Klasifikasi Citra Jenis Pesawat Menggunakan Algoritma CNN Dengan Arsitektur ResNet

Rachmat Dwi Putra
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
rachmatdp@student.telkomuniversity.ac.id

Aditya Firman Ihsan
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
adityaihsan@telkomuniversity.ac.id

Jenis pesawat terbang beragam berdasarkan bentuk, ukuran, dan tujuan penggunaannya, sehingga sulit untuk diklasifikasikan dan menjadikannya sebuah tantangan terutama dalam pengembangan teknologi citra. Dalam penelitian ini, algoritma CNN dengan arsitektur ResNet digunakan untuk mengembangkan sistem klasifikasi citra jenis pesawat. Model dibangun dan diuji pada dataset pribadi yang terdiri dari 4.520 citra dari delapan kelas jenis pesawat. Model dilatih dalam dua skenario (dengan dan tanpa augmentasi) dan dua ukuran batch (16, 32), tiga varian layer ResNet (50, 101, 152) menghasilkan dua belas model. Hasil evaluasi menunjukkan bahwa model terbaik dari ResNet-152 dengan augmentasi mampu mencapai f1-score hingga 0.954, membuktikan bahwa model CNN-ResNet yang dikembangkan dapat mengklasifikasikan citra pesawat secara efektif. Meski begitu, kesalahan masih terjadi pada kelas-kelas yang memiliki kemiripan visual, sehingga dibutuhkan eksperimen lanjutan dengan menguji kombinasi arsitektur dan parameter pelatihan yang lebih optimal untuk meningkatkan akurasi dan kemampuan generalisasi model yang lebih baik.

Kata kunci: Klasifikasi, Citra, Pesawat, CNN, ResNet, Augmentasi

1. Pendahuluan

Latar Belakang

Pesawat terbang memiliki berbagai jenis dengan ciri khas yang berbeda, mulai dari bentuk hingga ukuran, tergantung pada produsen pabrik pembuatnya dan tujuan penggunaannya. Keanekaragaman ini membuat pengenalan jenis pesawat menjadi tugas yang sulit, terutama bagi orang-orang yang sama sekali tidak memiliki latar belakang di dunia penerbangan. Dalam konteks keamanan dan operasional, kemampuan untuk mengenali jenis pesawat dengan cepat, baik yang terparkir di bandara maupun yang melintas di udara, menjadi sangat penting. Pengenalan ini juga dapat membantu mendeteksi pesawat tak dikenal, seperti pesawat militer atau pesawat tanpa izin, yang berpotensi mengancam keamanan wilayah udara suatu negara.

Sementara itu, penerapan teknologi klasifikasi citra berbasis kecerdasan buatan, seperti *Convolutional Neural Networks* (CNN), telah menunjukkan potensi besar dengan kemampuannya dalam melakukan pemrosesan otomatis tanpa pengawasan manusia. Dijelaskan juga bahwa CNN telah berhasil diterapkan dalam berbagai domain, diantaranya identifikasi, pengenalan, deteksi, segmentasi, dan klasifikasi terhadap suatu objek [1][2]. Algoritma CNN, khususnya dengan arsitektur ResNet (*Residual Network*), telah terbukti mampu mengatasi masalah berhasil mengatasi masalah degradasi dan menunjukkan peningkatan akurasi seiring bertambahnya kedalaman jaringan [3][4],

Berbagai penelitian terdahulu menunjukkan keberhasilan CNN dalam tugas klasifikasi citra, termasuk klasifikasi kendaraan, deteksi merek mobil, dan segmentasi pesawat. Sebagai contoh, penelitian [5] menggunakan ResNet-101 untuk segmentasi pesawat dan berhasil meningkatkan akurasi melalui fungsi *focal loss*. Penelitian [6] menggunakan ResNet-152 untuk klasifikasi kendaraan dalam berbagai kondisi pencahayaan dan mencapai akurasi hingga 99,68%. Penelitian lainnya, seperti [7], menunjukkan bahwa ResNet-50 dapat memberikan performa tinggi dengan akurasi mencapai 95% pada tugas klasifikasi citra mobil.

Beberapa penelitian juga telah membahas klasifikasi citra pesawat berdasarkan jenis atau modelnya, baik menggunakan pendekatan *supervised* seperti CNN, ANN, dan SVM, maupun pendekatan *self-supervised*. Namun demikian, sebagian besar penelitian tersebut menggunakan arsitektur CNN yang umum, atau metode *hybrid* tanpa fokus mendalam pada arsitektur CNN-ResNet. Oleh karena itu, penelitian ini dilakukan untuk mengimplementasikan metode CNN dengan variasi arsitektur ResNet dalam membangun sistem klasifikasi jenis pesawat. Penelitian ini diharapkan dapat memberikan kontribusi nyata dalam bidang teknologi penerbangan, seperti pengamanan jalur udara, serta menjadi landasan bagi pengembangan teknologi lebih lanjut, seperti deteksi pesawat tak dikenal, dan sebagainya.

Selain itu, di penelitian ini juga melakukan eksperimen pada tiga variasi *layer* arsitektur ResNet (ResNet-50 *layer*, ResNet-101 *layer*, dan ResNet-152 *layer*. Tujuan memilih 3 variasi *layer* arsitektur ResNet tersebut adalah untuk mengetahui apakah benar variasi *layer* tersebut dapat memberikan performa sistem yang baik. Adapaun di akhir penelitian ini, sistem dievaluasi performa masing-masing model menggunakan metrik evaluasi (*precision*, *recall*, dan *f1-score*) guna mendapatkan model terbaik dan menggunakannya dalam uji coba keberhasilan sistem dalam melakukan klasifikasi citra pesawat. Hasil penelitian ini diharapkan dapat memberikan rekomendasi arsitektur CNN-ResNet terbaik untuk klasifikasi citra jenis pesawat, serta membuka peluang penelitian dan penerapan

teknologi serupa dalam konteks yang lebih luas.

Topik dan Batasannya

Penelitian pembangunan sistem klasifikasi citra jenis pesawat ini dilakukan dengan mengimplementasikan algoritma CNN dengan arsitektur *Residual Network* (ResNet). Arsitektur ResNet dipilih untuk digunakan dibandingkan arsitektur lain dengan tujuan untuk melihat apakah benar arsitektur ResNet dapat meminimalisir banyaknya kekurangan yang dimiliki CNN seperti masalah degradasi, dll.

Penelitian dilakukan dengan 2 skenario, yaitu skenario klasifikasi citra tanpa menggunakan *data augmentation* dan skenario klasifikasi citra dengan menggunakan *data augmentation*. Selain itu, penelitian ini juga terbatas pada penggunaan model dengan 3 variasi jumlah *layer ResNet* dan konfigurasi 2 parameter. Adapun 3 variasi jumlah *layer* ResNet, diantaranya ResNet-50 *layer*, ResNet-101 *layer*, dan ResNet-152 *layer*. Sementara untuk konfigurasi 2 parameternya, yaitu parameter *batch size* sebanyak 16, dan *batch size* sebanyak 32. Sehingga akan ada total 12 model yang dibangun dan dilatih dan masing-masing performanya akan dibandingkan model mana yang terbaik untuk sistem klasifikasi citra jenis pesawat ini. Pembatasan dilakukan dengan tujuan agar menghemat waktu proses yang dibutuhkan dalam membangun sistem, serta adanya keterbatasan kebutuhan perangkat yang dimiliki untuk dipakai dalam membuat dan menjalankan program masih belum cukup memadai untuk menjalankan program yang berat.

Tujuan

Penelitian dilakukan dengan tujuan untuk membangun program klasifikasi citra berbagai jenis pesawat berdasarkan pabrikannya menggunakan *dataset* hasil karya pribadi dan hasil karya foto anggota Komunitas Fotografer Aviasi Indonesia (KFAI), yang telah mendapatkan izin dari pihak komunitas (ketua pengurus KFAI). Dibangunnya program ini juga untuk mengetahui cara mengimplementasikan beberapa model menggunakan algoritma *Convolutional Neural Network* (CNN) dengan berbagai macam varian arsitektur *Residual Network* (ResNet). Disamping itu, tujuan lainnya adalah untuk melihat hasil program dengan implementasi tersebut dapat menghasilkan hasil yang baik ketika menggunakan salah satu model dari yang telah diimplementasikan. Hasil program didapat dengan mengevaluasi model yang dibangun menggunakan metrik evaluasi performansi, diantaranya *precision*, *recall*, dan *f1-score*. Dan ketika hasil evaluasi telah didapatkan, maka dilakukan uji coba sistem untuk melihat keberhasilkan program telah dibangun dengan baik.

Organisasi Tulisan

Jurnal penelitian ini memiliki struktur penulisan terurut yang terdiri dari abstrak, bab pendahuluan, bab studi terkait, bab sistem yang dibangun, bab evaluasi, bab kesimpulan, dan daftar pustaka. Abstrak merupakan ringkasan informasi pembukaan mengenai penelitian yang dilakukan dan dijelaskan di dalam jurnal ini. Dimulai dari bab pertama yaitu pendahuluan yang menyajikan informasi mengenai latar belakang, topik, batasan, dan tujuan penelitian. Lalu dilanjutkan bab kedua yaitu studi terkait yang menyajikan informasi dasar yang penting untuk diketahui mengenai hal-hal yang berkaitan dengan penelitian. Kemudian dilanjutkan bab ketiga yaitu sistem yang dibangun yang menyajikan informasi mengenai alur sistem yang dibangun, dataset yang digunakan untuk penelitian, serta penjelasan eksperimen apa saja yang telah dilakukan dalam penelitian. Pada bab keempat yaitu evaluasi menyajikan informasi mengenai skenario pengujian, hasil eksperimen dan analisanya. Penulisan jurnal diakhir bab kelima yaitu kesimpulan menyajikan informasi mengenai kesimpulan dari penelitian yang telah dilakukan serta harapan dan saran untuk penelitian kedepannya. Sedangkan daftar pustaka merupakan daftar referensi yang digunakan untuk penelitian ini.

2. Studi Terkait

2.1 Klasifikasi Citra

Dalam kehidupan sehari-hari, manusia secara alami sejak lahir belajar untuk dapat melakukan klasifikasi objek melalui indra pencitraannya, yaitu mata. Klasifikasi objek citra ini juga dapat dilakukan oleh mesin komputer dengan melalui banyak proses belajar untuk dapat melakukan klasifikasi citra objek dengan baik. Klasifikasi citra merupakan topik riset klasik yang membahas isu inti dalam pengembangan visi komputer, misalnya deteksi objek, segmentasi, dll. Karena hal inilah, klasifikasi citra sering dijadikan sebagai dasar dalam berbagai bidang pengenalan visual dengan proses fundamental, seperti mengkategorikan gambar berdasarkan konten semantiknya sehingga objek dalam citra dapat dengan mudah dikenali oleh mesin komputer [8][9]. Dalam jurnal penelitian [10][11], klasifikasi citra didefinisikan sebagai proses yang dapat mengelompokkan elemen citra pada kelas-kelas tertentu dimana setiap kelasnya menunjukkan karakter setiap objek untuk dapat mengenalinya.

Mengklasifikasikan citra memiliki tujuan utama untuk mengidentifikasi fitur di dalam citra atau label dari citra-citra yang diberikan. Untuk melakukan klasifikasi citra dapat digunakan 2 pendekatan, pendekatan *supervised* yang menggunakan data terlatih dengan campur tangan manusia dan pendekatan *unsupervised* yang sepenuhnya dijalankan oleh komputer tanpa ada campur tangan manusia [12][13]. Dalam jurnal penelitian [8], dijelaskan bahwa penelitian klasifikasi citra pada awalnya berfokus pada ekstraksi fitur citra tradisional yang kemampuan generalisasi dan portabilitas yang rendah. Hal ini terjadi karena prosesnya lebih menekankan fitur citra yang ditentukan secara manual, sehingga menjadikan komputer untuk dapat melakukan proses citra seperti penglihatan biologis alami manusia. Oleh karena itu dibutuhkan pendekatan klasifikasi citra yang lebih modern seperti *deep learning*. *Deep learning* telah menjadi topik penelitian hangat dalam beberapa tahun terakhir. Adapun, *deep learning* memiliki beberapa metode yang bisa digunakan untuk mengklasifikasikan citra, salah satunya adalah *Convolutional Neural Network* (CNN). CNN merupakan model populer dalam hal studi penelitian

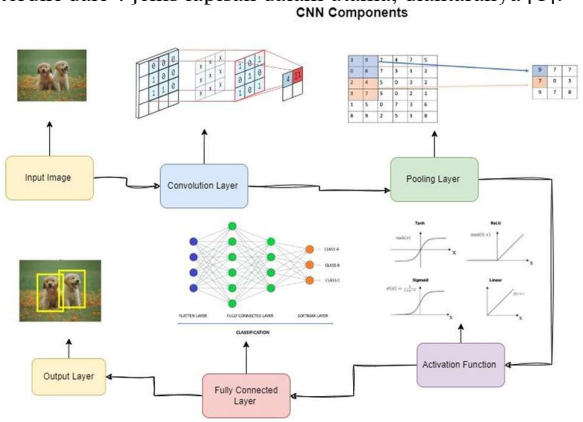
seperti klasifikasi citra yang telah terbukti dapat memberikan hasil yang lebih unggul (*state-of-the-art*) [13][14]. Maka dari itu, CNN dipilih untuk digunakan dalam penelitian klasifikasi citra pesawat yang akan dibahas lebih lanjut dalam jurnal ini.

2.2 Algoritma *Convolutional Neural Network* (CNN)

Convolutional Neural Network atau yang biasa disebut dengan CNN adalah sistem kecerdasan buatan yang termasuk dalam jenis *neural network* berbasis *multi-layer* pada *deep learning*. CNN merupakan sistem yang populer digunakan untuk melakukan berbagai penelitian terkait citra, seperti identifikasi, pengenalan, deteksi, segmentasi, dan klasifikasi terhadap suatu objek dalam suatu citra [1][2]. Dijelaskan dalam jurnal [1], CNN termasuk ke dalam kategori arsitektur *deep learning* yang dapat dipelajari langsung dari objek *input* untuk melakukan ekstraksi fitur otomatis yang bisa tanpa bantuan manual oleh manusia.

Dalam model-model *deep learning* terutama di bidang visi komputer, ekstraksi fitur otomatis yang dapat dilakukan oleh CNN dianggap menjadi suatu keberhasilan yang luar biasa. Selain itu, CNN dapat mempelajari fitur-fitur penting dalam merepresentasikan objek pada suatu citra. Sehingga karena hal-hal inilah, CNN dianggap lebih unggul dibandingkan algoritma klasifikasi citra lain seperti SVM, KNN, *Random Forest*, atau algoritma lainnya [1]. Namun dibalik keunggulan CNN tersebut, CNN memiliki beberapa kekurangan. Pada arsitektur CNN dalam, model-model yang kompleks tentu dibutuhkan. Karena kebutuhan presisi yang tinggi, membuat kebutuhan basis data citra menjadi semakin besar. Terlebih untuk melakukan hal-hal seperti deteksi, pelacakan, pengenalan, ataupun pengkategorian objek yang merupakan contoh dari tugas-tugas dalam visi komputer, CNN memerlukan akses *dataset* berlabel dalam jumlah yang tidak sedikit [1]. Dengan bertambahnya kedalaman arsitektur jaringan, disebutkan juga munculnya kekurangan CNN lainnya dalam jurnal [3], seperti masalah *gradient vanishing*/*explosion*, dan degradasi performa. Masalah *overfitting* juga bisa ditemukan dalam CNN dan bisa menjadi hambatan utama terlebih terjadinya *overfitting* disaat ketika sedang mencoba menciptakan generalisasi yang baik untuk suatu model CNN. Terjadinya *overfitting* digambarkan sebagai situasi suatu model memberikan hasil yang baik pada data pelatihan, akan tetapi berbeda hasilnya pada data pengujian diberikan hasil yang buruk. *Overfitting* adalah hal yang berbeda dengan *underfitting*. *Underfitting* terjadi ketika model dalam pelatihannya tidak cukup belajar ataupun tidak mendapatkan cukup informasi dari data pelatihan [1]. Dijelaskan dalam jurnal [1], dengan meningkatnya jumlah *layer* dalam arsitektur CNN yang bisa mencapai hingga ratusan *layer* dapat membuat arsitektur CNN lebih ringkas dan lebih efektif. Namun hal ini dapat beresiko, karena membuat jaringan arsitektur memiliki miliaran parameter sehingga dalam proses pelatihan model akan membutuhkan banyak sumber data dan sumber daya perangkat keras seperti GPU yang kuat dan canggih. Dengan adanya sumber daya komputasi tinggi GPU yang kuat ini dapat membantu proses pelatihan dan pengujian aneka kombinasi parameter berulang yang memerlukan banyak waktu dapat memberikan hasil yang memuaskan. Akan tetapi ada hal yang perlu diingat juga, bahwa mengatur pemilihan *hyperparameter* harus dilakukan dengan tepat. Karena pemilihan *hyperparameter* ini berdampak besar terhadap performa CNN, dan performa CNN sangat sensitif terhadap hal itu.

Secara umum, contoh struktur umum arsitektur CNN dapat dilihat pada GAMBAR 1. Dan arsitektur ini biasanya terdiri dari 4 jenis lapisan dalam utama, diantaranya [1]:



GAMBAR 1
Ilustrasi Komponen Arsitektur Dasar CNN

1. *Convolutional Layer*

Struktur CNN memiliki sebuah bagian paling penting, yaitu *convolutional layer*. *Convolutional layer* merupakan sekumpulan *filter* atau yang juga bisa disebut dengan *kernel* yang bisa diterapkan pada data sebelum digunakan. Masing-masing *kernel* memiliki bentuk lebar, tinggi, serta bobot yang berbeda-beda dan biasanya digunakan untuk mengekstrak karakteristik data masukan. Bobot dari kernel akan dipilih secara acak di tahap awal, dan akan diperbarui secara bertahap berdasarkan data pelatihan. *Kernel* ini jika digabungkan dengan *input* citra yang direpresentasikan dalam bentuk matriks berdimensi-N akan membentuk *feature map* yang berguna pada berbagai proses pengolahan citra di *layer* berikutnya dalam proses klasifikasi atau deteksi citra dalam konteks CNN.

2. *Pooling Layer*

Pooling layer atau yang juga bisa disebut sebagai *down-sampling layer* merupakan lapisan yang digunakan dalam mengurangi dimensi dari *feature map* dengan sambil mempertahankan data terpenting. Dalam *pooling*, *down-sampling* adalah bagian terpenting yang digunakan dalam mengurangi kompleksitas lapisan atas. Dengan kata lain yaitu pengurangan resolusi dalam pemrosesan citra. Operasi *pooling* terhadap *data input* dilakukan oleh sebuah *filter* dengan menggeser bagian atasnya di dalam lapisan *pooling*. Diantara lapisan-lapisan *pooling*, seperti *max, min*, dan *avg, max pooling* adalah yang paling umum digunakan.

3. *Function of Activation Layer*

Function of activation layer yang juga dapat disebut sebagai lapisan non-linearitas, memungkinkan *output* yang dihasilkan dapat diubah ataupun dihentikan. Sehingga lapisan ini dapat digunakan untuk mengatur atau

membatasi *output* agar tetap dalam rentang yang sesuai dan tidak mengalami saturasi berlebih. Adapun, lapisan non-linearitas ini dalam arsitektur CNN biasanya terletak setelah lapisan-lapisan berbobot atau yang juga biasa disebut sebagai *learnable layer*, seperti *convolutional layer* dan *fully connected layer*. Ada beberapa macam lapisan non-linearitas yang biasa digunakan dalam CNN, diantaranya *Sigmoid, Tanh,* dan *ReLU* (*Rectified Linear Unit*). *ReLU* adalah lapisan non-linearitas yang paling populer digunakan karena memiliki kesedehanaan fungsi dan gradien dibandingkan dengan *Sigmoid* dan *Tanh* yang sering mengalami masalah '*Vanishing Gradient*' dalam *backpropagation*. Selain itu *ReLU* dalam penggunaannya dapat memanfaatkan efisiensi waktu dan sumber daya sehingga hal ini menjadi keuntungan utama mengapa *ReLU* lebih populer untuk digunakan.

4. Fully Connected Layer

Bagian akhir dari struktur lapisan dalam arsitektur CNN adalah *fully-connected layer* yang berfungsi sebagai pengklasifikasi. *Fully-connected layer* merupakan lapisan yang terdiri dari kumpulan *neuron* yang yang saling terhubung dan terkoneksi langsung dengan semua *neuron* dari lapisan di atas dan bawahnya. Namun, dengan banyaknya *neuron* yang terkoneksi dengan *neuron* dari lapisan atas dan bawahnya, menjadikannya sebagai lapisan dalam struktur CNN yang memiliki banyak parameter yang sering digunakan karena rumitnya perhitungan dalam data pelatihan. Sehingga hal ini menjadi sebagai kelemahan karena menyebabkan proses pelatihan memerlukan banyak waktu. Untuk mengatasi hal ini diperlukan peminimalan jumlah koneksi dan *neuron* dengan menggunakan pendekatan *dropout*. *Output* dari *full-connected layer* ini dicapai dalam proses klasifikasi akhir pada *output layer* sebagai lapisan terakhir dalam arsitektur CNN. Di *output layer* tersebut sebelum didapatkan hasil prediksi klasifikasi, digunakan fungsi *loss* terlebih dahulu untuk menghitung prediksi *error* dari data pelatihan yang disertakan dalam proses *backpropagation*. Di proses tersebut, nilai *error* diukur menggunakan *output* dari *layer* sebelumnya. Dan bobot dari setiap *neuron* dalam *layer* akan selalu diperbarui menggunakan nilai *error*.

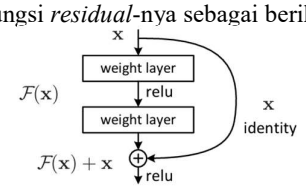
Dengan adanya beberapa kelemahan dalam struktur dasar arsitektur CNN, saat ini CNN telah memiliki beragam arsitektur yang dikembangkan dan dapat digunakan pada beraneka domain aplikasi sesuai seberapa baik fungsinya dapat bekerja, seperti VGG, AlexNet, Xception, Inception, dan ResNet. Dan arsitektur terpopuler untuk digunakan dalam domain klasifikasi, diantaranya ada VGG-16, ResNet, dan Inception. Penelitian modern telah menunjukkan bahwa performa CNN dapat meningkat ketika pendekatan *block* digunakan dibandingkan pendekatan penambahan layer yang semakin dalam demi meningkatkan akurasi. Dengan adanya peningkatan yang dilakukan dalam berbagai penelitian ini, didapatkan hasil CNN yang menjadi lebih meningkat dalam pembelajaran data pada kedalaman yang bervariasi dan ditambah dengan modifikasi beraneka struktural yang berbeda [1]. Dijelaskan dalam jurnal [3][4], arsitektur ResNet dikembangkan untuk mengatasi masalah degradasi yang menjadi kekurangan dari CNN. Hal ini adalah salah satu alasan mengapa arsitektur ResNet dipilih untuk dicoba digunakan dalam mengklasifikasi citra pesawat dalam penelitian ini. Dan penjelasan mengenai arsitektur ResNet akan dijelaskan pada sub-bab selanjutnya.

2.3 *Arsitektur Residual Network* (ResNet)

Residual Network (ResNet) merupakan salah satu arsitektur *Convolutional Neural Network* (CNN) yang dikembangkan oleh Kaiming He [4]. Pengembangan ResNet ini berhasil meraih kemenangan peringkat pertama pada lomba klasifikasi dalam ImageNet *Large Scale Visual Recognition Challenge* (ILSVRC) pada tahun 2015, berkat inovasinya dalam memanfaatkan *residual learning frameworkI*.

Arsitektur ResNet dirancang untuk mengatasi permasalahan optimasi pada jaringan yang sangat dalam, seperti masalah degradasi yang tidak mudah dioptimalkan dalam CNN. Degradasi merupakan masalah yang terjadi bukan disebabkan oleh *overfitting* meskipun ketika kedalaman lapisan semakin bertambah, tetapi tidak membantu menaikkan akurasi dan membuatnya turun dengan cepat. Justru, penambahan lapisan yang dalam ini membuat nilai *error* selama pelatihan menjadi lebih tinggi [4].

Perancangan arsitektur ResNet dibuat dengan menggunakan pendekatan *residual learning* pada beberapa lapisan yang ditumpuk. Pendekatan ini mempertimbangkan blok-blok *residual* yang dirancang dengan memetakan *residual* (*residual mapping*) untuk mempelajari fungsi *residual*. *Residual mapping* yang dipelajari memanfaatkan *shortcut connection* yang merupakan jalur koneksi yang melewati 1 atau banyak lapisan. Koneksi ini digambarkan hanya melakukan pemetaan identitas (*identity mapping*) GAMBAR 2 menunjukkan contoh ilustrasi blok *residual* yang perumusan fungsi *residual*-nya sebagai berikut [4]:



GAMBAR 2
Ilustrasi Arsitektur Blok *Residual*

$$y = F(x, \{W_i\}) + x \quad (1)$$

Keterangan:

- x, y : Vektor *input* x, vektor *output* y
- W : Menunjukkan *weight layer*
- $F(x, \{W\_i\})$: Fungsi *residual*

Dijelaskan dalam jurnal [4], ilustrasi blok *residual* pada GAMBAR 2 merupakan struktur blok *residual* yang memiliki 2 *weight layer* dan fungsi aktivasi *ReLU* dengan rumusmya $F = W_2\sigma(W_1x)$. *Input* awal x (identitas) diproses oleh kedua lapisan tersebut sehingga membentuk fungsi *residual* $F(x)$. Kemudian, hasil $F(x)$ dijumlahkan secara elemen-per-elemen dengan *input* awal x melalui *shortcut connection* yang ditandai sebagai σ tanpa menambah parameter dan menghasilkan fungsi *mapping* $F(x) + x$, lalu menerapkan aktivasi *ReLU* setelahnya.

Namun rumus ini berlaku jika dimensi x dan $F(x)$ yang sama. Jika berbeda karena disebabkan oleh suatu hal, seperti adanya perubahan jumlah *channel input/output*, maka akan dilakukan proyeksi *linear* W melalui *shortcut connection* untuk menyamakan dimensi. Lalu perumusannya akan menjadi seperti berikut:

$$y = F(x, \{W_i\}) + W_s x \quad (2)$$

Keterangan:

- x, y : Vektor *input* x, vektor *output* y
- $F(x, \{W_i\})$: Fungsi *residual*
- $W_s x$: Proyeksi *linear* W melalui *shortcut connection*
- W : Menunjukkan *weight layer*

Dalam eksperimen yang dilakukan oleh para peneliti jurnal [4], *identity mapping* ini dianggap cukup untuk mengatasi masalah degradasi dan lebih efisien. Sementara itu W digunakan ketika hanya saat diperlukan untuk mencocokkan dimensi.

Ekperimen klasifikasi menggunakan *dataset* ImageNet 2012 yang dilakukan peneliti jurnal [4], dilakukan dengan membandingkan keefektifan arsitektur ResNet yang dikembangkan dengan jaringan *plain* tanpa adanya penerapan blok *residual*. Pada jaringan *plain* digunakan 18 *layer* dam 34 *layer*. Hasil yang didapatkan, jaringan *plain* dengan 34 *layer* memiliki nilai *error* yang lebih tinggi selama pelatihan dibandingkan dengan jaringan *plain* 18 *layer*. Peneliti jurnal tersebut berasumsi hal ini diakibatkan oleh masalah degradasi dan adanya masalah *vanishing gradient*. Sementara itu, pada ResNet juga digunakan 18 *layer* dan 34 *layer*. Dan hasil yang didapatkan berbeda dengan jaringan *plain*, ResNet-34 *layer* bekerja lebih baik daripada ResNet-18 *layer* dan menjukkan nilai *error* yang jauh lebih rendah selama pelatihan. Hasil ini membuktikan keefektifan *Residual Learning* pada jaringan yang sangat dalam dan dapat mengatasi masalah degradasi serta memberikan keuntungan akurasi yang bagus. Adapun, eksperimen ResNet juga dilakukan pada jaringan yang lebih dalam dengan menggunakan 50, 101, dan 152 *layer*. Hasil eksperimen ResNet terhadap ketiga jumlah *layer* tersebut ternyata menunjukkan bahwa hasilnya lebih akurat dibandingkan dengan hanya menggunakan ResNet-34 *layer*. Karena adanya peningkatan akurasi dan tidak ditemukannya masalah degradasi pada jaringan ResNet yang semakin dalam, maka disimpulkan bahwa ResNet dianggap mampu bekerja efektif pada jaringan yang semakin dalam setelah dinilai dari perbandingan akurasi dan metrik evaluasi yang meningkat dari jaringan dengan lapisan rendah hingga dalam. Dari hal ini membuat peneliti menjadi tertarik ingin menguji keefektifan ResNet dengan jaringan yang dalam pada penelitian klasifikasi citra jenis pesawat ini.

2.4 Studi Klasifikasi Citra Pesawat

Sistem klasifikasi citra pesawat bertujuan untuk mengidentifikasi jenis atau model pesawat dari data gambar berdasarkan karakteristik visual tertentu. Perancangan sistem dapat digunakan sebagai alat yang membantu mengidentifikasi pesawat secara cepat, terutama dalam hal pengawasan keamanan wilayah udara. Selain itu, kemampuan sistem untuk membedakan jenis pesawat secara akurat juga dapat membantu mendeteksi objek udara yang mencurigakan, seperti pesawat militer tak dikenal atau pesawat tanpa izin terbang. Seiring kemajuan teknologi, berbagai metode telah digunakan untuk membangun sistem klasifikasi ini, seperti metode berbasis ekstraksi fitur manual dan teknik *deep learning end-to-end*. Namun, jumlah penelitian yang secara khusus mengklasifikasikan citra pesawat langsung dari data citra mentah tanpa tahapan ekstraksi fitur eksplisit masih tergolong terbatas.

Dalam jurnal penelitian [15], sistem klasifikasi citra pesawat dibangun menggunakan pendekatan *transfer learning* dengan CNN berbasis arsitektur AlexNet yang telah dilatih pada ImageNet dan kemudian dilakukan *fine-tune* pada *dataset* ARSI-11 dari citra *remote sensing*. Penelitian ini tidak melakukan ekstraksi fitur manual, tetapi menggunakan CNN secara *end-to-end* dan berhasil memperoleh mAP sebesar 97.53%, jauh lebih tinggi dari metode konvensional berbasis SIFT dan HOG.

Berbeda dengan jurnal tersebut, penelitian sistem klasifikasi citra pesawat yang dibangun dalam jurnal [16] menggunakan pendekatan *fine-grained classification* dengan ekstraksi fitur manual, yaitu SIFT yang dikodekan menjadi *Fisher Vector* dan diklasifikasikan menggunakan SVM. *Dataset* FGVC-Aircraft yang digunakan terdiri dari 100 varian pesawat. Meskipun fokusnya serupa, yaitu klasifikasi jenis pesawat, penelitian ini tidak menggunakan metode *end-to-end* karena proses ekstraksi fitur dilakukan secara terpisah sebelum klasifikasi, dan hasil klasifikasi varian menunjukkan akurasi sebesar 48.69%.

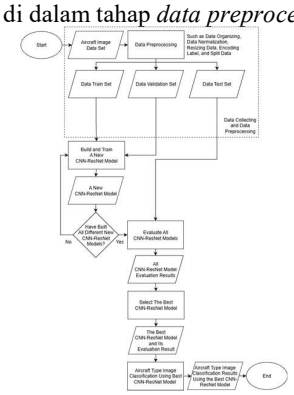
Sementara itu, jurnal penelitian [17] mengusulkan sistem klasifikasi citra pesawat dibangun dengan pendekatan *self-supervised learning* menggunakan *Auto-Encoding Multi-Transformations* (AEMT). *Wide Residual Network* (WRN-28-4) digunakan untuk mempelajari representasi citra melalui tugas awal tanpa label, yakni mengenali berbagai jenis transformasi seperti perubahan kecerahan, blur, dan distorsi perspektif. Representasi yang dihasilkan kemudian dimanfaatkan dalam tahap klasifikasi terawasi menggunakan label citra pesawat. Hasilnya, metode ini mencapai akurasi sebesar 92.79% dan *weighted*-F1 sebesar 92.72%, serta terbukti lebih unggul dibandingkan pendekatan *self-supervised* lainnya seperti RotNet dan Jigsaw.

Dari tiga jurnal yang dibahas, hanya dua di antaranya yang benar-benar menggunakan pendekatan klasifikasi citra pesawat secara langsung dari citra berlabel tanpa ekstraksi fitur manual. Hal ini menunjukkan bahwa penelitian serupa masih tergolong minim, terutama dengan pendekatan murni *end-to-end* menggunakan CNN modern seperti ResNet. Oleh karena itu, penelitian yang dilakukan untuk penulisan jurnal ini memanfaatkan arsitektur CNN ResNet untuk mengklasifikasikan citra pesawat berdasarkan label kelasnya, tanpa melalui proses ekstraksi fitur eksplisit. Selain itu, penelitian ini juga menggunakan dataset baru yang disusun sendiri dan terdiri dari delapan kelas pesawat, sehingga dapat memberikan kontribusi tambahan dalam pengembangan sistem klasifikasi citra berbasis *deep learning* untuk domain penerbangan.

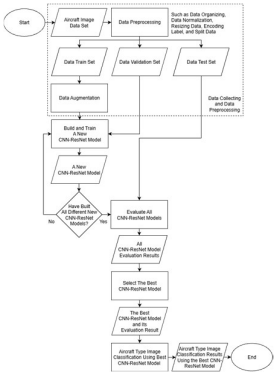
3. Sistem yang Dibangun

3.1 Diagram Alur Sistem

Penelitian dilakukan dengan 2 skenario yang berbeda, dan perbedaan tersebut ada pada pengelolaan data di dalam tahap *data preprocessing*. Skenario pertama (lihat GAMBAR 3) adalah dilakukan uji coba klasifikasi citra pesawat dengan tanpa melakukan *data augmentation* di dalam tahap *data preprocessing*. Sedangkan di skenario kedua (lihat GAMBAR 4), dilakukan uji coba klasifikasi citra pesawat dengan melakukan *data augmentation* pada *data train set* di dalam tahap *data preprocessing*.



GAMBAR 3
Flowchart Klasifikasi Citra Pesawat-Skenario 1



GAMBAR 4
Flowchart Klasifikasi Citra Pesawat-Skenario 2

Untuk penjelasan lebih detailnya, skenario pertama dimulai dari persiapan dataset yang sudah lengkap, dan ketika dataset sudah siap, pengujian masuk ke dalam tahap pertama yaitu *'Data Preprocessing'*. Lalu ketika data selesai diolah di tahap *data preprocessing*, beberapa data akan digunakan di tahap kedua, yaitu *'Build and Train A New CNN-ResNet Model'* dimana pembangunan dan pelatihan semua model baru akan dijalankan. Dan ketika semua model dipastikan sudah terbangun dan terlatih, model-model tersebut akan menjalani evaluasi di dalam tahap *'Evaluate All CNN-ResNet Models'*. Hasil evaluasi model dari tahap tersebut disimpan untuk dilakukan pemilihan model terbaik di dalam tahap *'Select The Best CNN-ResNet Model and its Evaluation Result'*. Setelah didapatkan model terbaik, tahap terakhir dari sistem ini adalah *'Aircraft Type Image Classification Using Best CNN-ResNet Model'*, yaitu menggunakan model terbaik untuk melakukan klasifikasi citra pesawat berdasarkan jenisnya. Kemudian hasil klasifikasinya ditampilkan untuk dianalisis lebih lanjut seberapa baik model terbaik tersebut mampu menjalankan tugas klasifikasi citra pesawat berdasarkan tipenya secara akurat.

Sementara itu, semua tahapan-tahapan yang dilakukan di dalam skenario 2 kurang lebih hampir sama dengan tahapan-tahapan yang juga dilakukan di dalam skenario 1. Perbedaanya ada 1 tahapan tambahan yang terletak di antara tahap '*Data Preprocessing*' dan '*Build and Train A New CNN-ResNet Model*', yaitu tahap '*Data Augmentation*'. Sebenarnya, tahap '*Data Augmentation*' ini masih termasuk bagian dari '*Data Preprocessing*'. Tujuan dari adanya tahap '*Data Augmenation*' ini akan dijelaskan di dalam sub-bab 3.3 Eksperimen bersama dengan penjelasan lengkap untuk semua tahapan yang dilakukan di penelitian pengujian sistem klasifikasi citra jenis pesawat ini.

3.2 Dataset

Dataset yang digunakan dalam penelitian ini adalah kumpulan citra pesawat yang dikelompokkan ke dalam beberapa folder berdasarkan jenis pabrik pembuat pesawat. Folder tersebut merepresentasikan kelas-kelas citra pesawat, sehingga setiap folder mencerminkan satu jenis pabrik pembuat pesawat tertentu.

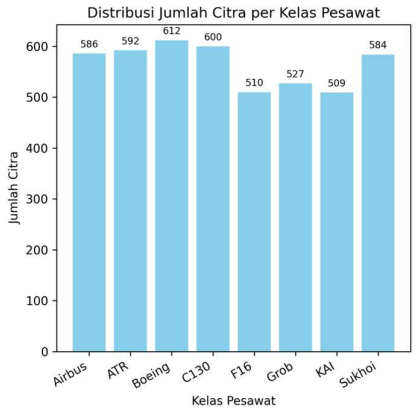
Secara keseluruhan, dataset citra pesawat terdiri dari 8 kelas dengan total 4.520 citra. Detail daftar kelas citra pesawat yang digunakan dalam penelitian ini disajikan pada TABEL 1, yang memuat informasi nama kelas (jenis pesawat), jumlah citra per kelas, dan penjelasan singkat mengenai masing-masing kelas. Untuk memberikan gambaran dan informasi visual citra dalam *dataset* yang akan digunakan dalam pelatihan model klasifikasi, GAMBAR 5 menampilkan masing-masing dua contoh citra dari setiap kelas. Sedangkan GAMBAR 6 memberikan informasi distribusi jumlah citra per kelas dalam bentuk diagram batang.

TABEL 1
Daftar Informasi Kelas Citra Pesawat

No.	Nama Kelas Citra Pesawat	Jumlah Citra	Penjelasan
1.	Airbus	586	Airbus adalah perusahaan dirgantara multinasional asal Eropa yang memproduksi pesawat komersial, helikopter, dan perlengkapan pertahanan. Didirikan pada tahun 1970, Airbus kini menjadi salah satu produsen pesawat terbesar di dunia dan pesaing utama Boeing. Adapun produk populer dari Airbus adalah seri A320, A330, dan A350.
2.	ATR	592	ATR (Aerei da Trasporto Regionale) adalah pesawat penumpang regional turboprop hasil kerja sama

			Prancis dan Italia yang mulai diperkenalkan pada tahun 1985. Dua varian utamanya meliputi seri ATR 42 dan ATR 72. Pesawat ini populer digunakan untuk penerbangan jarak pendek karena efisiensi bahan bakarnya.
3.	Boeing	612	Boeing adalah perusahaan dirgantara dan pertahanan asal Amerika Serikat yang didirikan pada tahun 1916. Perusahaan ini merupakan salah satu produsen pesawat terbang komersial terbesar di dunia, serta menyediakan produk dan layanan militer dan antariksa. Seri pesawat terkenalnya meliputi Boeing 737, 747, 777, dan 787.
4.	C130	600	C130 atau C-130 Hercules adalah pesawat angkut militer serbaguna buatan Amerika Serikat yang dikembangkan oleh Lockheed Martin. Pesawat ini pertama kali terbang pada 1954 dan mulai dioperasikan pada 1956. C-130 dikenal karena kemampuannya lepas landas dan mendarat di landasan pendek dan tidak beraspal, serta digunakan untuk berbagai misi seperti pengangkutan pasukan, logistik, evakuasi medis, dan bantuan kemanusiaan. Beberapa variannya meliputi C-130B, C-130H, dan C-130J Super Hercules.
5.	F16	510	F16 atau F-16 Fighting Falcon adalah pesawat tempur multirole buatan Amerika Serikat yang dikembangkan oleh General Dynamics dan mulai diperkenalkan pada tahun 1978. Pesawat ini terkenal karena kelincahan, kecepatan, dan biaya operasional yang relatif rendah. Beberapa variannya meliputi F-16A/B sebagai versi awal, serta F-16C/D yang memiliki peningkatan avionik dan kemampuan tempur lebih lanjut.
6.	Grob	527	Grob adalah perusahaan dirgantara asal Jerman yang didirikan pada tahun 1971. Perusahaan ini dikenal memproduksi pesawat ringan berbahan komposit, terutama untuk pelatihan dan pengawasan. Salah satu produknya yang terkenal adalah Grob G-120, pesawat latih yang digunakan oleh militer dan sekolah penerbangan di berbagai negara.
7.	KAI	509	KAI (Korean Aerospace Industries) adalah perusahaan dirgantara asal Korea Selatan yang didirikan pada tahun 1999 melalui penggabungan beberapa produsen pesawat lokal. KAI fokus pada pengembangan dan produksi pesawat militer, pelatihan, serta komponen dirgantara. Produk utamanya meliputi pesawat tempur ringan T-50 Golden Eagle dan pesawat latih KT-1 Woongbi.
8.	Sukhoi	584	Sukhoi adalah pesawat tempur buatan Rusia yang mulai dikembangkan sejak era Uni Soviet, dengan varian terkenalnya seperti Su-27 yang pertama kali terbang tahun 1977. Pesawat ini dikenal karena kemampuan manuver tinggi, jangkauan jauh, dan kecepatan supersonik. Beberapa variannya meliputi Su-30, Su-33, Su-34, dan Su-35, yang digunakan oleh berbagai negara termasuk Rusia, India, dan Indonesia.

GAMBAR 5
Contoh Citra Pesawat per Kelas



GAMBAR 6
Distribusi Jumlah *Dataset* per Kelas

Dataset ini bukanlah data yang dapat diakses secara publik. Karena sebagian *dataset* ini dikumpulkan dari hasil karya pribadi peneliti. Peneliti mengumpulkan karya tersebut sebagai dataset melalui pengumpulan hasil dari hobi fotografi pesawat yang telah dilakukan oleh peneliti secara konsisten sejak lama. Adapun sebagian dataset lainnya berasal dari hasil karya foto anggota Komunitas Fotografer Aviasi Indonesia (KFAI). Peneliti yang juga sebagai anggota komunitas telah mendapatkan izin resmi dari pihak komunitas untuk diperbolehkan menggunakannya sebagai dataset dalam penelitian ini.

Setelah *dataset* siap digunakan, langkah berikutnya adalah masuk dalam proses pengolahan data atau *data preprocessing*. Tahap ini bertujuan untuk memastikan *dataset* dalam kondisi siap digunakan dalam penelitian, dengan kualitas yang memadai. Proses ini bisa melibatkan berbagai langkah seperti *data organizing* dengan melakukan penamaan ulang *file* citra sesuai nama folder-nya dengan pemberian nomor file dan pengecekan manual data, dll. Dengan kualitas *dataset* yang baik, diharapkan sistem klasifikasi citra pesawat yang dibangun juga memiliki performa yang optimal.

3.3 Eksperimen

3.3.1 *Data Preprocessing*

Data preprocessing merupakan tahap pertama yang harus dilakukan sebelum melakukan tahapan utama dalam sistem yang dibangun. Seperti yang dijelaskan di sub-bab 3.2 *Dataset*, bahwa tahapan ini dilakukan dengan tujuan untuk dapat memenuhi kebutuhan penelitian dapat menghasilkan hasil performa yang baik, alangkah baiknya *dataset* yang dimiliki memiliki kualitas yang baik juga. Ada beberapa proses yang dilakukan dalam melakukan tahapan *data preprocessing* ini, diantaranya *data organizing*, *data normalization*, *resizing data*, *encoding label*, *split data*, dan *data augmentation*.

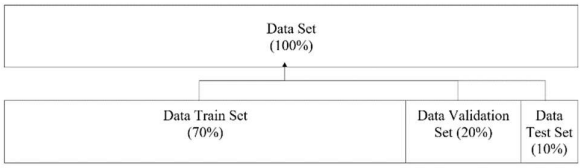
Data organizing adalah proses pengelolaan *dataset* mentah yang sebelumnya telah dikumpulkan oleh peneliti agar siap digunakan dalam proses pembangunan dan pelatihan model. Proses ini beberapa empat langkah. Pertama, pemilahan data citra sesuai jenis pabrikan pesawatnya dan memasukkannya ke dalam folder yang sesuai secara manual agar tidak ada file citra yang duplikat, serta menghapus manual file citra yang tidak cocok digunakan di penelitian ini. Kedua, semua file citra dilakukan penamaan ulang secara otomatis menggunakan *code program* Python agar konsisten dengan nama kelas berdasarkan folder kelas tempat citra tersebut disimpan, serta diberi penomoran berurutan untuk memudahkan identifikasi. Ketiga, semua file citra yang berukuran lebih dari 1 MB dikompresi menggunakan code program python untuk pemrosesan citra agar ukuran file menjadi maksimal 1 MB atau sekecil mungkin jika tidak memungkinkan mencapai ukuran tersebut, tanpa mengorbankan kualitas secara signifikan. Langkah ini dilakukan untuk memperkecil total ukuran dataset dari sekitar 24 GB menjadi sekitar 3–4 GB agar lebih efisien dalam penyimpanan dan pengolahan. Terakhir, mengubah semua ekstensi file menjadi satu ekstensi file yang sama yaitu *'.jpg'* secara otomatis menggunakan *code program* Python untuk mempermudah pembacaan data citra oleh sistem.

Data normalization adalah proses menormalisasi *pixel* citra awal [0-255] menjadi [0,1]. Tujuan dari dilakukannya *data normalization* ini diharapkan dapat membantu konvergensi model menjadi lebih cepat saat proses pelatihan model dan juga bermaksud menghindari

terjadinya *gradient explosion* yang menjadi kelemahan dari CNN.

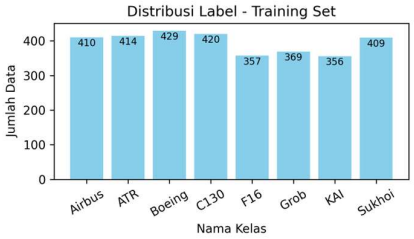
Resizing data merupakan proses pengukuran ulang ukuran dimensi data citra. *Resizing data* yang dilakukan di penelitian ini adalah menyeragamkan semua dimensi citra sama rata ke ukuran dimensi yang lebih kecil (224,224). Tujuan dilakukan *resizing data* ini adalah karena dalam prosesnya, CNN membutuhkan ukuran *input* data citra yang tetap dan sama rata, serta agar tidak membebankan ukuran memori dan komputasi dengan ukuran file citra yang sebelumnya sangat besar dan bisa memakan banyak RAM dan juga memperlambat kinerja pelatihan model.

Encoding label merupakan proses pengubahan nama kelas yang sebelumnya '*string*' menjadi bentuk '*int*'. Hal ini dilakukan untuk membantu sistem membaca label kelas yang harus dalam bentuk '*int*'. Sementara itu, ***split data*** adalah proses memecah *dataset* ke dalam beberapa bagian. *Split data* yang dilakukan di penelitian ini adalah membagi *dataset* menjadi 3 bagian, yaitu *data train set* sebesar 70%, *data validation set* sebesar 20%, dan *data test set* sebesar 10%. Ilustrasi pembagian *dataset* ini dapat dilihat di GAMBAR 7. *Data train set* digunakan untuk membangun model. Namun untuk skenario 2, *data train set* digunakan untuk melakukan *data preprocessing* selanjutnya, yaitu proses *data augmentation* terlebih dahulu untuk kemudian hasilnya digunakan untuk membangun model. Sementara itu, *data validation set* yang sepenuhnya data baru yang berbeda dengan *data train set* digunakan untuk memantau nilai *error* dan akurasi selama pelatihan model, serta untuk berusaha menghindari terjadinya *overfitting*. Dan untuk *data test set* yang sepenuhnya data baru yang benar-benar berbeda dari *data train* dan *validation set*, data ini digunakan untuk mengevaluasi seberapa baik akurasi sistem bekerja dalam melakukan klasifikasi terhadap citra pesawat baru yang belum dilihat selama pembangunan dan pelatihan model.

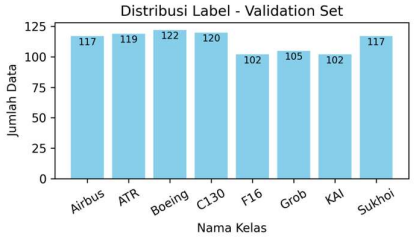


GAMBAR 7
Ilustrasi *Split Dataset*

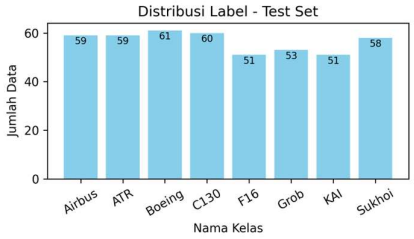
Setelah tahap *split data*, distribusi jumlah citra per label pada masing-masing subset juga ditampilkan untuk menunjukkan proporsi kelas dalam data. Distribusi jumlah citra per label pada *data train*, *validation*, dan *test set* dapat dilihat pada GAMBAR 8, GAMBAR 9, dan GAMBAR 10 secara berurutan. Visualisasi distribusi ini penting untuk memahami sebaran kelas dan mendukung analisis performa model di tiap *subset data*. Adapun seluruh *dataset* (*data train*, *validation*, dan *test set*) yang sudah terbagi ini, disimpan ke dalam sebuah *file* ekstensi '*.h5*'. Tujuannya adalah untuk menghemat waktu disaat memerlukan proses *running program* berulang agar tidak perlu lagi menjalankan proses *data preprocessing* yang sudah dilakukan sekali sebelumnya. Sehingga ketika akan menjalankan proses selanjutnya, yaitu *data augmentation* untuk skenario 2 ataupun proses tahapan utama pembangunan dan pelatihan model hanya perlu mengakses file *dataset* ber-ekstensi '*.h5*' tersebut tanpa harus mengulangi menjalankan tahap *data preprocessing* lagi dari awal.



GAMBAR 8
Distribusi Jumlah *Data Train* per Kelas



GAMBAR 9
Distribusi Jumlah *Data Validation* per Kelas



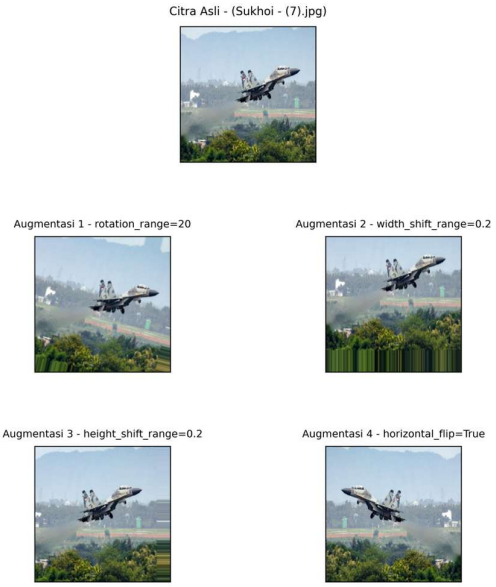
GAMBAR 10
Distribusi Jumlah *Data Test* per Kelas

Data augmentation adalah proses penambahan variasi citra agar sistem dapat belajar mengenali objek citra dari berbagai sudut dan kondisi yang berbeda. Proses ini merupakan proses yang sering ditambahkan dalam beraneka pengujian karena dianggap sebagai langkah untuk meminimalkan terjadinya *overfitting*. Ada berbagai teknik yang digunakan dalam proses ini, seperti *geometric transformation* (contoh: *flipping*, *rotation*, *cropping*, dan *translation*), *color space transformations*, *noise injection*, *random erasing*, dan *mixUp augmentation* [18]. Namun di penelitian ini, peneliti hanya melakukan teknik *rotation* (memutar posisi citra secara acak), teknik *flipping* (mencerminkan citra), dan teknik *shifting* (menggeser objek dalam citra). Adapun *data augmentation* ini dilakukan secara eksplisit dan manual sebelum digunakan untuk membangun dan melatih model di tahap tersebut. Hal ini dilakukan untuk menghemat waktu komputasi yang berbeda dengan saat melakukan *data augmentation* yang dilakukan secara *real-time*.

Citra hasil *data augmentation* beserta citra pelatihan aslinya disimpan secara permanen dalam sebuah file *dataset* berekstensi *'.h5'*. File ini tidak hanya menyimpan informasi data citra berformat *'.jpg'*, tetapi juga mencatat

nama file citra asli beserta label teknik augmentasi yang diterapkan (misalnya: *original, rotation*, *width_shift*, dan lain-lain). Dengan metode ini, jumlah total citra pelatihan bertambah menjadi 15.820 citra (3164 citra asli dan 12.656 citra hasil augmentasi). Dan proses pelatihan pada skenario kedua dapat langsung dilakukan dengan hanya mengakses file *dataset* hasil augmentasi berekstensi *'.h5'* tanpa perlu melakukan augmentasi ulang.

Gambaran visual hasil *data augmentation* ditampilkan pada GAMBAR 11, yang memperlihatkan satu citra pesawat asli dan empat citra versi augmentasinya berdasarkan teknik transformasi yang digunakan. Visualisasi ini menunjukkan bahwa meskipun berasal dari citra yang sama, tiap hasil augmentasi memiliki variasi bentuk atau posisi objek citra yang dapat memperkaya variasi data untuk proses pembangunan dan pelatihan model.



GAMBAR 11
Contoh Citra Hasil *Data Augmentation*

3.3.2 Pembangunan dan Pelatihan Model CNN-ResNet

Tahap utama penelitian ini adalah pembangunan dan pelatihan model CNN-ResNet. Untuk menjalankan skenario 1, proses pelatihan dilakukan dengan mengakses *data train* dan *validation set* yang yang telah melalui tahap *data preprocessing* tanpa *data augmentation*. Adapun data ini diakses dari file *'dataset.h5*' Sementara untuk menjalankan skenario 2, proses pelatihan menggunakan *data train set* yang telah diaugmentasi sebelumnya yang sudah tersimpan dalam file *'augmented_dataset_final.h5'* dan hanya mengakses file tersebut ketika diperlukan penggunaanya. Sedangkan *data validation set* tetap diakses dari file *'dataset.h5*' yang sama dengan skenario 1 untuk digunakan bersama di skenario 2 ini. Dengan demikian, proses pelatihan pada skenario 2 tidak lagi melakukan augmentasi secara *real-time*, melainkan langsung memanfaatkan data hasil augmentasi yang sudah disiapkan sebelumnya.

Model dibangun dengan mengimplementasikan penggunaan 3 variasi jumlah *layer* arsitektur ResNet yang berbeda, diantaranya ResNet-50 *layer*, ResNet-101 *layer*, dan ResNet-152 *layer*. Pengimplementasian ini dilakukan dengan tujuan untuk mengetahui pengaruh kedalaman jaringan terhadap performa sistem klasifikasi citra pesawat. Selain itu, masing-masing arsitektur juga diuji menggunakan dua konfigurasi parameter nilai *batch size*, yaitu 16 dan 32, dengan tujuan untuk menganalisis pengaruh parameter tersebut terhadap hasil klasifikasi.

Pendekatan *transfer learning* digunakan untuk mengembangkan model klasifikasi citra ini, dengan menggunakan bobot awal (*pre-trained weights*) dari *dataset* ImageNet. Dengan menggunakan '*include_top=False*', bagian atas arsitektur diganti dengan beberapa lapisan tambahan untuk memenuhi kebutuhan klasifikasi, yaitu '*GlobalAveragePooling2D*'. Ini digunakan untuk mengurangi jumlah parameter dibandingkan dengan *fully connected layer*, diikuti oleh *Dense layer* yang memiliki 512 *neuron* dan fungsi aktivasi *ReLU*, serta *Dropout* sebesar 0.3 untuk mengurangi risiko *overfitting*. Adapun *output Dense layer* dengan fungsi aktivasi *softmax* sebanyak jumlah kelas (yaitu 8 kelas) untuk menghasilkan prediksi *multi*-kelas. Selain itu, *Optimizer Adam* dengan *learning rate* sebesar 0.001 digunakan untuk mengkompilasi model karena sifatnya yang adaptif dan cepat mencapai konvergensi. Untuk fungsi *loss*, *categorical crossentropy* digunakan, karena dianggap sesuai untuk klasifikasi berbagai kelas. Dan *accuracy* adalah metrik evaluasi yang digunakan selama pelatihan.

Adapun setiap model dilatih hingga maksimal 30 *epoch*. Namun pelatihan akan berhenti lebih awal jika tidak ada peningkatan pada *validation loss* selama 5 *epoch* berturut-turut. Untuk menghentikan pelatihan lebih awal, *EarlyStopping* digunakan dengan parameter *restore_best_weights=True*. Penggunaan *EarlyStopping* ini bertujuan untuk mencegah terjadinya *overfitting*, yaitu ketika model terlalu menyesuaikan diri dengan data pelatihan hingga kehilangan kemampuan generalisasi terhadap data baru. Selain itu, teknik ini juga membantu meningkatkan efisiensi waktu pelatihan, karena proses berhenti secara otomatis saat model tidak menunjukkan perbaikan yang signifikan. Dengan pendekatan ini, sistem secara otomatis memilih *best epoch* untuk setiap model, yaitu *epoch* dengan performa validasi terbaik sebelum terjadi penurunan kinerja.

Setelah pelatihan selesai, setiap model disimpan dalam beberapa format, yaitu:

- File model ber-ekstensi *'.h5'* (format standar model Keras),
- File riwayat pelatihan (*training history*) berformat *'.json'* dan *'.pkl'*,
- Grafik visualisasi pelatihan berupa grafik *accuracy* dan *loss* per *epoch* yang menampilkan posisi *best epoch*, disimpan dalam format *'.png'*.

Penyimpanan ini dilakukan dengan tujuan untuk mempermudah melakukan evaluasi performa model dan menganalisanya serta untuk menghindari pelatihan ulang

jika ingin melakukan pengujian lanjutan. Visualisasi hasil pelatihan model berupa grafik *accuracy* dan *loss* per *epoch* dari masing-masing model yang dihasilkan secara otomatis ini digunakan untuk dianalisis lebih lanjut pada bab selanjutnya untuk menganalisa hasil evaluasi performa model pada setiap skenario pengujian. Adapun, proses pelatihan juga didesain menggunakan *checkpoint* di setiap *epoch* model. Desain ini digunakan untuk meminimalisir terjadinya pelatihan ulang ketika ada masalah dengan perangkat yang digunakan untuk menjalankan program. Sehingga ketika masalah terjadi, data pelatihan yang sudah dijalankan masih bisa diakses dan dilanjutkan hingga pelatihan model sepenuhnya selesai. Dan untuk efisiensi penyimpanan, setelah model sepenuhnya mencapai hasil terbaik dan tersimpan, semua file *checkpoint* yang tidak diperlukan akan dihapus secara otomatis.

3.3.3 Evaluasi Performansi Model

Semua model yang telah dilatih dievaluasi menggunakan *data test set*. Baik model maupun *data test set* diakses terlebih dahulu dari file ber-ekstensi *'.h5'* yang telah disimpan sebelumnya. File *data test set* tersebut berisi data citra baru yang belum pernah dilihat sebelumnya, baik di *data train* dan *validation set*. *Data test set* juga merupakan data tersimpan dari hasil *data preprocessing* yang telah dibersihkan namun belum melalui proses *data augmentation*, sehingga dapat mencerminkan performa murni dari model terhadap *data test set* yang tidak dimanipulasi. Evaluasi dilakukan dengan tujuan untuk mengetahui seberapa baik performa sistem yang dibangun. Setiap file model yang telah disimpan dalam file berekstensi *'.h5'* dimuat kembali satu per satu. Nama file model mencerminkan skenario pelatihan, konfigurasi arsitektur dan parameter nilai *batch size* yang digunakan saat pelatihan. Setelah model dimuat, proses evaluasi dilakukan dengan menguji model terhadap *data test set*, lalu membandingkan hasil prediksi label dengan label asli dari data tersebut.

Ketepatan klasifikasi dievaluasi dengan menghitung beberapa nilai terlebih dahulu, diantaranya menghitung jumlah contoh kelas yang dikenali dengan benar (*True Positive*), jumlah contoh yang dikenali dengan benar yang tidak termasuk dalam kelas (*True Negative*), dan contoh yang salah dimasukkan ke dalam kelas (*False Positive*) atau yang tidak dikenali sebagai contoh kelas (*False Negative*). Keempat hitungan ini biasa disebut sebagai *confusion matrix*. *Confusion matrix* adalah representasi *array* yang digunakan untuk melakukan klasifikasi biner, dan contoh *confusion matrix* dapat dilihat di TABEL 2 [19].

TABEL 2
Confusion Matrix

Data Class	***Classified as pos***	***Classified as neg***
pos	*True Positive (*TP*)*	*False Negative (*FN*)*
neg	*False Positive (*FP*)*	*True Negative (*TN*)*

Dengan bantuan nilai-nilai dari *confusion matrix*, model-model sistem klasifikasi yang telah dibangun dan dilatih bisa dianalisa dan dievaluasi menggunakan metrik seperti *precision*, *recall*, dan *f1-score*. *Precision* dihitung dengan membagi jumlah contoh positif yang diklasifikasikan dengan benar dengan jumlah contoh positif yang diberi label positif oleh sistem. *Recall* dihitung dengan membagi jumlah contoh positif yang diklasifikasikan dengan benar dengan jumlah contoh positif yang ada dalam data. Sedangkan *f1-score* adalah kombinasi hasil dari *precision* dan *recall* [19].

Seluruh hasil evaluasi dari setiap model, termasuk nilai *precision*, *recall*, dan *f1-score* dari model terkait, disusun dalam bentuk tabel dan disimpan sebagai file ber-ekstensi *'.csv'*. File ini kemudian menjadi dasar untuk analisa dan pemilihan model terbaik yang akan digunakan dalam tahap klasifikasi citra pada bagian selanjutnya.

3.3.4 Pemilihan Model CNN-ResNet Terbaik

Setelah seluruh model CNN-ResNet selesai dievaluasi, tahap berikutnya adalah melakukan pemilihan model terbaik. Proses ini dilakukan dengan membandingkan hasil evaluasi dari masing-masing model berdasarkan tiga metrik utama, yaitu *precision*, *recall*, dan *f1-score*. Di antara ketiga metrik tersebut, *f1-score* menjadi fokus utama dalam pemilihan model karena metrik ini merepresentasikan keseimbangan antara ketepatan (*precision*) dan kelengkapan (*recall*), yang menjadi hal penting dalam melakukan tugas klasifikasi citra multi-kelas yang menjadi tujuan utama penelitian ini.

Seluruh nilai *f1-score* dari hasil evaluasi model diurutkan dari yang tertinggi hingga terendah. Model yang memiliki *f1-score* tertinggi dianggap sebagai model dengan performa paling optimal dan terbaik, karena tidak hanya mampu mengenali kelas dengan benar, tetapi juga dianggap mampu meminimalkan kesalahan dalam mengklasifikasikan citra ke kelas yang salah.

Model terbaik yang terpilih dari tahap ini akan digunakan pada tahap akhir sistem, yaitu melakukan uji coba klasifikasi terhadap sejumlah citra dari semua kelas pesawat. Tahap klasifikasi ini bertujuan untuk menguji secara langsung bagaimana performa dari model terbaik saat dihadapkan pada berbagai kemungkinan prediksi, serta menganalisis kemampuan generalisasi model dalam mengenali citra pesawat dari berbagai kelas.

3.3.5 Klasifikasi Citra Jenis Pesawat Menggunakan Model CNN-ResNet terbaik

Tahap akhir dari sistem klasifikasi citra pesawat ini adalah menguji secara langsung model CNN-ResNet terbaik yang telah dipilih, dengan tujuan untuk mengetahui sejauh mana kemampuannya dalam mengklasifikasikan citra pesawat secara akurat. Pengujian ini tidak lagi bertujuan untuk pelatihan atau evaluasi performa secara kuantitatif, melainkan untuk mengamati dan menganalisis bagaimana hasil prediksi model terhadap sejumlah citra yang dipilih dari setiap kelas.

Dalam tahap ini, digunakan model terbaik yang telah dipilih pada tahap sebelumnya berdasarkan nilai *f1-score* tertinggi. Model tersebut diakses kembali dari file yang

telah disimpan sebelumnya dalam format *'.h5'*. Kemudian, model digunakan untuk melakukan klasifikasi terhadap sejumlah citra dari *data test set* yang belum pernah dilihat model selama proses pelatihan.

Untuk setiap kelas dalam *dataset* (total sebanyak 8 kelas), diambil masing-masing tiga contoh citra, dengan kriteria sebagai berikut:

1. Satu citra kelas terkait diprediksi benar (*true positive*),
2. Satu citra kelas salah diprediksi benar sebagai kelas terkait (*false positive*),
3. Satu citra kelas terkait diprediksi salah sebagai kelas lain (*false negative*).

Dengan demikian, total terdapat 24 citra uji yang dipilih secara selektif dari seluruh kelas. Semua citra ini diklasifikasikan menggunakan model CNN-ResNet terbaik yang telah dipilih pada tahap sebelumnya. Hasil klasifikasi ini digunakan untuk melihat seberapa baik model terbaik mengenali citra pesawat dari tiap kelas. Selain itu, hasil ini juga membantu untuk mengetahui faktor kesalahan apa saja yang masih sering terjadi dalam prediksi.

4. Evaluasi

4.1 Skenario, Perangkat dan Lingkungan Pengujian

Sistem klasifikasi citra pesawat dibangun menggunakan algoritma CNN menggunakan arsitektur ResNet. Sistem yang telah dibangun dilakukan pengujian dengan tujuan mengetahui performa sistem yang dibangun telah berjalan dengan baik atau buruk.

Dua skenario pengujian digunakan dalam penelitian ini: skenario klasifikasi citra tanpa menggunakan *data augmentation* dan skenario klasifikasi citra dengan menggunakan *data augmentation*. Setiap skenario diuji pada tiga varian *layer* arsitektur ResNet, yaitu ResNet-50 *layer*, ResNet-101 *layer*, dan ResNet-152 *layer*, serta dua konfigurasi parameter nilai *batch size* 16, dan 32. Oleh karena itu, total ada 12 kombinasi model yang dibangun, dilatih, dan diuji dalam penelitian ini. Setelah itu, semua model dievaluasi dan performanya dibandingkan. Adapun pengujian sistem dilakukan dengan mengevaluasi performa dari semua model menggunakan 3 metrik evaluasi, yaitu *precision*, *recall*, dan *f1-score*. Dari hasil evaluasi performa sistem, dipilih model CNN-ResNet terbaik dari 12 model yang ada dengan nilai *f1-score* yang dimiliki merupakan nilai yang tertinggi.

Penulisan *code program* untuk penelitian ini menggunakan bahasa pemrograman Python, dengan memanfaatkan pustaka TensorFlow dan Keras untuk mengimplementasikan CNN dan klasifikasi citra. Implementasi *code program* dibagi ke dalam 5 file terpisah, yang terdiri dari:

1. *Code Program Data Organization*
2. *Code Program* Utama *Data Preprocessing*
3. *Code Program* Pembangunan dan Pelatihan Model Tanpa Augmentasi
4. *Code Program* Pembangunan dan Pelatihan Model Dengan Augmentasi
5. *Code Program* Evaluasi Model dan Klasifikasi Citra serta Analisis Hasil

Sebagian *code program* dalam penelitian ini pada awalnya dijalankan secara lokal menggunakan Jupyter Notebook pada laptop pribadi dengan spesifikasi menengah. Laptop ini digunakan untuk menjalankan beberapa *code program*, dengan tujuan untuk menguji sejauh mana spesifikasi perangkat lokal mampu menangani beban kerja sistem. Namun, untuk proses yang memerlukan komputasi intensif, khususnya pada tahap pembangunan dan pelatihan model CNN-ResNet yang melibatkan banyak parameter dan *layer*, peneliti memanfaatkan layanan Google Colab Pro yang memiliki spesifikasi jauh lebih tinggi dan canggih. Penggunaan Colab Pro terbukti memberikan waktu pelatihan yang jauh lebih efisien, menghindari resiko *crash kernel*, serta mampu menghasilkan model dengan performa lebih optimal dibandingkan saat dijalankan di perangkat lokal. Untuk detailnya, TABEL 3 merangkum perbandingan spesifikasi perangkat dan alokasi penggunaan masing-masing dalam eksperimen:

TABEL 3
Perbandingan Spesifikasi Perangkat dan Lingkungan Eksperimen

Komponen	**Laptop Pribadi**	**Google Colab Pro**
CPU	Intel Core i5	2 vCPU (Intel Xeon-*based*)
GPU	Intel(R) Iris(R) Xe Graphics (8 GB)	NVIDIA A100 (VRAM 40 GB)
RAM	16 GB	83 GB (*High*-RAM *instance*)
Platform	Jupyter Notebook lokal Ver. 6.5.7	Google Colab Pro (*cloud-based*)
Versi Python	3.7.6	3.11.13
***Code* yang Dijalankan**	1. Program *Data Organization* 2. Program Utama *Data Preprocessing* 3. Program Pembangunan dan Pelatihan Model Tanpa Augmentasi 4. Program Pembangunan dan Pelatihan Model Dengan Augmentasi	1. Program Pembangunan dan Pelatihan Model Tanpa Augmentasi 2. Program Pembangunan dan Pelatihan Model Dengan Augmentasi 3. Program Evaluasi Model dan Klasifikasi Citra serta Analisis Hasil
Tujuan Penggunaan	Menguji kemampuan perangkat lokal dalam menjalankan *code program*	Mempercepat waktu komputasi, menghindari *crash*, dan memperoleh hasil model yang optimal dan efisien.

4.2 Hasil dan Analisis Pelatihan Model CNN-ResNet

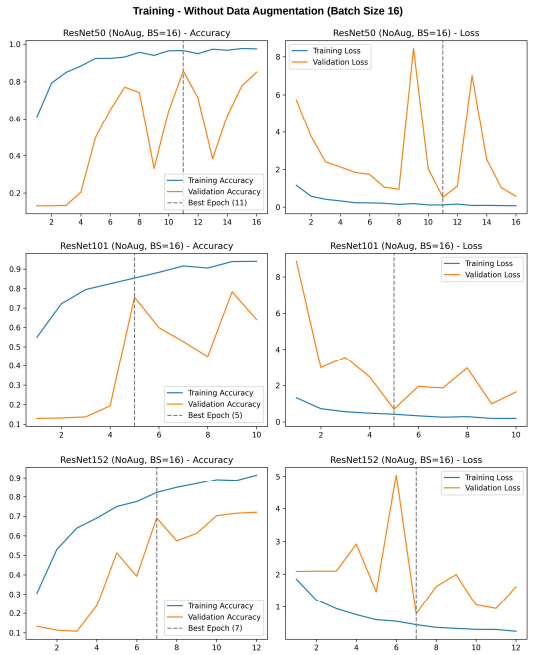
Penelitian ini berhasil membangun dan melatih total 12 model CNN-ResNet. Setiap model dibangun dan dilatih dengan kombinasi arsitektur ResNet (50, 101, dan 152 *layer*), skenario data (dengan dan tanpa *data augmentation*), serta konfigurasi parameter nilai *batch size* (16 dan 32). Pelatihan ini juga dilakukan hingga maksimum 30 *epoch*, dengan penerapan *EarlyStopping* untuk menghentikan pelatihan lebih awal apabila tidak terjadi peningkatan pada nilai *validation loss* selama 5 *epoch* berturut-turut. Selain meningkatkan efisiensi waktu pelatihan, pendekatan ini dilakuakan dengan tujuan untuk mencegah terjadinya *overfitting*, serta memastikan bahwa bobot model terbaik berasal dari *epoch* dengan performa validasi tertinggi.

Selama proses pelatihan, seluruh nilai *accuracy* dan *loss* pada *data train* dan *validation set* dicatat serta divisualisasikan ke dalam grafik yang disajikan dalam GAMBAR 12, GAMBAR 13, GAMBAR 14, dan GAMBAR 15. Keempat gambar tersebut menyajikan visualisasi *accuracy* dan *loss* dari ketiga model ResNet (ResNet-50 *layer*, ResNet-101 *layer*, dan ResNet-152 *layer*) untuk masing-masing kombinasi skenario data pelatihan dan konfigurasi parameter nilai *batch size*. Visualisasi tersebut digunakan untuk mengevaluasi pola pelatihan model serta mengidentifikasi indikasi *underfitting*, *overfitting*, atau kondisi pelatihan yang *fit*.

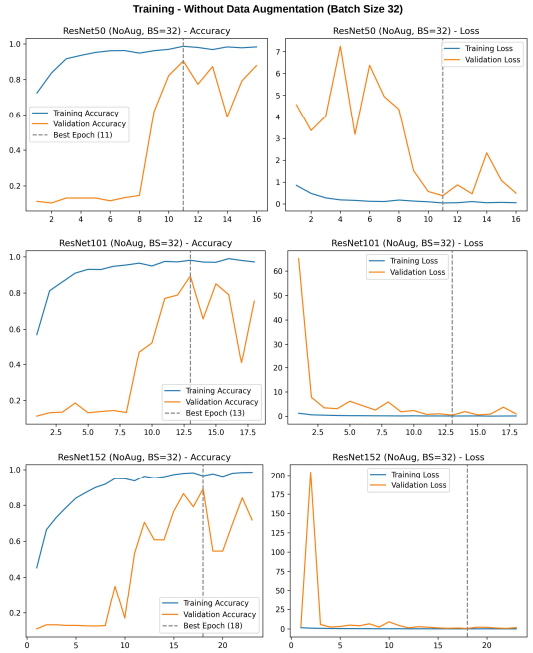
Adapun kriteria kondisi pelatihan model ditentukan sebagai berikut:

1. ***Overfitting***, digambarkan sebagai situasi suatu model memberikan hasil yang baik pada *data train set*, akan tetapi berbeda hasilnya pada data baru seperti *data validation* dan *test set* diberikan hasil yang buruk [1].
2. ***Underfitting***, digambarkan ketika model dalam pelatihannya tidak menghasilkan *accuracy* cukup baik pada data baru seperti *data validation* dan *test set* karena tidak cukup banyak belajar atau mendapatkan cukup informasi selama pelatihan [1].
3. ***Fit***, digambarkan sebagai kondisi ketika tidak ada indikasi *overfitting* ataupun *underfitting*, yang artinya model belajar cukup baik dan sistem bekerja dengan efisien.

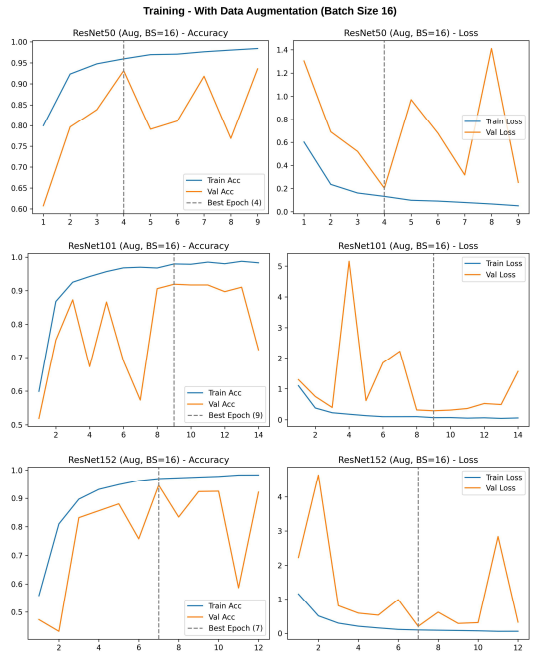
Peneliti menyajikan ringkasan kriteria kondisi pelatihan dari 12 model yang dapat dilihat pada TABEL 4. Kriteria dan analisis terhadap grafik pelatihan yang dihasilkan, ditulis oleh peneliti berdasarkan pembahasan yang sudah dijelaskan sebelumnya.



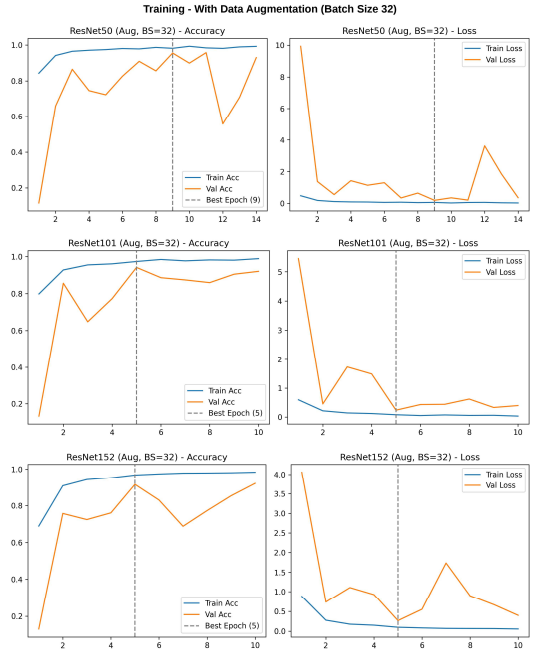
GAMBAR 12
Grafik Untuk Model Dengan Skenario Tanpa Data Augmentation, Batch Size 16



GAMBAR 13
Grafik Untuk Model Dengan Skenario Tanpa *Data Augmentation*, *Batch Size* 32



GAMBAR 14
Grafik Untuk Model Dengan Skenario Data Augmentation, Batch Size 16



GAMBAR 15
Grafik Untuk Model Dengan Skenario *Data Augmentation*, *Batch Size* 32

TABEL 4
Hasil Kondisi Saat Pelatihan Model

Model ke-	***Layer* ResNet**	***Data***	***batch_size***	**Jumlah *Epoch***	***Best Epoch***	**Keterangan**
1	50	*No Augmentation*	16	16	11	*Overfit ting*
2	50	*No Augmentation*	32	16	11	*Fit*
3	101	*No Augmentation*	16	10	5	*Underfitting*
4	101	*No Augmentation*	32	18	13	*Fit*
5	152	*No Augmentation*	16	12	7	*Underfitting*
6	152	*No Augmentation*	32	23	18	*Fit*
7	50	*Augmentation*	16	9	4	*Fit*
8	50	*Augmentation*	32	14	9	*Fit*
9	101	*Augmentation*	16	14	9	*Fit*
10	101	*Augmentation*	32	10	5	*Fit*
11	152	*Augmentation*	16	12	7	*Fit*
12	152	*Augmentation*	32	10	5	*Fit*

Berdasarkan hasil analisis yang dilakukan peneliti terhadap grafik pelatihan, diperoleh bahwa dua model mengalami *underfitting*, yaitu model ke-3 (ResNet-101 *layer*, tanpa *data augmentation*, *batch size* 16) dan model ke-5 (ResNet-152 *layer*, tanpa *data augmentation*, *batch size* 16). Kedua model ini menunjukkan *validation accuracy* yang rendah dan tidak terlalu meningkat di sekitar 0.7. Model ke-3 memiliki *training accuracy* yang belum maksimal dan *best epoch* yang terjadi sangat awal, menandakan bahwa proses pembelajaran terhenti sebelum model cukup memahami data. Sementara itu, meskipun model ke-5 memiliki *training accuracy* tinggi, tetap menunjukkan *validation loss* yang tidak stabil serta *validation accuracy* yang rendah, sehingga kemampuan generalisasinya belum tercapai.

Sebaliknya, model ke-1 (ResNet-50 *layer*, tanpa *data augmentation*, *batch size* 16) termasuk ke dalam kategori *overfitting*, karena memiliki *training accuracy* yang hampir mencapai 1.0, namun *validation accuracy* cenderung naik-turun yang tidak stabil dan *validation loss* tetap tinggi. Hal ini menunjukkan bahwa model terlalu belajar pada *data train set* dan tidak mampu mengeneralisasi dengan baik pada data baru seperti *data validation set*.

Sementara itu, sembilan model lainnya termasuk dalam kategori *fit*, ditandai dengan *validation accuracy* yang terus meningkat, *training accuracy* yang tinggi, serta *loss* yang menurun stabil tanpa adanya banyak kondisi kurva yang naik-turun yang tajam. Semua model yang dilatih menggunakan *data augmentation* (model ke-7 hingga ke-

12) tergolong *fit*, menunjukkan bahwa skenario menggunakan *data augmentation* berhasil meningkatkan keragaman data dan mencegah *overfitting*. Selain itu, model tanpa menggunakan *data augmentation* dengan *batch size* bernilai 32 yang lebih besar daripada 16 seperti model ke-2 (ResNet-50 *layer*, *batch size* 32), ke-4 (ResNet-101 *layer*, *batch size* 32), dan ke-6 (ResNet-152 *layer*, *batch size* 32) juga tergolong *fit*. Hal ini menunjukkan bahwa peningkatan nilai *batch size* turut berkontribusi terhadap kestabilan pelatihan pada skenario tanpa *data augmentation*.

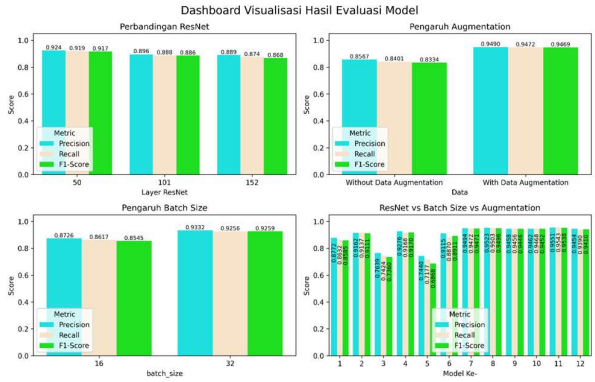
4.3 Hasil dan Analisis Evaluasi Performa Model CNN-ResNet

Tahapan evaluasi performa dilakukan terhadap 12 model CNN-ResNet yang telah dibangun dan dilatih dengan kombinasi dari tiga arsitektur ResNet (ResNet-50 *layer*, ResNet-101 *layer*, dan ResNet-152 *layer*), konfigurasi parameter nilai *batch size* (16 dan 32), serta dua skenario pelatihan (dengan dan tanpa *data augmentation*). Seluruh model sebelumnya telah disimpan dalam format file ber-ekstensi *'.h5'* dan kemudian diakses kembali untuk dilakukan evaluasi performa terhadap *data test set*, yang diakses dari file *'dataset.h5'* dan telah melalui proses *data preprocessing* tanpa melakukan *data augmentation*. Evaluasi dilakukan dengan menghitung *precision*, *recall*, dan *f1-score* dari hasil prediksi setiap model terhadap *data test set*. Seluruh hasil evaluasi ini disusun dalam bentuk tabel dan disimpan dalam file ber-ekstensi *'.csv'*, yang kemudian menjadi dasar untuk analisis performa dan pemilihan model terbaik pada tahap selanjutnya.

Rincian lengkap hasil evaluasi seluruh model, dapat dilihat pada TABEL 5. Selain itu, peneliti juga menyajikan hasil evaluasi dalam bentuk gambar visualisasi agar hasil pengujian ini dapat dilihat dan dianalisis dengan mudah. GAMBAR 16 menampilkan 4 visualisasi hasil evaluasi, diantaranya visualisasi perbandingan antar kombinasi arsitektur ResNet, visualisasi pengaruh penggunaan *data augmentation*, visualisasi pengaruh konfigurasi *batch size*, dan visualisasi antar semua model.

TABEL 5
Hasil Evaluasi Performansi Model

Model ke-	***Layer* ResNet**	***Data***	***batch_size***	***Precision***	***Recall***	***F1-Score***
1	50	*No Augmentation*	16	0.877224	0.863218	0.858484
2	50	*No Augmentation*	32	0.916190	0.913672	0.911137
3	101	*No Augmentation*	16	0.763891	0.742406	0.736036
4	101	*No Augmentation*	32	0.927585	0.916836	0.917005
5	152	*No Augmentation*	16	0.744003	0.717711	0.686827
6	152	*No Augmentation*	32	0.911507	0.886982	0.891129
7	50	*Augmentation*	16	0.949377	0.947165	0.947139
8	50	*Augmentation*	32	0.952331	0.950322	0.949630
9	101	*Augmentation*	16	0.945815	0.945564	0.944631
10	101	*Augmentation*	32	0.946192	0.946836	0.945179
11	152	*Augmentation*	16	0.955092	0.954284	0.953754
12	152	*Augmentation*	32	0.945446	0.939007	0.941049



GAMBAR 16
Visualisasi Analisis Hasil Evaluasi Performansi

Dari GAMBAR 16 nampak visualisasi pertama menunjukkan perbandingan performa rata-rata dari masing-masing arsitektur ResNet berdasarkan *precision*, *recall*, dan *f1-score*. Dari grafik ini terlihat bahwa arsitektur ResNet-50 *layer* memberikan performa rata-rata terbaik dibandingkan ResNet-101 *layer* dan ResNet-152 *layer*, meskipun memiliki jumlah *layer* yang lebih sedikit. Hal ini mengindikasikan bahwa kedalaman jaringan yang lebih tinggi tidak selalu menjamin performa yang lebih baik dalam tugas klasifikasi citra pesawat pada penelitian ini dan hal ini mungkin bisa terjadi disebabkan adanya pengaruh konfigurasi *layer* arsitektur model CNN-ResNet atau parameter lainnya.

Sementara itu, visualisasi kedua menunjukkan pengaruh penggunaan *data augmentation* terhadap performa model. Grafik ini memperlihatkan bahwa model yang dilatih dengan menggunakan *data augmentation* secara umum menunjukkan peningkatan nilai evaluasi dibandingkan model dengan tanpa *data augmentation*, yang mengindikasikan bahwa pendekatan skenario *data augmentation* berhasil meningkatkan kemampuan generalisasi model terhadap data baru. Selanjutnya, visualisasi ketiga memperlihatkan pengaruh konfigurasi parameter nilai *batch size* terhadap performa. Rata-rata *f1-score* model dengan *batch size* sebesar 32 lebih tinggi dibandingkan dengan *batch size* sebesar 16, yang menunjukkan bahwa penggunaan *batch size* yang lebih besar dapat memberikan kestabilan pelatihan dan hasil

yang lebih baik.
Terakhir, visualisasi keempat menampilkan perbandingan *precision*, *recall*, dan *f1-score* dari seluruh model yang diuji. Terlihat bahwa sebagian besar model menunjukkan keseimbangan antara *precision* dan *recall*, yang menandakan kemampuan generalisasi model terhadap data baru cukup baik. Meski begitu, seperti yang sudah dibahas dalam sub-bab sebelumnya, selama proses pelatihan ditemukan beberapa model ada yang menunjukkan gejala *overfitting* atau *underfitting*. Hal ini terlihat dari selisih antara jumlah epoch dan *best epoch*, serta pola grafik *accuracy* dan *loss* yang kurang stabil. Artinya, performa model tidak cukup dinilai hanya dari hasil akhir evaluasi, tapi juga perlu dilihat dari proses pelatihannya.

4.4 Hasil dan Analisis Pemilihan Model CNN-ResNet Terbaik

Model terbaik dipilih berdasarkan nilai *f1-score* tertinggi dari seluruh model yang telah dievaluasi. Nilai *f1-score* dari semua model diurutkan dari yang tertinggi hingga terendah. Dari pengurutan nilai *f1-score* ini didapatkan tiga model dengan performa tertinggi dengan rincian informasinya disajikan dalam TABEL 6. Terlihat dari ketiganya, model ke-11, yaitu model dengan arsitektur ResNet-152 *layer*, skenario pelatihan dengan *data augmentation*, dan *batch size* sebesar 16, menunjukkan *f1-score* paling tinggi sebesar 0.953754. Oleh karena itu, model ini dipilih sebagai model terbaik yang akan digunakan untuk melakukan klasifikasi citra jenis pesawat.

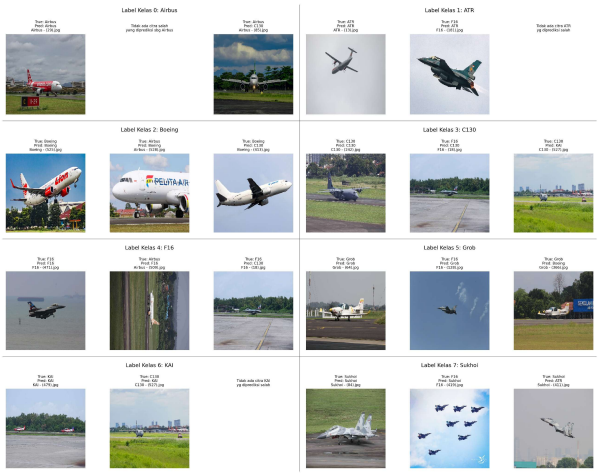
TABEL 6
3 Model CNN-ResNet Terbaik

Top	**Model ke-**	***Layer* ResNet**	***Data***	***batch_size***	***Precision***	***Recall***	***F1-Score***
1	11	152	*Augmentation*	16	0.955092	0.954284	0.953754
2	8	50	*Augmentation*	32	0.952331	0.950322	0.949630
3	7	50	*Augmentation*	16	0.949377	0.947165	0.947139

Dilihat dari TABEL 6 yang disajikan, peneliti memberikan analisis bahwa ternyata model terbaik tidak selalu berasal dari arsitektur terdalam atau *batch size* terbesar. Ketiga model terbaik ini memang menggunakan skenario *data augmentation* dan membuktikan bahwa hal ini membuktikan pengaruh skenario ini pada performa model. Namun dua model di antaranya justru menggunakan ResNet-50 *layer*, dibandingkan ResNet-101 *layer* atau ResNet-152 *layer* yang keduanya merupakan arsitektur jaringan yang lebih dalam. Selain itu, model dengan performa tertinggi justru menggunakan *batch size* sebesar 16, bukan 32. Hal ini menunjukkan bahwa performa terbaik bukan tentang semakin bertambah dalamnya suatu jaringan, tapi tentang kombinasi konfigurasi arsitektur yang pas.

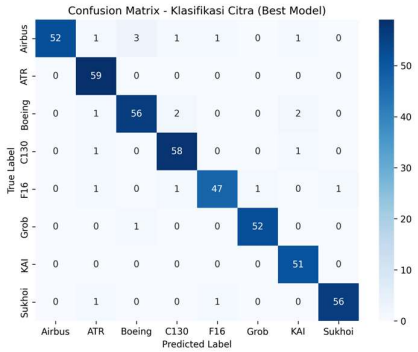
4.5 Hasil dan Analisis Klasifikasi Citra Menggunakan Model CNN-ResNet Terbaik

Model terbaik yang telah dipilih sebelumnya digunakan untuk menguji kemampuannya dalam mengklasifikasikan citra pesawat dari *data test set*, sebagai bentuk implementasi dari tujuan utama penelitian ini, yaitu membangun sistem klasifikasi citra jenis pesawat. Untuk dapat melihat hasil klasifikasi secara visual, dipilih total 24 citra secara selektif, masing-masing 3 citra dari setiap kelas pesawat, untuk mewakili tiga kondisi klasifikasi: *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN). TP merupakan kondisi klasifikasi citra dari kelas terkait diprediksi benar sebagai kelas tersebut. Sementara itu, FP merupakan kondisi klasifikasi citra dari kelas yang salah diprediksi benar sebagai kelas terkait. Dan FN merupakan kebalikan dari FP, yaitu kondisi klasifikasi citra dari kelas terkait diprediksi salah. Visualisasi hasil klasifikasi terhadap citra-citra tersebut disajikan pada GAMBAR 17.



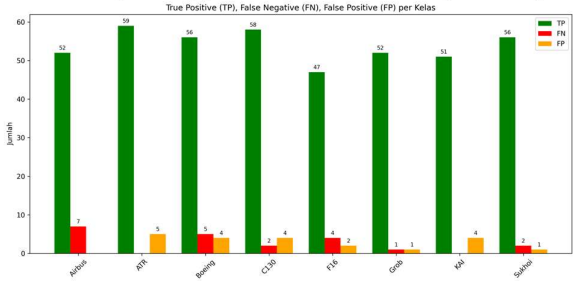
GAMBAR 17
Hasil Klasifikasi Citra Jenis Pesawat Dengan Model Terbaik

Untuk memudahkan menganalisis hasil klasifikasi citra ini, peneliti menyajikan hasilnya dalam 2 bentuk visualisasi. Pertama, bentuk visualisasi berupa *heatmap confusion matrix* yang dapat dilihat pada GAMBAR 18. Dari visualisasi *heatmap confusion matrix* ini, terlihat bahwa sebagian besar citra berhasil diklasifikasikan dengan benar, terutama pada kelas ATR, KAI, dan Grob, yang memiliki nilai TP tinggi dan minimnya kesalahan prediksi. Di sisi lain, beberapa kelas seperti Airbus dan Boeing menunjukkan adanya kesalahan klasifikasi ke beberapa kelas lain seperti C130 dan Sukhoi.



GAMBAR 18
Visualisasi *Heatmap Confusion Matrix* Untuk Hasil Klasifikasi Citra Pesawat Dengan Model Terbaik

Kedua, bentuk visualisasi berupa 3 grafik batang jumlah TP, FP, dan FN per kelas yang dapat dilihat pada GAMBAR 19. Dari grafik tersebut terlihat bahwa kelas ATR memiliki jumlah TP tertinggi (59 citra terklasifikasi benar), menunjukkan performa klasifikasi kelas tersebut berjalan sangat baik. Sementara itu, kelas Airbus dan Boeing menunjukkan jumlah FN yang cukup tinggi. Hal ini dapat dijelaskan karena kedua kelas ini memiliki kemiripan visual yang tinggi terutama pada bentuk badan pesawatnya, sehingga menyebabkan model kesulitan membedakan keduanya. Adapun FP tertinggi tercatat pada kelas ATR, C130, dan KAI yang ketiga kelas ini memang memiliki ciri visual serupa, yaitu sama-sama memiliki visual baling-baling pada bagian dari badan pesawatnya.



GAMBAR 19
Visualisasi Diagram Batang *Confusion Matrix* Untuk Hasil Klasifikasi Citra Pesawat Dengan Model Terbaik

Hasil klasifikasi menunjukkan bahwa model terbaik ini dianggap mampu mengenali sebagian besar kelas dengan cukup baik, terutama pada kelas-kelas yang memiliki karakteristik visual yang konsisten. Meskipun begitu, kesalahan klasifikasi masih terjadi pada kelas-kelas yang memiliki kemiripan visual, seperti Airbus dan Boeing atau ATR dan C130. Adanya temuan ini mengindikasikan bahwa variasi visual dan kemiripan bentuk antar pesawat dapat berdampak besar terhadap akurasi klasifikasi. Oleh karena itu, penerapan teknik *data augmentation* yang lebih variatif serta pelatihan yang lebih panjang sangat diperlukan untuk meningkatkan kemampuan generalisasi model dalam mengenali perbedaan antar kelas yang mirip.

5. Kesimpulan

Penelitian ini berhasil membangun dan mengevaluasi 12 model CNN-ResNet dengan kombinasi arsitektur ResNet-50 *layer*, ResNet-101 *layer*, dan ResNet-152 *layer*, dua skenario pelatihan (dengan dan tanpa *data augmentation*), serta dua konfigurasi *batch size* (16 dan 32). Hasil evaluasi menunjukkan bahwa penggunaan *data augmentation* dan pemilihan arsitektur yang sesuai sangat berpengaruh terhadap performa klasifikasi citra. Model terbaik diperoleh dari model ke-11 dengan arsitektur ResNet-152 *layer*, skenario pelatihan menggunakan *data augmentation*, dan *batch size* sebesar 16, yang menghasilkan *f1-score* tertinggi yaitu 0.953754. Menariknya, dua model terbaik lainnya justru berasal dari arsitektur ResNet-50 *layer*, yang membuktikan bahwa arsitektur yang lebih dalam tidak selalu memberikan hasil terbaik dan bergantung pada kombinasi arsitektur dengan konfigurasi parameter yang sesuai. Dan untuk membuktikannya harus melalui lebih banyak uji coba. Meskipun dalam proses pelatihan sebelumnya beberapa model sempat menunjukkan indikasi *overfitting* atau *underfitting* berdasarkan grafik *accuracy* dan *loss*-nya, sebagian besar model sebenarnya telah mencapai kondisi *fit*. Hal ini diperkuat oleh hasil evaluasi performa yang menunjukkan bahwa tidak ada model yang mengalami *overfitting* atau *underfitting* secara signifikan, dengan *f1-score* terendah berada pada kisaran 0.6–0.7 dan tertinggi di atas 0.9, serta nilai *precision* dan *recall* yang seimbang di semua model. Dengan adanya penemuan ini membuktikan bahwa penilaian performa model tidak bisa hanya dilihat dari grafik pelatihannya saja, melainkan perlu dilihat secara menyeluruh dan tak hanya dari hasil evaluasi akhir saja. Dan dengan demikian dapat disimpulkan bahwa model yang telah dibangun ini sudah menunjukkan performa yang baik. Dibuktikan juga melalui visualisasi hasil klasifikasi telah menunjukkan bahwa model memang mampu mengenali sebagian besar kelas dengan baik, namun masih mengalami beberapa kesalahan pada kelas dengan kemiripan visual tinggi seperti Airbus dan Boeing atau ATR dan C130. Temuan ini mengindikasikan bahwa faktor visual objek pesawat juga sangat mempengaruhi akurasi klasifikasi. Berdasarkan banyaknya temuan dalam penelitian ini, saran untuk penelitian selanjutnya adalah memperluas variasi teknik *data augmentation*, dan memperbanyak jumlah *data train set*. Selain itu, perlu juga mempertimbangkan pelatihan dengan jumlah *epoch* yang dicocokkan agar tidak terjadi kondisi *underfitting* atau *overfitting* selama pelatihan. Dan paling penting, untuk eksperimen lanjutan dapat dilakukan banyak uji coba kombinasi arsitektur yang pas dengan mengeksplorasi konfigurasi parameter lainnya seperti *learning rate*, jenis *optimizer*, atau integrasi teknik *transfer learning* untuk memperkuat kemampuan generalisasi model yang baik, serta didukung perangkat keras yang mumpuni agar proses pelatihan dapat berjalan optimal.

REFERENSI

[1] M. M. Taye, “Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions,” *Computation*, vol. 11, no. 3, p. 52, 2023.

[2] R. Radikto, D. I. Mulyana, M. A. Rofik, and M. O. Z. Zakaria, “Klasifikasi Kendaraan pada

Jalan Raya menggunakan Algoritma Convolutional Neural Network (CNN)," *Jurnal Pendidikan Tambusai*, vol. 6, no. 1, pp. 1668–1679, 2022.

[3] S. Cong and Y. Zhou, "A review of convolutional neural network architectures and their optimizations," *Artif Intell Rev*, vol. 56, no. 3, pp. 1905–1969, 2023.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] F. Chen, R. Ren, W. Xu, and T. de Voorde, "Airplane recognition from remote sensing images with deep convolutional neural network," in *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 264–267.

[6] M. A. Butt *et al.*, "Convolutional neural network based vehicle classification in adverse illuminous conditions for intelligent transportation systems," *Complexity*, vol. 2021, no. 1, p. 6644861, 2021.

[7] T. A. Sugiarto, M. A. Soeleman, and P. Pujiono, "Enhancing Augmentation-Based Resnet50 for Car Brand Classification," *Journal of Applied Intelligent System*, vol. 8, no. 3, pp. 400–412, 2023.

[8] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of image classification algorithms based on convolutional neural networks," *Remote Sens (Basel)*, vol. 13, no. 22, p. 4712, 2021.

[9] L. Xie, R. Hong, B. Zhang, and Q. Tian, "Image classification and retrieval are one," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, 2015, pp. 3–10.

[10] R. I. Borman, I. Ahmad, and Y. Rahmanto, "Klasifikasi Citra Tanaman Perdu Liar Berkhasiat Obat Menggunakan Jaringan Syaraf Tiruan Radial Basis Function," *Bulletin of Informatics and Data Science*, vol. 1, no. 1, pp. 6–13, 2022.

[11] H. Mayatopani, R. I. Borman, W. T. Atmojo, and A. Arisantoso, "CLASSIFICATION OF VEHICLE TYPES USING BACKPROPAGATION NEURAL NETWORKS WITH METRIC AND ECCENTRICITY PARAMETERS," *Jurnal Riset Informatika*, vol. 4, no. 1, pp. 65–70, 2021.

[12] S. S. Nath, G. Mishra, J. Kar, S. Chakraborty, and N. Dey, "A survey of image classification methods and techniques," in *2014 International conference on control, instrumentation, communication and computational technologies (ICCICCT)*, 2014, pp. 554–557.

[13] A. T. Vo, H. S. Tran, and T. H. Le, "Advertisement image classification using convolutional neural network," in *2017 9th International conference on knowledge and systems engineering (KSE)*, 2017, pp. 197–202.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Adv Neural Inf Process Syst*, vol. 25, 2012.

[15] L. Yan, S. Wan, P. Jin, and C. Zou, "Airplane fine-grained classification in remote sensing images via transferred CNN-based models," in *International Conference on Geo-Spatial Knowledge and Intelligence*, 2017, pp. 318–326.

[16] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," *arXiv preprint arXiv:1306.5151*, 2013.

[17] Y. Xu, Z. Cui, W. Guo, Z. Zhang, and W. Yu, "Self-supervised auto-encoding multi-transformations for airplane classification," in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, 2021, pp. 2365–2368.

[18] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 91–99, 2022.

[19] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf Process Manag*, vol. 45, no. 4, pp. 427–437, 2009.