* mencapai akurasi tertinggi sebesar 83% menggunakan kombinasi 32 filter berukuran 3x3 pada *layer* pertama dan 128 filter berukuran 5x5 pada *layer* kedua. RMSProp dan AdamW masing-masing mencapai akurasi 82%, SGD *optimizer* menghasilkan akurasi 75%, sedangkan Adadelta *optimizer* memperoleh akurasi 71%. Hasil penelitian ini menegaskan bahwa pemilihan *optimizer* dan pengaturan *hyperparameter* yang tepat memiliki dampak signifikan terhadap kinerja model dalam mendeteksi pola dalam data. Penelitian ini juga menekankan pentingnya optimalisasi filter dan ukuran pada setiap *layer* dalam meningkatkan akurasi model.

Kata Kunci: deepfake, *convolutional neural network, hyperparameter tuning*

Abstract *– This research analyzes the performance of Convolutional Neural Network (CNN) in detecting deepfake images with a focus on hyperparameter tuning. The study consists of two classes: fake images and real images, with each class containing 5000 data samples. Hyperparameter tuning is conducted using the Keras-tuner library, a framework used for automatic hyperparameter tuning on models built with Keras, eliminating the need for manual trial and error tuning. The hyperparameter search strategy employed is random search. The results of the study indicate that hyperparameter tuning significantly improves the model's detection accuracy. Various experiments were conducted to evaluate the impact of hyperparameter settings, such as the number and size of filters, learning rate, and optimizer. Analysis of different optimizers showed significant variations in performance, with Adam Optimizer achieving the highest accuracy of 83% using a combination of 32 filters sized 3x3 in the first layer and 128 filters sized 5x5 in the second layer. RMSProp and AdamW each achieved 82% accuracy, SGD Optimizer achieved 75% accuracy, while Adadelta Optimizer achieved 71% accuracy. The findings of this study affirm that the selection of optimizer and appropriate hyperparameter settings have a significant impact on the model's performance in detecting patterns in the data. This study also emphasizes the importance of optimizing filters and sizes in each layer to enhance model accuracy.*

Keywords: *deepfake, convolutional neural network, hyperparameter tuning*

PENDAHULUAN

Di era digital yang semakin maju, manipulasi media telah mencapai tingkat kecanggihan yang belum pernah terjadi sebelumnya. Salah satu bentuk manipulasi yang paling menonjol adalah *deepfake*. *Deepfake* adalah teknologi yang menggunakan *Artificial Intelligence* (AI) untuk membuat video, gambar, atau audio yang sangat mirip dengan aslinya, tetapi sebenarnya telah dimanipulasi atau dipalsukan. Istilah *deepfake* berasal dari gabungan antara *Deep Learning* dan *fake* yang berarti palsu (Heidari et al., 2024).

Teknologi *deepfake* bekerja dengan cara mempelajari pola dari data visual dan audio seseorang, seperti gerakan wajah, suara, dan ekspresi, kemudian menggunakannya untuk menghasilkan konten yang tampak asli tetapi sebenarnya sudah diubah. Proses ini biasanya melibatkan Jaringan Saraf Tiruan yang dilatih pada dataset besar yang berisi gambar atau rekaman audio dari target.

Deteksi *deepfake* dianggap sebagai permasalahan klasifikasi biner yang berusaha membedakan antara konten asli atau bukan. Secara umum, domain penelitian dari deteksi *deepfake* dapat dibagi menjadi beberapa kategori yaitu deteksi gambar *deepfake*, suara *deepfake*, video *deepfake*, dan multimedia *deepfake*. Masing-masing kategori memiliki tantangan dan membutuhkan pendekatan yang berbeda untuk menyelesaikannya.

Salah satu jenis *deepfake* adalah gambar *deepfake*. Gambar *deepfake* mengacu kepada pembuatan gambar palsu yang dapat dibuat lebih nyata dengan menggunakan *pasca*-pemrosesan yang sesuai. Salah satu bidang yang paling sulit dalam pemalsuan gambar adalah deteksi gambar wajah palsu. Foto palsu dapat membangun identitas palsu di *platform* media sosial, sehingga memungkinkan terjadinya pencurian informasi pribadi secara tidak sah. Generator gambar palsu dapat digunakan untuk membuat foto dengan konten yang tidak pantas.

Deepfake dapat digunakan untuk tujuan positif, seperti dalam pembuatan film dan efek visual, atau untuk tujuan negatif, seperti membuat berita palsu, pornografi *non*-konsensual, atau penipuan. Keberadaan *deepfake* menimbulkan berbagai isu etis dan hukum, terutama terkait dengan privasi, keamanan, dan penyebaran informasi yang menyesatkan. *Deepfake* dapat meningkatkan bahaya yang sudah ada terhadap negara-negara dan budaya *online* dengan meningkatkan ketidakpercayaan dan keraguan pada sebagian besar sumber daya *online* (Khder et al., 2023).

Deepfake sulit dideteksi karena menggunakan rekaman asli, memiliki audio yang terdengar otentik, dan dioptimalkan untuk menyebar dengan cepat di media sosial sehingga upaya untuk mendeteksi dan mencegah penyalahgunaan teknologi ini terus berkembang seiring dengan kemajuan teknologi *deepfake* itu sendiri.

Penelitian (Heidari et al., 2024) menyebutkan bahwa *Deep Learning* adalah metode yang paling banyak digunakan dalam mendeteksi *deepfake* baik *deepfake* berupa gambar, suara, video, atau multimedia. Algoritma *Deep Learning* yang paling banyak digunakan dalam deteksi gambar *deepfake* adalah *Convolutional Neural Network* (CNN). CNN telah terbukti sangat efektif dalam berbagai tugas pemrosesan gambar, termasuk deteksi *deepfake*. CNN memiliki kemampuan unik untuk mempelajari dan mengekstraksi fitur-fitur yang relevan dari data visual secara hierarkis. Arsitektur ini terdiri dari beberapa lapisan konvolusi yang diikuti oleh lapisan *pooling* dan *fully connected*, memungkinkan model untuk menangkap pola-pola kompleks dalam gambar pada berbagai tingkat abstraksi. Karakteristik tersebut membuat CNN sangat cocok untuk permasalahan deteksi *deepfake*, di mana perbedaan antara gambar asli dan yang dimanipulasi seringkali sangat halus dan sulit dideteksi oleh mata manusia.

Efektivitas CNN dalam mendeteksi *deepfake* sangat bergantung pada desain arsitektur jaringan dan pemilihan nilai-nilai *hyperparameter* yang tepat. *Hyperparameter* adalah parameter-parameter yang mengontrol proses pelatihan dan struktur model, seperti jumlah *layer*, ukuran filter konvolusi, *learning rate*, *batch size*, fungsi aktivasi, dan metode regularisasi. Pemilihan *hyperparameter* yang optimal dapat secara signifikan meningkatkan kinerja model, sementara konfigurasi yang tidak tepat dapat mengakibatkan *underfitting* atau *overfitting*, yang pada akhirnya mengurangi kemampuan generalisasi model (Bochinski et al., 2017).

Peran *hyperparameter tuning* menjadi sangat penting. *Hyperparameter tuning* adalah proses sistematis untuk mengoptimalkan nilai-nilai *hyperparameter* guna meningkatkan performa model. Proses melibatkan eksplorasi ruang *hyperparameter* yang luas untuk menemukan kombinasi parameter yang menghasilkan kinerja terbaik pada dataset yang diberikan.

Analisis performa CNN dengan *hyperparameter tuning* dalam deteksi *deepfake* tidak hanya bertujuan untuk meningkatkan akurasi deteksi, tetapi juga untuk memahami *trade-off* antara berbagai aspek kinerja model. Misalnya, model yang sangat kompleks mungkin mencapai akurasi tinggi pada data pelatihan tetapi memiliki kemampuan generalisasi yang buruk atau terlalu lambat untuk digunakan dalam aplikasi *real-time*.

Penelitian ini berfokus pada pengembangan metode yang tahan terhadap teknik *deepfake* yang terus berkembang. *Hyperparameter tuning* digunakan sebagai salah satu pendekatan untuk meningkatkan

performa algoritma CNN. Penelitian ini diharapakan dapat lebih meningkatkan kepercayaan publik terhadap konten digital dengan menyediakan alat yang lebih andal untuk mendeteksi manipulasi serta membantu melindungi individu dan organisasi dari potensi dampak negatif *deepfake*, seperti penipuan atau pencemaran nama baik.

METODOLOGI PENELITIAN

Penelitian ini menggunakan metodologi *Cross Industry Standard Process for Data Mining* (CRISP-DM). CRISP-DM merupakan salah satu standarisasi dalam penambangan data (*data mining*) yang terdiri dari enam fase. CRISP-DM menyediakan standar proses baku untuk data mining yang dapat diterapkan ke dalam strategi pemecahan masalah umum pada bisnis atau pada unit penelitian (Budiman et al., 2012). Adapun fase dari CRISP-DM yaitu sebagai berikut:

1) *Business Understanding* atau pemahaman domain (penelitian). Pada fase ini dibutuhkan pemahaman secara substansi dari kegiatan *data mining* yang akan dilakukan seperti menentukan tujuan atau sasaran, membuat perencanaan serta jadwal penelitian, dan menentukan kriteria keberhasilan yang jelas. Penelitian ini bertujuan untuk menganalisis performa *Convolutional Neural Network* dengan melakukan *hyperparameter tuning* dalam mendeteksi gambar *deepfake*.

2) Data Understanding atau pemahaman data. Pada fase ini dilakukan pengumpulan data, mempelajari data yang ada agar bisa mengenal karakteristik dari data yang akan digunakan. Pada penelitian ini, Peneliti menggunakan dataset yang diperoleh dari Kaggle (Le, 2022). Dataset terdiri dari 2 kelas yaitu "*fake*" dan "*real*" dengan jumlah gambar 95134 untuk kelas "*fake*" dan 95201 untuk kelas *real*. Namun dalam penelitian ini hanya menggunakan 5000 gambar untuk setiap kelas. Setiap gambar berukuran 256x256 piksel yang merupakan citra berwarna. Contoh dataset dapat dilihat pada Gambar 1 dan Gambar 2.

Gambar 1. Contoh Gambar *Fake* (Le, 2022)

Gambar 2. Contoh Gambar *Real* (Le, 2022)

3) *Data preparation* atau persiapan data. Pada fase ini dilakukan *pre-processing* sesuai dengan kebutuhan. Data *preprocessing* yang dilakukan meliputi *resize* ukuran citra dari 256x256 piksel menjadi 28x28 piksel, mengubah citra dari RGB ke *grayscale*, dan melakukan normalisasi citra. Pada tahap ini juga dilakukan partisi data yaitu 80% data digunakan untuk pelatihan dan 20% data digunakan untuk pengujian.

4) *Modeling*, pada fase ini bertujuan untuk menentukan teknik *data mining* yang akan diimplementasikan untuk menyelesaikan permasalahan. Pada penelitian ini menggunakan *Convolutional Neural Network* (CNN) yang selanjutnya dilakukan *tuning hyperparameter* untuk memaksimalkan kinerja model.

Convolutional Neural Network (CNN) adalah arsitektur *deep learning* yang dirancang khusus untuk memproses data dengan struktur *grid*, terutama gambar. CNN terinspirasi dari cara kerja visual *cortex* otak manusia dan terdiri dari beberapa lapisan utama: *convolutional layer* yang mengekstrak fitur menggunakan filter, *activation layer* yang menambahkan *non*-linearitas, *pooling layer* yang mengurangi dimensi data, dan *fully connected layer* yang melakukan klasifikasi akhir. Keunikan CNN terletak pada kemampuannya untuk secara otomatis mempelajari hierarki fitur, mulai dari fitur sederhana

seperti tepi dan tekstur hingga fitur kompleks seperti bentuk dan objek.

Cara kerja CNN dimulai dengan operasi konvolusi terhadap data *input*, di mana filter bergerak melintasi gambar untuk menghasilkan *feature map*. Setiap *feature map* kemudian diaktivasi menggunakan fungsi aktivasi yang telah ditentukan, selanjutnya dilakukan *pooling* untuk mengurangi dimensi sambil mempertahankan informasi penting. Proses ini diulang beberapa kali untuk mengekstrak fitur yang semakin kompleks. Hasil ekstraksi fitur diratakan (*flattened*) dan diteruskan ke *fully connected layer* untuk klasifikasi. (Lecun et al., 2015).

Tuning hyperparameter pada *Convolutional Neural Network* (CNN) adalah hal yang sangat penting untuk mengoptimalkan kinerja model dalam mempelajari pola dan fitur dari data, sehingga model dapat memberikan hasil yang lebih akurat dan generalisasi yang lebih baik. *Tuning hyperparameter* merupakan proses eksperimental yang dilakukan untuk menemukan kombinasi terbaik dari nilai-nilai hyperparameter, yang secara signifikan dapat mempengaruhi performa CNN.

Tuning hyperparameter dilakukan dengan menggunakan *library* keras-tuner, sebuah *framework* yang digunakan untuk tuning hyperparameter secara otomatis pada model yang dibangun menggunakan Keras. Keras Tuner memfasilitasi proses pencarian *hyperparameter* terbaik dengan cara yang lebih sistematis dan efisien, sehingga tidak perlu melakukan *tuning hyperparameter* secara manual melalui *trial* and *error*. Adapun strategi pencarian *hyperparameter* menggunakan *random search*.

Pada penelitian ini, arsitektur CNN yang dibangun terdiri dari dua *layer* dengan jumlah filter antara 32-128 dengan 16 step, ukuran kernel dari 3x3 sampai 5x5, fungsi aktivasi menggunakan reLu, *learning* rate 0.01 dan 0.001, *optimizer* yang akan dievaluasi yaitu Adam, AdamW, Adadelta, SGD, dan RMSProp, serta jumlah *epoch* yang digunakan adalah 10.

5) Evaluation, fase ini bertujuan untuk mengevaluasi dan menginterpretasikan hasil yang diperoleh pada fase sebelumnya. Evaluasi secara mendalam diperlukan agar model yang diperoleh sesuai dengan sasaran atau tujuan yang telah ditetapkan pada fase pertama. Beberapa teknik evaluasi algoritma dari model diantaranya pengukuran tingkat keakuratan model, presisi dan *recall*.

$$Akurasi = \frac{\sum TP + \sum TN}{Jumlah\ Data} \quad (1)$$

$$Presisi = \frac{(TP)}{(TP)+(FP)} \quad (2)$$

$$Recall = \frac{(TP)}{(TP)+(FN} \quad (3)$$

TP merupakan *True Positive*, FN merupakan *False Negative*, FP merupakan *False Positive*, dan TN merupakan *True Negative*. Suatu kondisi dikatakan TP atau TN Ketika suatu observasi dikenali dengan benar, sedangkan FP dan FN merupakan suatu observasi salah diidentifikasi.

Recall adalah rasio jumlah *record* relevan yang ditemukan dengan jumlah *record* yang ditemukan tidak relevan dan relevan, sedangkan presisi adalah rasio jumlah *record* relevan yang ditemukan dengan jumlah *record* yang ditemukan tidak relevan dan relevan. (Jizba, 2000).

6) *Deployment* atau penyebaran adalah fase penyusunan laporan atau presentasi dari pengetahuan yang didapat dari evaluasi pada proses *data mining*.

HASIL DAN PEMBAHASAN

Hasil akurasi deteksi *deepfake* dengan *tuning hyperparameter* pada penelitian ini dapat dilihat pada Tabel 1.

Tabel 1 Hasil Akurasi Deteksi *Deepfake* dengan *Tuning Hyperparameter*

Optimizer	**Jumlah Filter**	**Ukuran Filter**	***Learning Rate***	**Akurasi**
Adadelta	L1: 48 L2: 32	[3x3] [3x3]	0.01	71%
SGD	L1: 48 L2: 64	[3x3] [3x3]	0.01	75%
RMS Prop	L1: 80 L1: 128	[5x5] [3x3]	0.01	82%
AdamW	L1: 32 L2: 80	[3x3] [3x3]	0.001	82%
Adam	**L1: 32 L2: 128**	**[3x3] [5x5]**	**0.001**	**83%**

Berdasarkan Tabel 1 dapat dilihat bahwa akurasi terbaik diperoleh dengan menggunakan Adam *optimizer* yaitu 83% dengan jumlah filter pada *layer* pertama adalah 32 filter berukuran 3x3, jumlah filter pada *layer* kedua adalah 128 filter berukuran 5x5, dan nilai *learning rate* 0.001. Kombinasi dari jumlah filter

dan ukuran yang berbeda pada setiap *layer* berkontribusi signifikan terhadap performa model. Filter yang lebih banyak pada *layer* kedua, yang memiliki ukuran yang lebih besar, membantu dalam menangkap pola-pola yang lebih kompleks dalam data.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
conv2d_1 (Conv2D)	(None, 22, 22, 128)	102,528
flatten (Flatten)	(None, 61952)	0
dense (Dense)	(None, 32)	1,982,496
dense_1 (Dense)	(None, 2)	66

Total params: 2,085,410 (7.96 MB)
Trainable params: 2,085,410 (7.96 MB)
Non-trainable params: 0 (0.00 B)

Gambar 3. Arsitektur Terbaik dengan Adam *Optimizer*

Adadelta *optimizer* memperoleh akurasi tertinggi sebesar 71% dengan jumlah filter pada *layer* pertama adalah 48 filter berukuran 3x3, jumlah filter pada *layer* kedua adalah 32 filter berukuran 3x3, dan nilai *learning rate* 0.01. Penggunaan Adadelta sebagai *optimizer* menunjukkan kemampuan yang cukup baik dalam menangani pembaruan bobot, meskipun akurasinya tidak setinggi beberapa *optimizer* lainnya. Jumlah filter yang lebih banyak pada *layer* pertama dapat membantu dalam mengekstraksi fitur yang lebih kompleks dari data, sementara *layer* kedua berfungsi untuk memperhalus hasil yang diperoleh dari *layer* sebelumnya. Ukuran filter yang seragam di kedua *layer* menunjukkan pendekatan yang konsisten dalam menangkap pola-pola yang relevan dari *input*.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 48)	480
conv2d_1 (Conv2D)	(None, 24, 24, 32)	13,856
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 48)	884,784
dense_1 (Dense)	(None, 2)	98

Total params: 899,218 (3.43 MB)
Trainable params: 899,218 (3.43 MB)
Non-trainable params: 0 (0.00 B)

Gambar 4. Arsitektur Terbaik dengan Adadelta *Optimizer*

SGD *optimizer* memperoleh akurasi tertinggi sebesar 75% dengan jumlah filter pada *layer* pertama adalah 48 filter berukuran 3x3, jumlah filter pada *layer* kedua adalah 64 filter berukuran 3x3, dan nilai *learning rate* 0.01. Penggunaan SGD sebagai *optimizer* menunjukkan kinerja yang baik, dengan akurasi yang mencapai 75%. Jumlah filter yang signifikan pada layer pertama dan *layer* kedua berkontribusi pada kemampuan model dalam mengekstrak dan memproses fitur dari data dengan efisiensi yang baik. Filter yang lebih banyak pada *layer* kedua (64 filter) memungkinkan model untuk menangkap lebih banyak variasi dan kompleksitas dalam data, yang berpotensi meningkatkan akurasi klasifikasi. Dalam konteks ini, SGD berhasil mencapai hasil yang cukup kompetitif, menunjukkan bahwa teknik optimasi yang tepat dapat secara signifikan mempengaruhi kinerja model.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 48)	480
conv2d_1 (Conv2D)	(None, 24, 24, 64)	27,712
flatten (Flatten)	(None, 36864)	0
dense (Dense)	(None, 96)	3,539,040
dense_1 (Dense)	(None, 2)	194

Total params: 3,567,426 (13.61 MB)
Trainable params: 3,567,426 (13.61 MB)
Non-trainable params: 0 (0.00 B)

Gambar 5. Arsitektur Terbaik dengan SGD *Optimizer*

RMSProp *optimizer* memperoleh akurasi tertinggi sebesar 82% dengan jumlah filter pada *layer* pertama adalah 80 filter berukuran 3x3, jumlah filter pada *layer* kedua adalah 128 filter berukuran 3x3, dan nilai *learning rate* 0.01.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 24, 24, 80)	2,080
conv2d_1 (Conv2D)	(None, 20, 20, 128)	256,128
flatten (Flatten)	(None, 51200)	0
dense (Dense)	(None, 80)	4,096,080
dense_1 (Dense)	(None, 2)	162

Total params: 4,354,450 (16.61 MB)
Trainable params: 4,354,450 (16.61 MB)
Non-trainable params: 0 (0.00 B)

Gambar 6. Arsitektur Terbaik dengan RMSProp *Optimizer*

AdamW *optimizer* memperoleh akurasi tertinggi sebesar 82% dengan jumlah filter pada *layer* pertama adalah 32 filter berukuran 3x3, jumlah filter pada *layer* kedua adalah 80 filter berukuran 3x3, dan nilai *learning rate* 0.001. Dengan penggunaan 80 filter di *layer* kedua, model mampu menghasilkan representasi fitur yang lebih kaya dan terperinci, meningkatkan akurasi prediksi. Filter yang lebih banyak pada *layer* kedua memungkinkan model untuk beradaptasi lebih baik terhadap keragaman data, sehingga mampu melakukan generalisasi yang lebih baik pada data yang tidak terlihat. *Layer* pertama dengan 32 filter berfungsi untuk menyiapkan dasar yang kuat bagi proses ekstraksi fitur, memastikan bahwa informasi penting dapat ditangkap dengan baik. Secara keseluruhan, hasil ini menunjukkan bahwa penggunaan AdamW sebagai *optimizer* dapat memberikan hasil yang kompetitif, berkat kemampuannya dalam mengelola pembaruan bobot dengan efisien dan stabil.

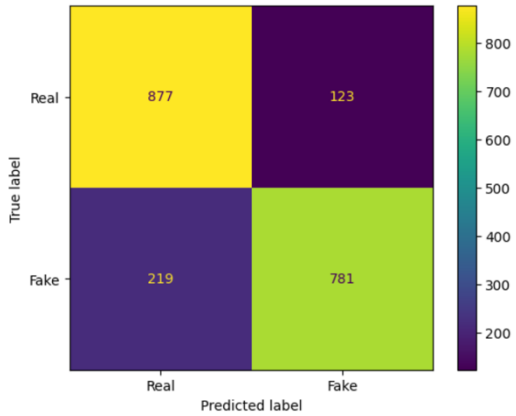
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
conv2d_1 (Conv2D)	(None, 24, 24, 80)	23,120
flatten (Flatten)	(None, 46080)	0
dense (Dense)	(None, 80)	3,686,480
dense_1 (Dense)	(None, 2)	162

Total params: 3,710,082 (14.15 MB)
Trainable params: 3,710,082 (14.15 MB)
Non-trainable params: 0 (0.00 B)

Gambar 7. Arsitektur Terbaik dengan AdamW *Optimizer*

Confusion matrix pada penelitian ini dapat dilihat pada Gambar 8.

Volume 9, Nomor 2, November 2024 P–ISSN 2597-419X, E-ISSN 2579-468X
Ojs : http://journal.uin-alauddin.ac.id/index.php/insypro
Email : insypro@uin-alauddin.ac.id



Gambar 8. *Confusion Matrix*

Berdasarkan Gambar 8, terlihat bahwa terdapat 123 citra dari kelas *Real* yang salah dideteksi sebagai kelas *Fake*. Hal tersebut menunjukkan adanya tantangan dalam akurasi deteksi model, di mana sejumlah citra yang seharusnya diidentifikasi sebagai kelas *Real* justru terklasifikasi keliru. Kesalahan ini dapat mengindikasikan bahwa fitur-fitur yang digunakan oleh model untuk membedakan antara kelas *Real* dan *Fake* tidak cukup kuat atau mungkin ada faktor lain yang mempengaruhi, seperti kualitas gambar yang hanya terdiri dari 28x28 piksel, pencahayaan, atau variasi dalam representasi data yang tidak tertangkap oleh model. Selain itu, terdapat 219 citra dari kelas *Fake* yang salah dideteksi sebagai kelas *Real*. Hal ini menunjukkan bahwa model juga mengalami kesulitan dalam mengidentifikasi citra yang seharusnya termasuk dalam kelas *Fake*.

Evaluasi presisi dan *recall* pada setiap kelas dengan *hyperparameter* terbaik yaitu menggunakan Adam *optimizer* dengan jumlah filter pada *layer* pertama adalah 32 filter berukuran 3x3, jumlah filter pada *layer* kedua adalah 128 filter berukuran 5x5, dan nilai *learning rate* 0.001 dapat dilihat pada Tabel 2.

Tabel 2. Presisi, *Recall* dan F-1 *Score* setiap Kelas

Kelas	Presisi	*Recall*	F1-*Score*
Real	82%	88%	84%
Fake	86%	78%	82%

Presisi kelas *Real* 80% menunjukkan persentase deteksi yang benar dari semua hasil yang terdeteksi sebagai kelas *Real*. Artinya, dari semua prediksi yang diidentifikasi sebagai kelas *Real*, 80% adalah benar-benar kelas *Real*, sedangkan sisanya adalah kesalahan (*false positive*). *Recall* 88% pada kelas *Real* menunjukkan persentase data yang benar-benar kelas *Real* yang berhasil terdeteksi oleh model. Artinya, dari semua data yang seharusnya diklasifikasikan sebagai kelas *Real*, model berhasil mendeteksi 88% dengan benar, sedangkan 12% tidak terdeteksi (*false negative*). F1-*Score* 84% dari kelas *Real* merupakan rasio antara presisi dan *recall*. Dengan nilai 84%, hal ini menunjukkan keseimbangan antara presisi (80%) dan *recall* (88%). F1-*score* lebih relevan ketika ada kebutuhan untuk menyeimbangkan antara kesalahan tipe *false positive* dan *false negative*. Secara keseluruhan dapat disimpulkan bahwa hasil deteksi kelas *Real* memiliki *recall* yang lebih tinggi (88%) daripada presisi (80%), artinya model cenderung lebih sering mendeteksi kelas *Real* (sedikit terjadi *missed detection* atau *false negative*), tetapi ada beberapa deteksi yang salah diklasifikasikan sebagai kelas *Real* (*false positive*) sehingga presisinya lebih rendah.

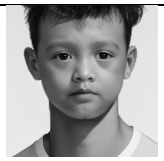
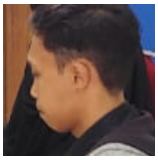
Presisi 86% pada kelas *Fake* menunjukkan dari semua prediksi yang teridentifikasi sebagai kelas *Fake*, 86% benar-benar adalah kelas *Fake* yang berarti jumlah kesalahan deteksi (*false positive*) lebih sedikit dibandingkan dengan kelas *Real*. Nilai *Recall* 78% menunjukkan bahwa dari semua data yang seharusnya diklasifikasikan sebagai kelas *Fake*, model hanya berhasil mendeteksi 78% dengan benar, artinya, 22% dari data kelas *Fake* tidak terdeteksi dengan benar (*false negative*). Nilai F1-*Score* 82% menunjukkan F1-*score* untuk kelas *Fake* adalah 82%, lebih rendah dari F1-*score* kelas *Real* yang menunjukkan keseimbangan antara presisi dan *recall*, tetapi masih ada penurunan *recall* yang cukup signifikan dibandingkan presisi, sehingga F1-*score* sedikit lebih rendah. Secara keseluruhan kelas *Fake* memiliki presisi yang tinggi (86%) dalam mendeteksi kelas *Fake*, artinya lebih jarang membuat kesalahan *false positive*. Namun, *recall* yang lebih rendah (78%) menunjukkan bahwa model lebih sering gagal mendeteksi beberapa data yang seharusnya masuk kelas *Fake* (*false negative*) sehingga menghasilkan F1-*score* yang lebih rendah (82%) dibandingkan kelas *Real*.

Kelas *Real* memiliki *recall* yang lebih tinggi daripada kelas *Fake*, artinya model lebih mampu mendeteksi semua *instance* kelas *Real*, tetapi dengan nilai presisi yang lebih rendah. Kelas *Fake* memiliki presisi yang lebih tinggi, tetapi *recall* yang lebih rendah, artinya model lebih akurat dalam mendeteksi kelas *Fake*, tetapi ada lebih banyak data kelas *Fake* yang tidak terdeteksi. Secara keseluruhan, nilai F1-*score* menunjukkan bahwa kinerja model lebih seimbang untuk kelas *Real* (84%) dibandingkan dengan kelas *Fake* (82%).

Evaluasi lebih lanjut dalam penelitian ini dilakukan dengan mengambil sampel berupa gambar secara langsung yang terdiri dari 5 gambar *real* dan 5 gambar *fake* yang di *generate* oleh aplikasi artguru.ai dan blackbox.ai. Berikut adalah hasil deteksi dari model terbaik dalam penelitian ini.

Volume 9, Nomor 2, November 2024 P–ISSN 2597-419X, E-ISSN 2579-468X
Ojs : http://journal.uin-alauddin.ac.id/index.php/insypro
Email : insypro@uin-alauddin.ac.id

Tabel 3. Hasil Pengujian Model Terbaik

Input	Kelas Sebenarnya	Deteksi Model
	Fake	Fake
	Fake	Fake
	Fake	Real
	Fake	Real
	Real	Real
	Real	Real
	Real	Real
	Real	Fake

Berdasarkan Tabel 3. dapat dilihat bahwa terdapat 2 sampel kelas *Fake* yang dideteksi sebagai kelas *Real* dan 1 sampel kelas *Real* yang salah dideteksi sebagai kelas *Fake*. Jika dianalisis lebih dalam, 2 gambar *Fake* yang salah dideteksi memiliki karakteristik yang sangat mirip dengan kelas *Real*, sedangkan gambar kelas *Real* yang salah dideteksi sebagai kelas *fake* disebabkan karena pada gambar telah diterapkan filter atau efek sehingga memiliki pencahayaan yang mirip dengan karakteristik dari gambar *Fake*. Hasil penelitian ini dapat menjadi pertimbangan untuk membuat dataset yang lebih variatif, misalnya dengan menerapkan filter atau efek pada gambar *Real* sehingga dapat membantu model menghasilkan generalisasi yang lebih baik.

KESIMPULAN

Penelitian ini berfokus pada analisis performa *Convolutional Neural Network* (CNN) dalam mendeteksi gambar *deepfake* dengan melakukan *tuning hyperparameter*. Hasil yang diperoleh menunjukkan bahwa *tuning hyperparameter* secara signifikan dapat meningkatkan akurasi deteksi model. Dalam berbagai eksperimen, model yang diterapkan menunjukkan variasi dalam akurasi tergantung pada pengaturan *hyperparameter* seperti jumlah filter, ukuran filter, dan *learning rate*.

Berdasarkan analisis yang dilakukan terhadap berbagai *optimizer* terlihat bahwa masing-masing *optimizer* menunjukkan performa yang bervariasi tergantung pada pengaturan *hyperparameter*, terutama jumlah dan ukuran filter pada setiap *layer*.

Secara keseluruhan, hasil penelitian menunjukkan bahwa pemilihan *optimizer* dan pengaturan *hyperparameter* yang tepat sangat mempengaruhi kinerja model dalam mendeteksi pola dalam data. Optimalisasi filter dan ukuran yang digunakan di setiap *layer* memiliki kontribusi signifikan terhadap akurasi model, menegaskan pentingnya *tuning hyperparameter* dalam proses pengembangan model *deep learning*.

Penelitian selanjutnya disarankan untuk memperbesar ukuran dan keragaman *dataset* yang digunakan dalam pelatihan seperti menerapkan efek atau filter pada dataset untuk membantu model dalam generalisasi dan mengurangi *overfitting*. Selain itu, pada penelitian selanjutnya juga dapat dilakukan eksplorasi lebih dalam terhadap kombinasi *hyperparameter* lain, termasuk pengaturan regularisasi dan teknik augmentasi data untuk membantu meningkatkan kinerja model. Model yang telah dibangun pada penelitian ini selanjutnya dapat diimplementasikan ke dalam sistem deteksi yang mampu bekerja secara *real-time* untuk aplikasi praktis, seperti dalam keamanan siber dan verifikasi media, agar dapat segera mendeteksi konten *deepfake* sebelum tersebar luas.

DAFTAR PUSTAKA

Bochinski, E., Senst, T., & Sikora, T. (2017). Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms. *2017 IEEE International Conference on Image Processing (ICIP)*, 3924–3928.

Budiman, I., Prahasto, T., & Christyono, Y. (2012). Data Clustering menggunakan metodologi Crisp-DM untuk pengenalan pola proporsi pelaksanaan tridharma. *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*.

Heidari, A., Jafari Navimipour, N., Dag, H., & Unal, M. (2024). Deepfake detection using deep learning methods: A systematic and comprehensive review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *14*(2), e1520.

Jizba, R. (2000). Measuring search effectiveness. *Creighton University Health Sciences Library and ….* http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Measuring+Search+Effectiveness#2

Khder, M. A., Shorman, S., Aldoseri, D. T., & Saeed, M. M. (2023). Artificial Intelligence into Multimedia Deepfakes Creation and Detection. *2023 International Conference on IT Innovation and Knowledge Discovery (ITIKD)*, 1–5. https://doi.org/10.1109/ITIKD56332.2023.10099744

Le, T.-N. (2022). *Deepfake and Real Images*.

Lecun, Y., Bengio, Y., & Hinton, G. (2015). *Deep Learning*. Macmillan Publishers Limited.