Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.4, August 2025, pp. 1770~1777
e-ISSN: 2548-6861

Mushroom Classification Using *Convolutional Neural Network MobileNetV2* Architecture for Overfitting Mitigation and Enhanced Model Generalization

Fauzan Arif Prayogi [1]*, Fariz Hasim Arvianto [2]*, Dimas Rizki Pratama [3]*, Sugiyanto Sugiyanto [4]**
* Program Studi Teknik Informatika, Universitas Dian Nuswantoro
** Program Pendidikan Jarak Jauh Informatika, Universitas Dian Nuswantoro
111202315477@mhs.dinus.ac.id [1], 111202315467@mhs.dinus.ac.id [2], 111202314994@mhs.dinus.ac.id [3],
sugiyanto@dsn.dinus.ac.id [4]

Article Info

Article history:

Received 2025-07-11
Revised 2025-07-18
Accepted 2025-07-19

Keyword:

Computer Vision,
Convolutional Neural Network,
MobileNetV2,
Classification,
Overfitting.

ABSTRACT

Fungal identification is a significant challenge due to the morphological similarities among different species. Previous studies using Convolutional Neural Networks (CNNs) for mushroom classification still face overfitting issues, which lead to poor performance on new data. Therefore, this research develops a MobileNetV2-based Convolutional Neural Network (CNN) model capable of classifying three mushroom species (Amanita, Boletus, and Lactarius) with a primary focus on mitigating overfitting. The dataset consists of 3,210 RGB images, divided into 1,979 training data, 493 validation data, and 738 testing data. The model is developed using transfer learning with MobileNetV2, combined with additional layers such as Conv2D, pooling, and Dense, along with Dropout for regularization. The training process employs the Adam optimizer with a learning rate of 1.0×10^{-5} and is monitored with EarlyStopping and ModelCheckpoint. The model successfully addresses overfitting, achieving a minimal generalization gap of 1.33%, compared to 7% in previous studies. The evaluation results show a training accuracy of 77.35%, validation accuracy of 78.79%, and testing accuracy of 76.02%, with precision of 80.6% and recall of 68.1%. The consistent performance, with a maximum difference of only 2.77% across the three datasets, demonstrates superior generalization ability and provides a strong foundation for the implementation of a reliable automatic mushroom identification system.

I. PENDAHULUAN

Jamur merupakan organisme eukariotik yang memainkan peran penting dalam ekosistem dan kehidupan manusia. Keberagaman spesies jamur yang terdapat di berbagai habitat, mulai dari hutan hingga perkotaan, menjadikannya salah satu sumber daya alam yang bernilai tinggi. Selain fungsinya sebagai dekomposer dalam siklus nutrisi, jamur juga memiliki nilai ekonomi yang signifikan, baik sebagai bahan pangan, obat-obatan, maupun dalam industri lainnya [1]. Meskipun banyak manfaatnya, identifikasi jamur sering kali menjadi tantangan, terutama bagi non-ahli, mengingat keragaman bentuk dan warna yang dimiliki oleh spesies jamur tertentu.

Seiring dengan perkembangan teknologi, khususnya dalam bidang kecerdasan buatan (AI), telah banyak diterapkan metode identifikasi otomatis menggunakan teknik pembelajaran mesin. Salah satu teknik yang banyak digunakan untuk klasifikasi citra adalah *Convolutional Neural Network* (CNN), yang memiliki kemampuan unggul dalam mengenali pola-pola kompleks pada gambar [3]. CNN, dengan arsitektur dalam *deep learning*, telah terbukti efektif dalam berbagai aplikasi pengolahan citra, termasuk dalam identifikasi jamur [1] [2]. Namun, penelitian-penelitian sebelumnya dalam klasifikasi jamur menggunakan CNN masih menghadapi tantangan signifikan berupa *overfitting*, yang mengakibatkan model memiliki kinerja tinggi pada data

pelatihan namun gagal melakukan generalisasi yang baik pada data baru.

Berbagai penelitian sebelumnya telah mengaplikasikan metode CNN untuk pengklasifikasian jamur berdasarkan citra dengan hasil yang bervariasi. Penelitian oleh Haksoro dan Setiawan (2021) menggunakan *transfer learning* dengan MobileNet V2 untuk mengklasifikasikan jamur beracun dan konsumsi, mencapai akurasi tertinggi sebesar 92,19%. Penelitian [2] juga menggunakan CNN untuk mengklasifikasikan jamur berdasarkan genus, dengan hasil akurasi tertinggi mencapai 89% pada data pelatihan dan 82% pada validasi. Penelitian sebelumnya [1] menunjukkan bahwa CNN dapat mencapai tingkat akurasi 100% dalam mengidentifikasi jamur, dengan optimasi yang dilakukan menggunakan data *preprocessing* dan *augmentasi* untuk memperkaya dataset. Namun, penelitian-penelitian tersebut masih memiliki kelemahan signifikan dalam hal mitigasi *overfitting*. Pada penelitian [1], akurasi 100% yang dicapai merupakan indikator kuat adanya *overfitting* yang parah, sementara penelitian [2] menunjukkan model terindikasi *overfitting* dengan selisih akurasi *training* dan validasi sebesar 7%.

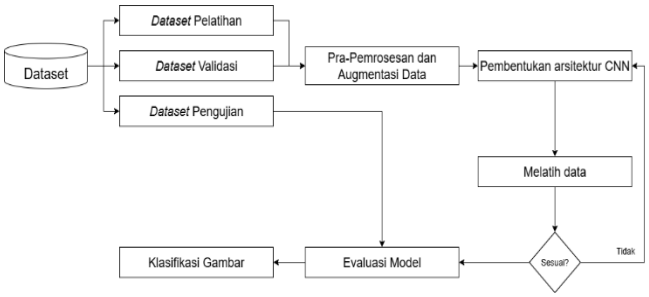
Berdasarkan tinjauan pustaka tersebut, teridentifikasi bahwa belum ada penelitian sebelumnya yang secara eksplisit mengoptimasi arsitektur MobileNetV2 dengan kombinasi teknik regularisasi dan *fine-tuning* yang komprehensif untuk klasifikasi jamur dengan fokus utama pada mitigasi *overfitting*. Kesulitan dalam mengidentifikasi jamur yang disebabkan oleh kesamaan morfologi antara berbagai jenis jamur, dikombinasikan dengan masalah *overfitting* yang konsisten muncul dalam penelitian-penelitian sebelumnya, menjadikan permasalahan utama yang perlu dijawab dalam penelitian ini adalah: "Bagaimana mengembangkan model CNN berbasis MobileNetV2 yang dapat mengklasifikasikan jamur dengan akurasi tinggi sambil meminimalkan *overfitting*?"

Untuk menjawab permasalahan tersebut, penelitian ini bertujuan untuk mengembangkan model CNN yang dapat mengklasifikasikan beberapa jenis jamur berdasarkan citra digital dengan menggunakan arsitektur MobileNet V2 yang memprioritaskan robustisitas dan kemampuan generalisasi. Secara khusus, penelitian ini bertujuan untuk mengidentifikasi dan mengklasifikasikan jenis jamur berdasarkan citra digital menggunakan arsitektur CNN MobileNet V2, mengevaluasi kinerja model CNN dengan berbagai metode augmentasi data dan *preprocessing* untuk meningkatkan akurasi identifikasi, dan mengukur efektivitas MobileNet V2 dalam mengklasifikasikan jenis jamur yang berbeda dengan mempertimbangkan sumber daya komputasi yang terbatas.

Penelitian ini diharapkan dapat memberikan kontribusi teoretis berupa demonstrasi bahwa *trade-off* antara akurasi tinggi dan kemampuan generalisasi yang baik dapat dicapai melalui pemilihan arsitektur yang tepat dan penerapan teknik regularisasi yang efektif. Secara praktis, penelitian ini akan menghasilkan model yang memberikan fondasi yang kuat untuk pengembangan sistem identifikasi jamur otomatis yang dapat diandalkan dan dapat diimplementasikan pada perangkat dengan sumber daya terbatas. Manfaat dari penelitian ini mencakup pengembangan model klasifikasi jamur yang lebih *robust* dan dapat diandalkan untuk aplikasi *real-world*, kontribusi dalam mitigasi masalah *overfitting* yang selama ini menjadi tantangan konsisten dalam penelitian klasifikasi jamur, serta penyediaan baseline model yang dapat digunakan sebagai referensi untuk pengembangan sistem identifikasi jamur yang lebih canggih di masa depan. Dengan menggunakan pendekatan yang belum pernah dieksplorasi sebelumnya dalam konteks klasifikasi jamur, penelitian ini akan memperluas penerapan CNN, khususnya MobileNet V2, dalam klasifikasi jamur yang lebih efisien dan akurat dengan fokus pada kemampuan generalisasi yang superior.

II. METODE

Alur penelitian ini dijelaskan secara rinci dalam Gambar 1, yang menampilkan keseluruhan tahapan dan komponen metodologi yang digunakan. Visualisasi ini membantu pemahaman tentang keterkaitan antar elemen penelitian dan kontribusinya terhadap pencapaian tujuan utama.



Gambar 1. Diagram Alir Penelitian

A. Dataset

Gambar 2. Sampel Citra dari Masing-masing *Class*

Data yang digunakan pada penelitian ini merupakan dataset publik yang berasal dari kaggle dengan judul Mushroom Classification Dataset berupa citra jamur, kami

menggunakan 3 jenis jamur yaitu Amanita, Boletus, dan Lactarius. Data citra kami bagi menjadi 3 bagian yaitu *train*, *test*, dan *validation*, dengan pembagiannya: 1979 data *train*, 493 data *validation*, dan 738 data *test*, dengan total 3,210 data gambar RGB. Sample dari masing masing *class* dapat dilihat pada Gambar 2.

Eksperimen dalam penelitian ini dilakukan menggunakan konfigurasi *hardware* dan *software* yang spesifik untuk memastikan reproduksibilitas dan konsistensi hasil. Spesifikasi sistem yang digunakan disajikan dalam Tabel 1.

TABEL I
SPESIFIKASI HARDWARE DAN SOFTWARE

Software	Hardware
Microsoft Windows 11 Operating System, Python 3.12.6, TensorFlow 2.18.0, Matplotlib, NumPy, Scikit-learn	Processor**:** AMD Ryzen 5 5600G (6 Core, 12 Thread, Base Clock 3.9 GHz) RAM: 32GB SSD: 1TB

B. Preprocessing Data

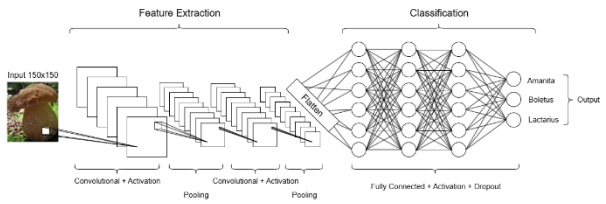
Tahapan analisis dalam penelitian ini dimulai dengan eksplorasi data (*Exploratory Data Analysis*/EDA) untuk memahami distribusi dan hubungan antar variabel, serta memastikan data yang digunakan dalam model klasifikasi jamur sudah bersih dan siap digunakan. EDA ini membantu untuk memperoleh wawasan mengenai karakteristik data, seperti jumlah gambar per kelas, ukuran gambar, serta sebaran data yang mungkin mempengaruhi performa model [4]

C. Augmentasi Data

Augmentasi data dan normalisasi dilakukan untuk meningkatkan variasi data yang dapat membantu model dalam generalisasi yang lebih baik [5]. Augmentasi data seperti rotasi, pergeseran, dan pembalikan horizontal digunakan untuk menciptakan variasi dari gambar yang sama, sedangkan normalisasi dilakukan untuk menyesuaikan rentang nilai *pixel* gambar agar seragam dan lebih efisien dalam pemrosesan oleh model [6]. Selain itu, penanganan kelas yang tidak seimbang juga dilakukan dengan menggunakan pembagian dataset yang tepat antara subset pelatihan, validasi, dan pengujian untuk memastikan model dapat dievaluasi dengan data yang representatif dan menghindari *overfitting* [7].

D. Pengembangan Arsitektur CNN

Gambar 3 mengilustrasikan arsitektur model CNN yang diimplementasikan dalam penelitian ini.



Gambar 3. Arsitektur CNN yang Digunakan dalam Penelitian

Berikut adalah detail arsitektur CNN MobileNet V2 dengan *custom layers* berdasarkan implementasi kode yang telah dibuat yang ditunjukkan pada tabel 2.

TABEL II
DETAIL ARSITEKTUR CNN YANG DIGUNAKAN

Layer	Input Shape	Output Shape	Keterangan
Input Layer	(?, 150, 150, 3)	(?, 150, 150, 3)	Input gambar RGB 150x150
MobileNet V2 Base	(?, 150, 150, 3)	(?, 5, 5, 1280)	Pretrained MobileNet V2 (tanpa top layer)
Conv2D (256 filters)	(?, 5, 5, 1280)	(?, 5, 5, 256)	Kernel 3x3, ReLU, padding='same'
MaxPooling2D	(?, 5, 5, 256)	(?, 2, 2, 256)	Pool size 2x2
Conv2D (512 filters)	(?, 2, 2, 256)	(?, 2, 2, 512)	Kernel 3x3, ReLU, padding='same'
MaxPooling2D	(?, 2, 2, 512)	(?, 1, 1, 512)	Pool size 2x2
GlobalMaxPooling2D	(?, 1, 1, 512)	(?, 512)	Pooling global untuk flatten
Dropout (0.3)	(?, 512)	(?, 512)	Regularisasi 30%
Dense (64 units)	(?, 512)	(?, 64)	Fully connected, ReLU
Dropout (0.3)	(?, 64)	(?, 64)	Regularisasi 30%
Dense (64 units)	(?, 64)	(?, 64)	Fully connected, ReLU
Dropout (0.3)	(?, 64)	(?, 64)	Regularisasi 30%
Dense (64 units)	(?, 64)	(?, 64)	Fully connected, ReLU
Dropout (0.3)	(?, 64)	(?, 64)	Regularisasi 30%
Output Dense	(?, 64)	(?, 3)	Softmax untuk 3 kelas

Model ini menggunakan teknik yang disebut Transfer Learning, yaitu memanfaatkan MobileNetV2 yang sudah dilatih sebelumnya untuk mengenali gambar umum, kemudian disesuaikan untuk mengenali jamur. Sebagian besar bagian dari MobileNetV2 dibekukan (tidak diubah), hanya 4 layer terakhir yang disesuaikan dengan data jamur. Di atas MobileNetV2, peneliti menambahkan beberapa layer tambahan seperti 2 layer konvolusi untuk ekstraksi fitur yang lebih spesifik, 3 layer dense (fully connected) dengan 64 neuron masing-masing untuk klasifikasi, dan layer dropout dengan tingkat 0.3 di antara setiap layer dense untuk mencegah overfitting. Terakhir, output layer menggunakan 3 neuron dengan aktivasi softmax untuk mengklasifikasikan 3 jenis genus jamur.

E. Transfer Learning

Transfer learning adalah teknik *dalam machine learning* yang memanfaatkan model yang telah dilatih sebelumnya pada dataset besar untuk diterapkan pada tugas baru yang serupa [8]. *Transfer learning* merupakan pendekatan yang sangat efektif dalam mengatasi tantangan terkait dataset kecil dan sumber daya komputasi yang terbatas, khususnya dalam

domain pencitraan medis dimana data berlabel sangat langka dan mahal untuk diperoleh [9]. Pendekatan ini sangat efektif dalam *computer vision* karena fitur-fitur tingkat rendah yang dipelajari dari dataset besar seperti ImageNet dapat digunakan untuk berbagai tugas klasifikasi gambar yang berbeda [10].

Transfer learning dengan CNN bertujuan untuk meningkatkan performa pada tugas baru dengan memanfaatkan pengetahuan dari tugas serupa yang telah dipelajari sebelumnya [11]. Hal ini memberikan kontribusi besar dalam analisis citra medis karena mengatasi masalah kelangkaan data sekaligus menghemat waktu dan sumber daya komputasi [9]. Proses *transfer learning* dapat dilakukan dengan dua pendekatan utama: *feature extraction*, dimana *layer-layer* awal dari model *pre-trained* digunakan sebagai *feature extractor* yang tetap, dan *fine-tuning*, dimana seluruh atau sebagian parameter model *pre-trained* disesuaikan dengan dataset baru [12]. Studi komparatif menunjukkan bahwa *transfer learning* meningkatkan akurasi semua model CNN, dengan peningkatan yang bervariasi tergantung pada arsitektur yang digunakan [13].

F. Convolution Neural Network (CNN)

Convolutional Neural Network adalah jenis arsitektur *deep learning* yang secara khusus dirancang untuk mengolah data yang memiliki struktur *grid* seperti gambar [14]. CNN menggunakan operasi konvolusi untuk mengekstrak fitur hierarkis dari data *input*, dimulai dari fitur-fitur dasar seperti tepi dan tekstur pada lapisan awal, hingga fitur-fitur yang lebih kompleks pada lapisan yang lebih dalam. Keunggulan utama CNN terletak pada kemampuannya untuk menangani data visual dengan efisien melalui penggunaan filter konvolusi yang dapat mendeteksi pola lokal dalam gambar, *pooling layer* untuk reduksi dimensi, dan *fully connected layer* untuk klasifikasi akhir [15].

Arsitektur CNN terdiri dari beberapa komponen utama: lapisan konvolusi yang berfungsi sebagai *feature extractor*, *lapisan pooling* untuk *downsampling*, lapisan aktivasi untuk non-linearitas, dan lapisan *fully connected* untuk klasifikasi [16]. Setiap lapisan konvolusi menggunakan sejumlah *filter* yang dipelajari selama proses *training* untuk mengidentifikasi fitur-fitur tertentu dalam gambar. Proses pembelajaran ini memungkinkan CNN untuk secara otomatis mengenali pola-pola yang relevan untuk tugas klasifikasi yang diberikan [17]. Dalam beberapa tahun terakhir, paradigma komputasi *deep learning* telah dianggap sebagai standar emas dalam komunitas *machine learning*, dan secara bertahap menjadi pendekatan komputasi yang paling banyak digunakan dalam bidang ML, sehingga mencapai hasil luar biasa pada beberapa tugas kognitif kompleks [18].

G. MobileNetV2

MobileNetV2 merupakan arsitektur CNN yang dioptimalkan untuk aplikasi mobile dan *embedded systems* dengan fokus pada efisiensi komputasi dan ukuran model yang kompak. Arsitektur ini dikembangkan oleh Sandler et al. (2018) [19] sebagai perbaikan dari MobileNetV1 dengan memperkenalkan dua konsep utama*: inverted residuals* dan *linear bottlenecks.*

Inverted residuals berbeda dengan *residual block* tradisional dimana ekspansi dilakukan pada awal blok, diikuti dengan *depthwise convolution*, dan kemudian kompresi melalui *pointwise convolution. Linear bottlenecks* menghilangkan fungsi aktivasi non-linear pada *output bottleneck* untuk mempertahankan informasi yang penting. Kombinasi kedua teknik ini memungkinkan MobileNetV2 untuk mencapai akurasi yang tinggi dengan jumlah parameter yang sangat sedikit dibandingkan dengan arsitektur CNN konvensional.

Keunggulan MobileNetV2 terletak pada penggunaan *depthwise separable convolutions* yang memisahkan operasi konvolusi spasial dan *channel-wise*, sehingga secara dramatis mengurangi jumlah operasi komputasi yang diperlukan. Penelitian terbaru menunjukkan efektivitas MobileNetV2 dalam berbagai aplikasi praktis, termasuk deteksi COVID-19 menggunakan citra X-ray dada dengan tingkat akurasi yang tinggi [20], dan implementasi dengan CapsNet yang menunjukkan akurasi klasifikasi hingga 96,58% dengan waktu eksekusi yang sangat efisien yaitu 0,58 detik [21]. Arsitektur ini juga menggunakan *width multiplier* dan *resolution multiplier* sebagai *hyperparameter* untuk mengontrol *trade-off* antara akurasi dan efisiensi, memungkinkan adaptasi model sesuai dengan keterbatasan *resources* pada perangkat target [19].

H. Evaluasi

Implementasi model klasifikasi akan menghasilkan pola atau *rule* yang berfungsi sebagai dasar dalam melakukan prediksi nilai. Sebelum digunakan untuk prediksi, *rule* tersebut harus melalui tahap evaluasi dan validasi guna mengetahui tingkat akurasi hasil prediksi yang akan dihasilkan. Proses evaluasi dan validasi terhadap *rule* klasifikasi dilakukan dengan menggunakan *Confusion Matrix* [22].

TABEL III
REPRESENTASI CONFUSION MATRIX

Correct Classification	**Classified as**	
	+	-
+	True Positives (A)	False Negatives (B)
-	False Positives (C)	True Negatives (D)

Confusion Matrix merupakan teknik yang digunakan dalam bidang *data mining* untuk mengukur tingkat akurasi. Proses evaluasi yang menggunakan *Confusion Matrix* akan menghasilkan nilai akurasi, presisi, dan recall. Akurasi menunjukkan persentase data yang berhasil diklasifikasikan dengan tepat setelah melalui tahap pengujian. Presisi mengukur rasio kasus yang diprediksi positif dan juga benar-benar positif pada data aktual [23]. Sementara itu, recall mengukur rasio kasus positif yang benar-benar berhasil diprediksi sebagai positif. Rumus (1) akurasi dengan

menggunakan tabel *Confusion Matrix* dilakukan dengan rumus berikut:

$$Akurasi = \frac{(A+D)}{(A+B+C+D)} X100\% \quad (1)$$

Presisi merupakan perbandingan antara item relevan yang terpilih dengan total keseluruhan item yang dipilih. Presisi dapat diartikan sebagai tingkat kesesuaian jawaban yang diberikan terhadap permintaan informasi. Rumus (2) presisi adalah:

$$Presisi = \frac{A}{(C+A)} X100\% \quad (2)$$

Recall merupakan perbandingan antara item relevan yang terpilih dengan keseluruhan jumlah item relevan yang ada. Recall dihitung dengan rumus (3):

$$Recall = \frac{A}{(A+D)} X100\% \quad (3)$$

III. HASIL DAN PEMBAHASAN

Bab ini menyajikan analisis komprehensif terhadap hasil eksperimen yang telah dilakukan. Pembahasan difokuskan pada tiga aspek utama: kinerja proses pelatihan model, analisis mendalam terhadap mitigasi *overfitting* yang menjadi tujuan utama penelitian, dan evaluasi kinerja generalisasi model pada data uji yang belum pernah dilihat sebelumnya. Hasil yang diperoleh kemudian dikontekstualisasikan melalui perbandingan dengan penelitian-penelitian terdahulu untuk menegaskan kontribusi ilmiah dari penelitian ini.

A. Augmentasi Data

Dalam penelitian ini, augmentasi data dilakukan untuk meningkatkan keberagaman data pelatihan dan mencegah model dari *overfitting*. Teknik Augmentasi yang dilakukan adalah:

1) *Rescaling* (Normalisasi): Semua *pixel* pada gambar diubah skala nilainya agar berada dalam rentang [0, 1].
2) Rotasi Gambar: Gambar dapat diputar secara acak dalam kisaran 90 derajat.
3) Pergeseran Horizontal dan Vertikal: Gambar dapat digeser secara acak baik secara horizontal maupun vertikal dalam kisaran 10% dari ukuran gambar.
4) *Shear Transformation*: Transformasi shear (geser bentuk gambar) yang dapat memperkenalkan distorsi pada gambar secara acak.
5) *Zoom*: Gambar dapat di-*zoom* secara acak dalam kisaran 50% dari ukuran gambar asli.
6) Pembalikan Horizontal: Gambar dibalik secara horizontal (*flip*).
7) Pengisian Gambar: Digunakan untuk mengisi ruang kosong yang ditinggalkan oleh transformasi (misalnya saat rotasi atau pergeseran gambar). Metode pengisian ini menggunakan nilai piksel terdekat untuk mengisi ruang kosong tersebut.

TABEL IV
RANGKUMAN METRIK KINERJA MODEL PADA EPOCH KUNCI

Epoch	Akurasi Training	Loss Training	Akurasi Validasi	Loss Validasi	Keterangan
1	0.3372	1.6495	0.4266	1.0703	Awal Pelatihan
15	0.5954	0.8454	0.6993	0.6946	Pertengahan Pelatihan
34	**0.7735**	**0.5731**	**0.7879**	**0.5722**	**Epoch Optimal (Best Model)**
39	0.8024	0.5404	0.7716	0.6101	Pelatihan Dihentikan

B. Kinerja Proses Pelatihan Model

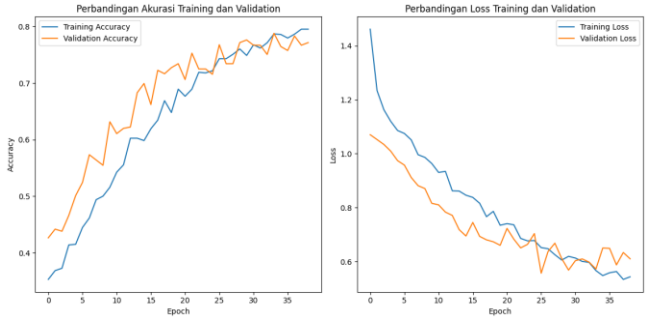
Model CNN berbasis MobileNetV2 yang di-*fine-tune* menggunakan arsitektur tambahan (Conv2D, pooling, GlobalMaxPooling, dan beberapa Dense + Dropout) diawali dengan sebuah *Input Layer* berukuran (150, 150, 3). Lapisan berikutnya memanfaatkan MobileNetV2 sebagai basis *transfer learning*, menghasilkan representasi fitur dengan ukuran (5, 5, 1280). Setelah itu, dilanjutkan dengan Conv2D yang memiliki *output* (5, 5, 256), diikuti MaxPooling2D yang mengecilkan dimensi menjadi (2, 2, 256). Tahapan berikutnya adalah Conv2D tambahan yang mengolah representasi menjadi (2, 2, 512), lalu diperkecil lagi dengan MaxPooling2D menjadi (1, 1, 512). Selanjutnya, GlobalMaxPooling2D digunakan untuk mengubah representasi fitur tersebut ke dalam vektor berdimensi 512. Model ini juga mengintegrasikan beberapa lapisan Dropout untuk mencegah *overfitting* serta lapisan Dense dengan neuron berturut-turut sebanyak 64, 64, dan 64 sebelum akhirnya menghasilkan output akhir berupa Dense layer berukuran 3, yang sesuai dengan jumlah kelas prediksi yang diinginkan. Total parameter dalam arsitektur ini mencapai 6.428.867, dengan 4.583.683 parameter dapat dilatih (*trainable*) dan 1.845.184 parameter tidak dapat dilatih (*non-trainable*).

Kinerja model selama tahap ini dipantau secara ketat untuk memastikan proses pembelajaran berjalan optimal dan konvergen menuju solusi yang baik. Model dilatih dengan 100 epoch yang menggunakan *optimizer* Adam dengan laju pembelajaran (*learning rate*) awal sebesar 1.0×10−5. Untuk mengontrol proses pelatihan dan mencegah *overfitting*, dua teknik utama diimplementasikan: *EarlyStopping* dan *ModelCheckpoint*. *EarlyStopping* dikonfigurasi untuk menghentikan pelatihan jika metrik val_loss (kerugian pada data validasi) tidak menunjukkan perbaikan minimal selama tiga epoch berturut-turut. Sementara itu, *ModelCheckpoint* bertugas menyimpan bobot model terbaik berdasarkan val_loss terendah yang tercapai selama pelatihan.

Berdasarkan log pelatihan yang dihasilkan, proses pelatihan dihentikan secara otomatis pada epoch ke-39. Namun, mekanisme *EarlyStopping*, yang dikonfigurasi dengan parameter *restore_best_weights=True*, secara cerdas mengidentifikasi bahwa kinerja optimal model tidak berada pada epoch terakhir, melainkan pada epoch ke-34. Pada titik ini, model mencapai nilai val_loss terendah. Oleh karena itu, bobot dari epoch ke-34 inilah yang dipulihkan dan digunakan sebagai model *final* untuk evaluasi lebih lanjut.Analisis kuantitatif dari Tabel 4 memperkuat observasi visual. Pada epoch ke-34, yang merupakan titik kinerja puncak, model mencatatkan akurasi pelatihan sebesar 77.35% dengan *loss* 0.5731, sementara akurasi validasi mencapai 78.79% dengan *loss* 0.5722. Fakta bahwa kinerja validasi sedikit lebih unggul daripada kinerja pelatihan merupakan indikator yang sangat kuat bahwa model tidak mengalami *overfitting*.

Perbandingan kurva menunjukkan bahwa kedua metrik (akurasi dan *loss*) pada data validasi mengikuti pola yang serupa dengan data pelatihan sepanjang proses, tanpa adanya peningkatan *loss* bersamaan dengan stagnasi atau penurunan tajam akurasi pada validasi. Hal ini menandakan tidak terjadinya *overfitting* yang signifikan karena perbedaan akurasi di bawah ambang toleransi 5%.

Visualisasi dari proses pelatihan, seperti yang ditunjukkan pada Gambar 4 (Perbandingan Akurasi *Training* dan *Validation*) dan (Perbandingan *Loss Training* dan *Validation*), memberikan bukti empiris pertama mengenai kesehatan model.



Gambar 4. Akurasi dan *Loss* pada Data *Training* vs *Validation*

C. *Analisis Mendalam Mitigasi Overfitting*

Tujuan utama penelitian ini adalah mengembangkan model CNN yang robust dan mampu melakukan generalisasi dengan baik pada data baru. Fokus utama diarahkan pada mitigasi masalah *overfitting*, yaitu fenomena dimana model menjadi terlalu "hafal" dengan data pelatihan sehingga kinerjanya menurun drastis ketika dihadapkan pada data yang belum pernah dilihat sebelumnya.

Bagian ini akan membuktikan secara rigor bahwa model yang dikembangkan berhasil memitigasi *overfitting* melalui analisis faktor-faktor penyebabnya. *Overfitting* secara kuantitatif dapat diidentifikasi melalui "celah generalisasi" (*generalization gap*), yaitu selisih signifikan antara metrik kinerja pada data pelatihan dan data validasi. Sebuah model dikatakan *overfit* ketika *training_loss* << *validation_loss* dan *training_accuracy* >> *validation_accuracy*.

Penelitian rujukan [2] menunjukkan gejala *overfitting* dengan akurasi pelatihan 89% dan akurasi validasi 82%, menghasilkan celah generalisasi sebesar 7%. Kondisi ini mengindikasikan bahwa model tersebut mempelajari pola-pola spesifik pada data pelatihan yang tidak dapat digeneralisasi ke data validasi.

Sebaliknya, model yang dikembangkan dalam penelitian ini menunjukkan karakteristik yang berbeda secara signifikan. Pada epoch optimal (epoch 34), model mencapai akurasi pelatihan 77.35% dan akurasi validasi 78.79%, menghasilkan celah generalisasi sebesar 77.35% − 78.79% = −1.44%. Celah generalisasi negatif ini merupakan bukti kuat yang menentang adanya *overfitting*, menunjukkan bahwa kemampuan prediksi model sama baiknya, atau bahkan lebih baik, pada data yang tidak digunakan untuk melatihnya.

Keberhasilan mitigasi *overfitting* ini dapat diatribusikan pada dua faktor utama: efisiensi arsitektur MobileNetV2 yang telah dioptimalkan untuk generalisasi dan penerapan teknik regularisasi secara eksplisit melalui *multiple dropout layers* dan *early stopping mechanism*.

D. *Evaluasi Kinerja Generalisasi pada Data Uji*

Tahap akhir dari evaluasi adalah menguji model *final* pada set data uji (*test set*), yang terdiri dari 738 citra yang sama sekali tidak pernah digunakan selama proses pelatihan maupun validasi. Pengujian ini merupakan ujian sesungguhnya terhadap kemampuan generalisasi model.

Model *final*, dengan bobot yang dipulihkan dari epoch ke-34, dievaluasi pada set data uji dan berhasil mencapai akurasi akhir sebesar 76.02%.

Untuk memvalidasi robustisitas model, penting untuk membandingkan kinerjanya secara konsisten di ketiga set data: pelatihan, validasi, dan uji.

1) Akurasi Pelatihan (pada epoch 34): 77.35%
2) Akurasi Validasi (pada epoch 34): 78.79%
3) Akurasi Uji (*final*): 76.02%
4) Presisi Uji (*final*): 80,6%
5) Recall Uji (final): 68,1%

Ketiga nilai akurasi ini berada dalam rentang yang sangat sempit, dengan selisih maksimal hanya sekitar 2.77 poin persentase (antara akurasi validasi tertinggi dan akurasi uji). Konsistensi yang luar biasa ini adalah validasi akhir dari tujuan penelitian. Ini membuktikan bahwa kinerja yang diamati selama fase validasi bukanlah suatu kebetulan, melainkan cerminan sejati dari kemampuan model untuk melakukan generalisasi pada data baru yang tidak dikenal. Model ini dapat diandalkan, dan kinerjanya dapat diprediksi

E. Analisis Komparatif dan Kontekstualisasi Hasil

Dalam konteks wacana ilmiah, kontribusi sebuah penelitian tidak hanya diukur dari satu metrik tunggal seperti akurasi. Argumen utama dari penelitian ini adalah bahwa pencapaian model yang robust dan tidak *overfit* merupakan kontribusi yang lebih signifikan daripada sekadar mencapai angka akurasi yang tinggi namun rapuh.

Akurasi sebesar 76.02% yang dicapai bukanlah sebuah keterbatasan, melainkan hasil dari sebuah *trade-off* yang berhasil. Model ini secara sengaja "menolak" untuk mempelajari derau (*noise*) dalam data pelatihan—sebuah proses yang mungkin dapat mendorong akurasi pelatihannya lebih tinggi (misalnya, ke 85-90%)—demi mendapatkan kinerja yang kuat dan konsisten pada data yang belum pernah dilihat. Pendekatan ini secara langsung menjawab kelemahan yang diidentifikasi dalam Bab Pendahuluan, seperti penelitian oleh Rahmadhani & Marpaung (2023) [2] dengan celah generalisasi 7%, dan penelitian oleh Farhan (2024) [1] yang melaporkan akurasi 100%, yang merupakan indikator kuat dari *overfitting* yang parah.

TABEL V
PERBANDINGAN KINERJA MODEL DENGAN PENELITIAN TERDAHULU

Penelitian (Studi)	Arsitektur	Akurasi Training	Akurasi Validasi /Uji	Celah Generalisasi	Keterangan
Penelitian Ini	CNN + MobileNetV2	77.35 %	76.02%	1.33%	Model Robust, Tidak Overfitting
Rahmadhani & Marpaung (2023)	CNN	89.00 %	82.00%	7.00%	Terindikasi Overfitting

Tabel 5 menyajikan perbandingan yang kuat secara visual. Celah generalisasi yang minimal (bahkan negatif jika dibandingkan dengan data validasi) dari penelitian ini sangat kontras dengan celah yang signifikan pada penelitian rujukan. Hal ini memposisikan hasil penelitian ini sebagai hasil yang lebih unggul dari segi ketelitian ilmiah dan keandalan model, yang merupakan fondasi penting sebelum upaya peningkatan akurasi lebih lanjut dilakukan.

IV. KESIMPULAN

Penelitian ini telah berhasil mengembangkan dan memvalidasi sebuah model *Convolutional Neural Network* (CNN) berbasis arsitektur MobileNetV2 untuk tugas klasifikasi tiga jenis jamur. Tujuan utama penelitian, yaitu untuk membangun sebuah model yang tahan terhadap *overfitting*, telah tercapai dengan sukses. Hal ini dibuktikan secara kuantitatif melalui celah generalisasi yang minimal antara akurasi pelatihan (77.35%) dan akurasi validasi (78.79%), dan dikonfirmasi lebih lanjut oleh kinerja yang kuat dan konsisten pada data uji dengan akurasi akhir sebesar 76.02% dengan presisi 80,6% dan recall 68,1%.

Keberhasilan ini diatribusikan pada efek sinergis dari dua komponen utama: Penggunaan arsitektur MobileNetV2 yang secara inheren efisien dalam hal parameter dan dirancang untuk mencegah *overfitting*, dan penerapan teknik regularisasi secara eksplisit, terutama *EarlyStopping*, yang memastikan pemilihan model pada titik generalisasi optimalnya. Penelitian ini menyimpulkan bahwa mengorbankan beberapa poin persentase pada akurasi pelatihan untuk mendapatkan keuntungan signifikan dalam hal robustisitas, keandalan, dan kemampuan generalisasi model adalah sebuah strategi yang bernilai dan sahih secara ilmiah. Model yang dihasilkan merupakan fondasi yang kuat untuk pengembangan lebih lanjut.

Hasil evaluasi menunjukkan bahwa model mencapai tingkat akurasi 76.02% pada fase pengujian dengan konsistensi yang baik antara data pelatihan, validasi, dan uji, mengindikasikan model tidak mengalami overfitting. Ketika diterapkan dalam dunia nyata, model mampu melakukan klasifikasi gambar jamur berdasarkan kategori Genus dengan memberikan prediksi kelas yang tepat. Model menunjukkan presisi tinggi (80.6%) yang mengindikasikan pendekatan konservatif dalam klasifikasi, meskipun recall (68.1%) masih perlu ditingkatkan. Meskipun demikian, penelitian ini masih memiliki beberapa keterbatasan dan membutuhkan pengembangan lebih lanjut. Untuk meningkatkan performa model, dapat dilakukan beberapa pendekatan seperti memperbanyak dataset atau mengganti dataset gambar dengan kualitas yang lebih baik. Selain itu, implementasi arsitektur CNN yang lebih modern dan efisien seperti EfficientNetV2 atau Xception, dan juga mengkombinasikan beberapa model CNN dengan arsitektur berbeda seperti MobileNetV2, EfficientNet, dan ResNet menggunakan teknik *ensemble learning* seperti *voting*, *bagging*, atau *stacking* untuk meningkatkan akurasi dan recall secara bersamaan.

DAFTAR PUSTAKA

[1] A. M. Al Farhan, “Identifikasi Jenis Jamur Menggunakan Convolutional Neural Network dan Random Forest,” *Universitas PGRI Madiun*, pp. 550–559, 2024.

[2] U. Sri Rahmadhani and N. Lysbetti Marpaung, "Klasifikasi Jamur Berdasarkan Genus Dengan Menggunakan Metode CNN," *Jurnal Informatika: Jurnal Pengembangan IT*, vol. 8, no. 2, 2023.

[3] E. Iedfitra Haksoro and A. Setiawan, "Pengenalan Jamur yang Dapat Dikonsumsi Menggunakan Metode Transfer Learning pada Convolutional Neural Network," Online, 2021. doi: 10.31961/eltikom.v5i2.428.

[4] R. Indrakumari, T. Poongodi, and S. R. Jena, "Heart Disease Prediction using Exploratory Data Analysis," in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 130–139. doi: 10.1016/j.procs.2020.06.017.

[5] B. Zhang, J. Lin, L. Du, and L. Zhang, "Harnessing Data Augmentation and Normalization Preprocessing to Improve the Performance of Chemical Reaction Predictions of Data-Driven Model," *Polymers (Basel)*, vol. 15, no. 9, May 2023, doi: 10.3390/polym15092224.

[6] R. Hao, K. Namdar, L. Liu, M. A. Haider, and F. Khalvati, "A Comprehensive Study of Data Augmentation Strategies for Prostate Cancer Detection in Diffusion-Weighted MRI Using Convolutional Neural Networks," *J Digit Imaging*, vol. 34, no. 4, pp. 862–876, Aug. 2021, doi: 10.1007/s10278-021-00478-7.

[7] M. Sivakumar, S. Parthasarathy, and T. Padmapriya, "Trade-off between training and testing ratio in machine learning for medical image processing," *PeerJ Comput Sci*, vol. 10, 2024, doi: 10.7717/PEERJ-CS.2245.

[8] F. Zhuang *et al.*, "A Comprehensive Survey on Transfer Learning," Jun. 2020, [Online]. Available: http://arxiv.org/abs/1911.02685

[9] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros, and T. Ganslandt, "Transfer learning for medical image classification: a literature review," Dec. 01, 2022, *BioMed Central Ltd*. doi: 10.1186/s12880-022-00793-7.

[10] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Adv Neural Inf Process Syst*, vol. 22, Nov. 2014, [Online]. Available: http://arxiv.org/abs/1411.1792

[11] T. Rahman *et al.*, "Transfer learning with deep Convolutional Neural Network (CNN) for pneumonia detection using chest X-ray," *Applied Sciences (Switzerland)*, vol. 10, no. 9, May 2020, doi: 10.3390/app10093233.

[12] J. A. Raj, L. Qian, and Z. Ibrahim, "Fine-tuning -- a Transfer Learning approach," Nov. 2024, [Online]. Available: http://arxiv.org/abs/2411.03941

[13] X. Du, Y. Sun, Y. Song, H. Sun, and L. Yang, "A Comparative Study of Different CNN Models and Transfer Learning Effect for Underwater Object Classification in Side-Scan Sonar Images," *Remote Sens (Basel)*, vol. 15, no. 3, Feb. 2023, doi: 10.3390/rs15030593.

[14] I. D. Mienye, T. G. Swart, G. Obaido, M. Jordan, and P. Ilono, "Deep Convolutional Neural Networks in Medical Image Analysis: A Review," Mar. 01, 2025, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/info16030195.

[15] Y. Lecun, E. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," 1998. doi: 10.1109/5.726791.

[16] Purwono, A. Ma'arif, W. Rahmaniar, H. I. K. Fathurrahman, A. Z. K. Frisky, and Q. M. U. Haq, "Understanding of Convolutional Neural Network (CNN): A Review," *International Journal of Robotics and Control Systems*, vol. 2, no. 4, pp. 739–748, 2022, doi: 10.31763/ijrcs.v2i4.888.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017, doi: 10.1145/3065386.

[18] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Dec. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[20] Y. Kaya and E. Gürsoy, "A MobileNet-based CNN model with a novel fine-tuning mechanism for COVID-19 infection detection," May 01, 2023, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s00500-022-07798-y.

[21] J. Zhang, X. Yu, X. Lei, and C. Wu, "A novel CapsNet neural network based on MobileNetVV structure for robot image classification," 2022. doi: 10.3389/fnbot.2022.1007939.

[22] X. Deng, Q. Liu, Y. Deng, and S. Mahadevan, "An improved method to construct basic probability assignment based on the confusion matrix for classification problem," *Inf Sci (N Y)*, vol. 340–341, pp. 250–261, May 2016, doi: 10.1016/j.ins.2016.01.033.

[23] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview," Aug. 2020, doi: 10.48550/arXiv.2008.05756.