# KLASIFIKASI SPESIES ULAR MENGGUNAKAN DEEP LEARNING DENGAN ARSITEKTUR EFFICIENTNETV2S

**TUGAS AKHIR**


Disusun oleh :

**Rizqi Ananta Ekta Putra**

**123210023**



**PROGRAM STUDI INFORMATIKA**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK INDUSTRI**

**UNIVERITAS PEMBANGUNAN NASIONAL "VETERAN"**

**YOGYAKARTA**


**2025**

# KLASIFIKASI SPESIES ULAR MENGGUNAKAN DEEP LEARNING DENGAN ARSITEKTUR EFFICIENTNETV2S

**TUGAS AKHIR**

Sebagai syarat untuk memperoleh gelar sarjana S-1
Program Studi Informatika, Jurusan Informatika, Fakultas Teknik Industri,
Universitas Pembangunan Nasional "Veteran" Yogyakarta


Disusun oleh :

**Rizqi Ananta Ekta Putra**

**123210023**


**PROGRAM STUDI INFORMATIKA**
**JURUSAN TEKNIK INFORMATIKA**
**FAKULTAS TEKNIK INDUSTRI**
**UNIVERITAS PEMBANGUNAN NASIONAL "VETERAN"**
**YOGYAKARTA**
**2025**

# HALAMAN PENGESAHAN PEMBIMBING

## KLASIFIKASI SPESIES ULAR MENGGUNAKAN DEEP LEARNING DENGAN ARSITEKTUR EFFICIENTNETV2S

Disusun Oleh:
Rizqi Ananta Ekta Putra
123210023

Telah diuji dan dinyatakan lulus oleh pembimbing
pada tanggal: 22 Juli 2025

Menyetujui,
Pembimbing

Rifki Indra Perwira, S.Kom., M.Eng
NIDN. 0508078301

Menyetujui,
Koordinator Program Studi

Dessyanto Boedi P., S.T., M.T.
NIDN. 0505127501

# HALAMAN PENGESAHAN PENGUJI

## KLASIFIKASI SPESIES ULAR MENGGUNAKAN DEEP LEARNING DENGAN ARSITEKTUR EFFICIENTNETV2S

Disusun Oleh:
Rizqi Ananta Ekta Putra
123210023

Telah diuji dan dinyatakan lulus oleh penguji
pada tanggal: ..22 Juli 2025..

Menyetujui,

| Penguji 1 | Penguji 2 |
| --- | --- |
| Rifki Indra Perwira, S.Kom., M.Eng | Juwairiah, S.Si, M.T, |
| NIDN. 0508078301 | NIDN. 0527077601 |
| **Penguji 3** | **Penguji 4** |
| Oliver Samuel Simanjuntak, S.Kom, M.Eng, | Budi Santosa, S.Si, M.T, |
| NIDN. 0525058302 | NIDN. 0510097001 |

# SURAT PERNYATAAN
# KARYA ASLI TUGAS AKHIR

Sebagai mahasiswa Program Studi Informatikai Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta, yang bertanda tangan di bawah ini, saya:

Nama : Rizqi Ananta Ekta Putra
NIM : 123210023

Menyatakan bahwa karya ilmiah saya yang berjudul:
**Klasifikasi Spesies Ular Menggunakan *Deep Learning* Dengan Arsitektur *EfficientNetV2S***

Merupakan karya asli saya dan belum pernah dipublikasikan dimanapun. Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, maka saya bersedia menerima konsekuensi apa pun yang diberikan Program Studi Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Yogyakarta
Pada Tanggal : 9 Juli 2025

Yang Menyatakan


Rizqi Ananta Ekta Putra
NIM. 123210023

# PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini:

Nama : Rizqi Ananta Ekta Putra
NIM : 123210023
Prodi : Informatika

Dengan ini saya menyatakan bahwa judul Tugas Akhir
**Klasifikasi Spesies Ular Menggunakan *Deep Learning* Dengan Arsitektur *EfficientNetV2S***

adalah hasil kerja saya dan benar bebas dari plagiasi kecuali cuplikan serta ringkasan yang terdapat di dalamnya telah saya jelaskan sumbernya (Sitasi) dengan jelas. Apabila pernyataan ini terbukti tidak benar maka saya bersedia menerima sanksi sesuai peraturan Mendiknas RI No 17 Tahun 2010 dan Peraturan Perundang-undangan yang berlaku.

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab.

Dibuat di : Yogyakarta
Pada Tanggal : 9 Juli 2025

Yang Menyatakan


Rizqi Ananta Ekta Putra
NIM. 123210023

# ABSTRAK

Di wilayah tropis, memiliki banyak keanekaragaman reptil dari berbagai spesies ular. Di Indonesia sendiri jumlah korban akibat gigitan ular terus mengalami peningkatan. Pada tahun 2017, tercatat 35 orang meninggal dunia karena gigitan ular, angka ini naik menjadi 47 orang pada 2018, dan kembali meningkat 54 orang pada 2019. Pada tahun 2020 bulan Januari hingga 2021, gigitan ular di seluruh Indonesia tercatat sekitar 627 kasus, 62 orang merupakan jumlah korban meninggal dunia. Gigitan ular berbisa memicu racun dapat memberikan dampak bahaya bagi tubuh manusia, dimulai dari mengganggu fungsi pernapasan, gangguan pendarahan, hingga dapat menyebabkan disabilitas permanen. Klasifikasi ular digunakan dalam membantu manusia untuk mengidentifikasi spesies ular, menggunakan klasifikasi ini dapat mengurangi resiko gigitan ular berbisa bagi manusia dan dapat mendukung menjaga ekosistem ular.

*EfficientNet* adalah salah satu arsitektur *Convolutional Neural Network* (CNN) dalam bidang *deep learning* digunakan pada pemrosesan dan pengenalan gambar. Dengan menerapkan transfer learning dari *EfficientNetV2S* yang terlatih sebelumnya pada dataset *ImageNet*. Selanjutnya pra-pemrosesan data dengan melakukan beberapa proses *augmentasi* yaitu *rotation range, zoom range, shear range, width, height,* dan *horizontal flip,* serta perubahan ukuran citra menjadi 384 x 384 piksel dan normalisasi piksel (*rescale*). Implementasi dilakukan dengan membekukan sebagian besar layer awal dari model *pre-trained EfficientNetV2S*. Pendekatan ini menggunakan bobot awal dari ImageNet dan memungkinkan model melakukan *fine-tuning* untuk menyesuaikan dengan data spesifik. Pengujian dilakukan dengan jumlah epoch 20 dengan *optimizer Adam* dan *learning rate* 0,0001.

Hasil penelitian ini, dapat disimpulkan dengan penerapan arsitektur *EfficientNetV2S* telah berhasil membangun sistem identifikasi spesies ular yang dapat diandalkan secara otomatis dan akurat. Penelitian ini tidak hanya berhasil menerapkan *Convolutional Neural Network* (CNN) dengan arsitektur *EfficientNetV2S* untuk membedakan ular berbisa dan tidak berbisa, tetapi juga membuktikan bahwa model ini mampu mencapai akurasi testing 93,12% klasifikasi berdasarkan dataset yang tersedia. Dengan akurasi training dan validasi yang konsisten di atas 90% dan nilai *loss* yang terus menurun, sistem menunjukkan kemampuan belajar yang efektif tanpa indikasi *overfitting* yang signifikan.



**Kata Kunci: Ular, Klasifikasi, *Deep Learning*, *CNN*, *Transfer Learning, EfficientNetV2S***

## *ABSTRACT*

*Tropical regions are home to a rich diversity of reptiles, including numerous snake species. In Indonesia, the number of victims from snakebites has continued to rise. In 2017, 35 fatalities were recorded, increasing to 47 in 2018, and rising again to 54 in 2019. From January 2020 to 2021, there were approximately 627 reported snakebite cases nationwide, with 62 resulting in death. Bites from venomous snakes can have serious consequences for human health, including respiratory failure, bleeding disorders, and even permanent disability. Snake classification plays an important role in helping people identify snake species, which can significantly reduce the risk of venomous bites and support efforts to preserve snake biodiversity.*

*EfficientNet is a Convolutional Neural Network (CNN) architecture widely used in image recognition and processing tasks within the field of deep learning. In this study, transfer learning was applied using the EfficientNetV2S model pre-trained on the ImageNet dataset. The image data underwent several preprocessing and augmentation steps, including rotation, zooming, shearing, width and height shifts, horizontal flipping, resizing to 384 × 384 pixels, and pixel normalization. Most of the early layers of the EfficientNetV2S model were frozen to retain their learned weights, allowing fine-tuning to adapt to the specific dataset used in this study. The training was conducted for 20 epochs using the Adam optimizer with a learning rate of 0.0001.*

*The results show that the use of the EfficientNetV2S architecture successfully built a reliable and accurate automated snake species identification system. Not only was the model able to distinguish between venomous and non-venomous snakes using CNN techniques, but it also achieved a test accuracy of 93.12% based on the available dataset. With both training and validation accuracy consistently above 90%, and steadily decreasing loss values, the system demonstrated effective learning performance without significant signs of overfitting.*



***Keywords: Snake Identification, Classification, Deep Learning, CNN, Transfer Learning, EfficientNetV2S***

# KATA PENGANTAR

Puji syukur saya panjatkan ke hadirat Allah SAW atas limpahan rahmat, karunia, dan izin-Nya, sehingga saya dapat menyelesaikan penelitian tugas akhir dengan judul "Klasifikasi Spesies Ular Menggunakan *Deep Learning* Dengan Arsitektur EfficientNetV2S" dengan baik. Penulisan tugas akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana (S1) pada Program Studi Informatika, Fakultas Teknik Industri, Universitas Pembangunan Nasional "Veteran" Yogyakarta.

Pada kesempatan kali ini, penulis ingin menyampaikan terima kasih kepada semua pihak yang telah memberikan dukungan dan bimbingan, sehingga tugas akhir ini dapat diselesaikan tepat waktu. Oleh karena itu, penulis juga mengucapkan terima kasih kepada :

1. Allah SWT, yang telah memberikan hidayah, kesabaran, petunjuk, dan kemudahan dalam menyelesaikan tugas akhir
2. Orang Tua dan Keluarga, yang selalu memberikan doa, dukungan, dan motivasi kepada saya.
3. Bapak Dr. Awang Hendrianto Pratomo, S.T., M.T., M.Eng. selaku dosen wali.
4. Bapak Rifki Indra Perwira, S.Kom., M.Eng, sebagai dosen pembimbing tugas akhir.
5. Ibu Juwairiah, S.Si., M.T., Bapak Oliver Samuel S. S.Kom., M.Eng., dan Bapak Budi Santosa, S.Si., M.T., selaku dosen penguji yang telah masukan saran dan kritik pada penulis.
6. Seluruh staff pengajar Jurusan Informatika, Fakultas Teknik Industri, Universitas Pembangunan Nasional "Veteran" Yogyakarta yang telah berbagi ilmu pengetahuan selama proses perkuliahan berlangsung.
7. Sahabat seperjuangan dan teman-teman mahasiswa Informatika 2021 yang telah memberikan doa dan motivasi serta bantuannya dalam pelaksanaan Tugas Akhir penulis.
8. Serta kepada semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah memberikan banyak dukungan selama penyusunan penelitian tugas akhir ini.

Penelitian pada tugas akhir ini, penulisa menyadari bawah masih jauh dari kata sempurna dan tidak terhindar dari kekurangan dan kesalahan. Oleh karna itu, penulis menerima dengan terbuka dan berterima kasih atas masukan dan saran yang diberikan. Semoga tugas akhir ini dapat memberikan kontribusi yang bermanfaat bagi semua pihak yang terkait. Terima kasih.

Yogyakarta, 18 Juni 2025

Penulis

**DAFTAR ISI**

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang Masalah

Di wilayah tropis, banyaknya aneka ragam reptil dari berbagai spesies seperti ular. Ular merupakan satwa liar yang sering kali muncul sekitar manusia. Reptil memiliki habitat yang dekat dekan manusia dan beberapa dapat ditemukan berbagai tempat seperti pohon, sawah, pekarangan, saluran air, bahkan kadang memasuki rumah warga. Ular merupakan kelompok reptilia atau hewan melata memiliki sifat hewan ektotermik atau berdarah dingin. Berarti ular tidak mampu menghasilkan panas tubuh secara mandiri dan harus bergantung pada suhu lingkungan untuk beraktivitas. Inilah yang menjadi salah satu alasan ular sering ditemukan di sekitar permukiman manusia (Dafa & Suyanto, 2021). Di Indonesia sendiri jumlah korban akibat gigitan ular terus mengalami peningkatan. Pada tahun 2017, tercatat 35 orang meninggal dunia karena gigitan ular, angka ini naik menjadi 47 orang pada 2018, dan kembali meningkat menjadi 54 orang pada 2019. Pada tahun 2020 bulan Januari hingga 2021, gigitan ular di seluruh Indonesia tercatat sekitar 627 kasus, 62 orang merupakan jumlah korban meninggal dunia (Pandangan Jogja, 2021). Indonesia ialah negara dengan banyaknya keanekaragaman hayati termasuk beragam spesies ular yang ada di dalamnya. Karena habitat ular cukup dekat dengan manusia maka sering terjadi perselisihan, di mana habitat ular berbisa dan tidak berbisa melimpah di Indonesia. Ketidaktahuan dalam membedakan spesies ular berbisa dan tidak berbisa sering memicu ketakutan atau penanganan yang salah, yang berujung pada risiko gigitan ular. Gigitan ular adalah salah satu penyakit paling mematikan. Ini menyebabkan antara 81000 dan 138000 kematian dan 400.000 korban mengalami cacat fisik atau kelainan permanen secara global setiap tahun (Ahmed et al., 2024).

Penelitian ini membantu manusia dalam menghindari risiko dan dapat mengambil keputusan yang tepat saat berhadapan dengan ular. Gigitan ular bisa menjadi keadaan darurat medis yang serius hingga dapat membahayakan nyawa manusia. Bisa atau racun ular dapat memberikan dampak bahaya bagi tubuh manusia, dimulai dari mengganggu fungsi pernapasan, gangguan pendarahan, hingga dapat menyebabkan disabilitas permanen (Wintoko & Prameswari, 2020). Ketakutan manusia terhadap ular, baik berbisa maupun tidak berbisa, sering kali memicu tindakan berlebihan, seperti membunuh ular tanpa mengetahui dampaknya terhadap ekosistem. Karena ketakutan manusia dengan membunuh ular akan merugikan diri sendiri, karena dengan adanya ular tanpa membunuhnya dapat membantu menyeimbangkan ekosistem lingkungan. Jika ular terus dibasmi, maka mungkin akan mengakibatkan muncul masalah baru seperti, populasi tikus meningkat yang dapat merugikan manusia itu sendiri. Sebagian jenis ular, termasuk yang berbisa seperti cobra, sebenarnya takut pada manusia dan cenderung menghindar. Mereka tidak akan menyerang kecuali merasa terancam atau diprovokasi. Meskipun terlihat di permukiman, kasus gigitan sangat jarang, namun kewaspadaan tetap penting saat berhadapan dengan ular berbisa (Marida & Radhi, 2019).

Pembelajaran mesin diperlukan untuk membedakan antara ular berbisa dan tidak berbisa pada masalah ini. *Deep Learning* (DL) merupakan sebuah teknik berbasis jaringan

saraf tiruan telah banyak digunakan dalam beberapa tahun terakhir sebagai salah satu metode *Machine Learning* (ML). *Deep Learning* dapat diterapkan diberbagai bidang, dengan penerapan ini sebagai bentuk pembelajaran umum dan bisa digunakan untuk menyelesaikan beragam masalah di berbagai sektor. (Rakhmat & Rizkiawarman, 2023). *EfficientNet* adalah salah satu arsitektur *CNN* dalam bidang *deep learning* yang khusus digunakan untuk pemrosesan dan pengenalan gambar. Dalam penelitian ini, digunakan arsitektur *EfficientNetV2S*, salah satu model *CNN* yang memiliki performa tinggi dalam klasifikasi gambar. *EfficientNet*, yang dirancang dengan pendekatan efisiensi tinggi, menawarkan kombinasi optimal antara akurasi dan kecepatan komputasi. *EfficientNet* merupakan model deep learning dengan pendekatan *compound scaling* yang biasa digunakan untuk melakukan kerja-kerja pengolahan citra (Tan & Le, 2019). *EfficientNet* dapat menjadi pilihan yang tepat untuk mengatasi tantangan dalam klasifikasi gambar, karena model da pat mengenali pola visual secara mendalam termasuk untuk gambar identifikasi spesies tertentu seperti spesies ular berbisa dan tidak berbisa. *EfficientNetV2*, sebagai salah satu varian dari *EfficientNet*, telah diaplikasikan secara luas pada berbagai tugas klasifikasi gambar dan terbukti memberikan performa yang luar biasa baik dari segi akurasi maupun efisiensi.

Penelitian sebelumnya yang menggunakan arsitektur *EfficientNetV2S* yaitu degan judul "*Evaluasi Ensemble Stacking Arsitektur EfficientNetB3 dan EfficientNetV2S (Studi Kasus Klasifikasi Penyakit Alzheimer)*" oleh (Kasus et al., 2023). Penelitian tersebut mengatakan bahwa model *EfficientNetV2S* mampu memberikan performa yang lebih baik dibandingkan *EfficientNetB3*. Hal ini dapat dilihat dari akurasi validasi yang lebih tinggi untuk *EfficientNetV2S* 95,45% dengan nilai *loss* 0,2275 sedangkan *EfficientNetB3* 92,54% dengan nilai *loss* 0,3726. Dari hasil tersebut arsitektur *EfficientNetV2S* tidak hanya lebih akurat dalam mengenali pola dari data, tetapi juga lebih baik dalam menghindari *overfitting*.

Selain penelitian di atas ada juga penelitian yang dilakukan oleh (Yuniari et al., 2024) yaitu "*Analisis Klasifikasi Citra Penokohan Topeng Bali Menggunakan Model EfficientnetV2 Dan Xception*". Model *Xception* menghasilkan akurasi sebesar 97%, sementara model *EfficientNetV2* bekerja lebih baik dengan menghasilkan akurasi 99%.

Pada penelitian lain dengan objek ular, yaitu menggunakan *CNN* dengan arsitektur *ConvNeXt* oleh (Qisthan, 2023) dalam penerapan "Analisis Performa Metode *Convolutional Neural Network* Dengan Arsitektur *ConvNeXt* Dalam Klasifikasi Spesies Ular Berbisa Dan Tidak Berbisa Di Indonesia", berdasarkan penelitian dengan menggunakan dataset berupa 18 spesies ular dengan jumlah 3959 gambar. Hasil yang didapatkan dari arsitektur model *ConvNeXt* terbukti mampu melakukan klasifikasi spesies ular dengan hasil yang memuaskan. Dalam penelitian ini, varian *ConvNeXtXLarge* menunjukkan performa terbaik dengan mencapai *accuracy, precision, recall*, dan *F-1 Score* sebesar 92% setelah 40 *epoch*, mengungguli variasi *ConvNeXt* lainnya.

Selain penelitian tersebut ada penelitian lain yang menggunakan objek yang serupa oleh (Bloch & Friedrich, 2021), penelitian tersebut menerapkan model dengan versi *EfficientNetB4* berjudul "*EfficientNets and Vision Transformers for Snake Species Identification Using Image and Location Information*." Mereka menggunakan dataset yang berisi 772 spesies ular. Menggunakan dua model berbeda yaitu *EfficientNet-B4* dan *Vision Transformer* (ViT). Hasil penelitian menunjukkan bahwa penggabungan prediksi model dan

informasi lokasi meningkatkan akurasi, dengan ensembel kedua model mencapai skor rata-rata *F1-makro* sebesar 82,88% pada data uji independen. Meskipun begitu terdapat saran pada penelitian ini untuk melakukan eksplorasi lebih lanjut penggunaan arsitektur *EfficientNetV2S* yang memiliki kecepatan proses pelatihan dan lebih efisien daripada *EfficientnetV1.*

Berdasarkan saran oleh (Bloch & Friedrich, 2021) menggunakan *EfficientNetB4*, penelitian ini mengeksplorasi potensi *EfficientNetV2S* sebagai percobaan lebih. *EfficientNetV2S* memiliki kelebihan dalam efisien dengan kemampuan yang lebih ringan dan proses pelatihan lebih cepat. Dalam penelitian ini, menggunakan dan mengimplementasi metode klasifikasi gambar menggunakan *EfficientNetV2S* untuk mengenali spesies ular berbisa dan tidak berbisa menggunakan dataset diambil dari *kaggle* "*Different Snakes Species*" dan melakukan *web scrapping* unutk menambah dataset, kemudian diambil jenis ular yang banyak ditemui di wilayah tropis dengan jumlah 20 spesies ular diantaranya 10 berbisa dan 10 tidak berbisa. Model dikembangkan memakai *Convolutional Neural Network (CNN)* berbasis arsitektur *EfficientNetV2S*, dengan klasifikasi, teknik seperti *augmentasi* data dan optimisasi *hyperparameter* akan digunakan. Model ini juga akan dirancang untuk memberikan hasil klasifikasi beserta probabilitas prediksi, sehingga dapat memberikan informasi lengkap dan mendukung pengambilan keputusan yang lebih akurat.

Dengan penelitian ini, diharapkan mampu menghasilkan akurasi yang akurat tetapi juga mampu memberikan probabilitas klasifikasi untuk mendukung pengambilan keputusan yang lebih andal. Penelitian penerapan *EfficientNetV2S* ini dapat membantu masyarakat maupun peneliti dalam mengetahui spesies ular berbisa atau tidak, dengan tujuan untuk meningkatkan kesadaran dan keamanan masyarakat dalam menghadapi ular. Karena ular juga penting dalam mengendalikan perannya di dalam ekosistem. Peran model digunakan untuk membantu masyarakat untuk lebih memahami peran ular di alam sekaligus mengurangi ketakutan yang berlebihan terhadap mereka. Hingga masyarakat dapat mengidentifikasi apakah seekor ular berbisa atau tidak tanpa harus langsung menghadapi risiko. Hal ini dapat membantu mereka mengambil langkah yang tepat, seperti menjauh dari ular berbisa atau membiarkan ular non-berbisa kembali ke habitat alaminya.

## 1.2 Rumusan Masalah

Pada penelitian ini rumusan masalahnya adalah sebagai berikut:

1. Penerapan pengolahan citra berbasis *Convolutional Neural Network* arsitektur *EfficientNetV2S* dalam klasifikasi *multiclass* citra spesies ular.
2. Pengaruh penggunaan arsitektur EfficientNetV2S terhadap tingkat akurasi klasifikasi spesies ular.

## 1.3 Batasan Masalah

Batasan masalah ini dirancang agar untuk membantu penelitian agar tidak terlalu luas yang akan dibahas, adapun cakupan batasan masalah pada penelitian ini sebagai berikut :

1. Menggunakan dataset dari *kaggle "https://www.kaggle.com/datasets/goelyash/165-different-snakes-species"* dan melakukan *web scrapping* dengan kategori ular dipilih 20 spesies, dengan total gambar sebanyak 4640.

2. Spesies ular yang akan diklasifikasi merupakan 20 spesies ular diwilayah tropis khususnya 18 spesies terdapat di Indonesia diantaranya 10 berbisa dan 10 tidak berbisa.
3. Citra Ular pada *input* yang digunakan dalam pengujian hanya satu jenis citra ular.
4. Citra Ular pada *input* yang digunakan dalam pengujian adalah merupakan 20 spesies ular yang ada pada dataset yaitu *Ahaetulla prasina, Bitis gabonica, Boiga irregularis, Boiga kraepelini, Bungarus multicinctus, Chrysopelea ornata, Daboia russelii, Dendrelaphis pictus, Gonyosoma oxycephalum, Laticauda colubrina, Lycodon capucinus, Morelia viridis, Ophiophagus hannah, Protobothrops mucrosquamatus, Psammodynastes pulverulentus, Python molurus, Rhabdophis subminiatus, Tropidolaemus subannulatus, Tropidolaemus wagleri,* dan *Xenochrophis piscator.*

## 1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk implementasi model klasifikasi.

1. Menerapkan *Convolutional Neural Network* dengan arsitektur *EfficientNetV2S* untuk klasifikasi spesies ular.
2. Mengetahui akurasi dari arsitektur *EfficientNetV2S* dalam mendeteksi spesies ular berdasarkan *dataset* yang tersedia dan memberikan solusi praktis untuk klasifikasi spesies ular.

## 1.5 Manfaat Penelitian

Adapun manfaat yang diharapkan dari melakukan penelitian ini, sebagai berikut:

1. Penelitian ini diharapkan dapat mempermudah proses klasifikasi spesies ular, baik berbisa maupun tidak.
2. Dengan penerapan model klasifikasi ini diharapkan juga dapat menjadi referensi untuk pengembangan teknologi serupa pada bidang lain.
3. Penelitian ini, dapat memberikan kontribusi dalam kemajuan ilmu pengetahuan, terutama dalam penerapan *Deep Learning* untuk masalah klasifikasi berbasis gambar.

## 1.6 Tahapan Penelitian

Berikut merupakan tahapan penelitian yang akan dilakukan:

1. Studi Literatur

Menentukan tujuan penelitian, yaitu penerapan model klasifikasi spesies ular berbisa dan tidak berbisa menggunakan *Deep Learning*.

2. Pengumpulan Data

Data yang digunakan berupa citra ular yang diperoleh melalui *kaggle* atau *web scrapping* melalui *website GBIF*.

3. Data *Preprocessing*

Data akan melalui tahap *preprocessing* untuk meningkatkan variasi data. Pada tahap ini akan dilakukan *augmentasi* data untuk mendapatkan variasi data yang akan digunakan dalam proses pelatihan, uji, dan validasi.

4. *Modeling*

Menerapkan arsitektur *EfficientNetV2S* dengan metode *transfer learning* menggunakan bobot yang telah dilatih sebelumnya pada dataset *ImageNet* dengan menggunakan *TensorFlow/Keras*. Model akan diuji dengan data validasi dan diuji dengan data pengujian.

5. Evaluasi Model

   Mengevaluasi performa model menggunakan akurasi, *confusion matrix*, *precision*, *recall*, dan *F1-score* untuk memastikan hasil klasifikasi yang optimal.
6. *Deployment* mengimplementasikan model ke dalam *platform website*.

## 1.7 Sistematika Penulisan

Sistematika penulisan tugas akhir ini disusun untuk memudahkan pemahaman tentang struktur dan isi tugas akhir, sistematika ini disusun sebagai berikut:

**Bab I Pendahuluan**

Bab I ini menjelaskan latar belakang masalah, perumusan masalah, tujuan penelitian, manfaat penelitian, serta sistematika pembahasan dari penelitian ini. Bagian ini bertujuan untuk memberikan gambaran umum tentang alasan dilakukannya penelitian dan struktur keseluruhan laporan penelitian.

**Bab II Tinjauan Literatur**

Bab II ini berisi kajian literatur yang relevan dengan topik penelitian, termasuk teori-teori dasar mengenai pengolahan citra digital, *Convolutional Neural Network* (CNN) dengan arsitektur *EfficientNetV2S*, serta teknik pengolahan citra digital. Selain itu, dibahas juga penelitian terdahulu yang berhubungan dengan klasifikasi ular, dan metode evaluasi performa model. Tinjauan ini bertujuan untuk menyediakan landasan teoritis yang mendukung penelitian.

**Bab III Metodologi Penelitian**

Bab III ini menjelaskan metodologi penelitian yang digunakan, mulai dari tahap pengumpulan dan *preprocessing* data, proses *augmentasi* gambar, arsitektur model yang digunakan, serta teknik *fine-tuning* untuk meningkatkan akurasi. Selain itu, dijelaskan pula skenario pelatihan model, serta pengujian dengan metrik evaluasi yang digunakan dalam mengukur performa model. Bab ini bertujuan untuk menjelaskan langkah-langkah sistematis yang diambil dalam pelaksanaan penelitian.

**Bab IV Hasil dan Pembahasan**

Bab IV ini memaparkan hasil-hasil yang diperoleh dari penelitian, termasuk hasil terhadap performa model *EfficientNetV2S* dalam klasifikasi spesies ular. Hasil evaluasi model dibandingkan dengan penelitian sebelumnya serta dibahas implikasi dari penerapan metode yang digunakan.

**Bab V Penutup**

Bab V ini berisi kesimpulan yang diambil dari hasil penelitian dan untuk menjawab rumusan masalah yang diuraikan, serta saran penelitian dibahas lebih lanjut. Kesimpulan ini bertujuan untuk merangkum temuan utama dan memberikan rekomendasi yang dapat bermanfaat bagi pengembangan teknologi di masa mendatang.

# BAB II
# TINJAUAN LITERATUR

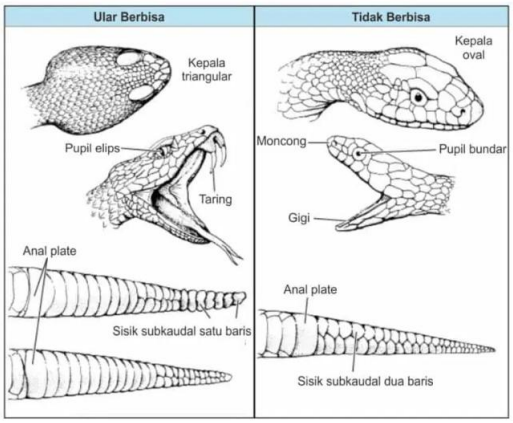
## 2.1 Ular

Ular termasuk kelompok reptil karnivora yang tidak memiliki kaki dan bertubuh panjang dalam ordo *Squamata*, sub-ordo *Serpentes*. Terdapat sekitar 3.000 spesies ular yang dapat ditemukan di hampir semua habitat di dunia, kecuali Antartika, Islandia, Irlandia, Greenland, dan Selandia Baru (BRI, 2023). Terdapat lebih dari 349 spesies ular yang hidup di Indonesia, jumlah ini dapat bertambah lagi karena eksplorasi terhadap keanekaragaman ular di Indonesia masih berlangsung. Dari jumlah tersebut spesies ular di Indonesia mencakup sekitar 10% dari total spesies ular yang ada di seluruh dunia. (BRIN, 2023). Ular memiliki bentuk tubuh yang beradaptasi di berbagai jenis habitat. Ular menjadi salah satu reptilia yang berhasil dapat beradaptasi dan tersebar di seluruh dunia, ular dapat ditemukan di berbagai ekosistem seperti hutan, padang rumput, gurun, sungai, danau, dataran tinggi, sawah, laut, dan juga di pemukiman manusia. Keberagaman jenis ular dan kemampuan untuk beradaptasi di berbagai ekosistem menjadikan ular sebagai salah satu reptil yang memiliki peran penting dalam menjaga keseimbangan ekosistem, yaitu sebagai pengedali populasi hewan pengerat dan serangga sebagai mangsa bagi ular.

Ular memiliki peran penting sebagai predator yang membantu menjaga keseimbangan rantai makanan, dengan struktur tubuh ular yang unik seperti rahang bawah yang fleksibel untuk menelan mangsa yang lebih besar, kulit bersisik yang mempermudah gerak, serta lidah sebagai sensor untuk menangkap molekul bau. Ular menggunakan sensor dalam mendeteksi dan melacak mangsa, melalui organ *Jacobson* yang berfungsi mendeteksi partikel udara saat menjulurkan lidahnya, kemudian menarik lidahnya dan memasukkan ke dalam organ *Jacobson*. Keberadaan ular juga sering muncul di pemukiman masyarakat yang sering kali menimbulkan kepanikan, terutama ular sering masuk ke area rumah, kebun, sawah yang selalu dilihat oleh manusia. Karena manusia sering kali menemui ular di pemukiman, kemudian timbul rasa takut bahkan menganggap ular sebagai ancaman yang harus dihindari atau dimusnahkan.

Dalam ekosistem, ular memiliki peran penting yaitu dalam menjaga keseimbangan lingkungan. Jika ular punah maka akan terjadi gangguan dalam rantai makanan karena populasi hewan yang menjadi mangsa ular bisa berkembang tanpa kendali, yang berakibat pada ketidakseimbangan ekosistem. (Pantur et al., 2024). Kekhawatiran hingga membuat takut manusia terhadap ular yang dianggap bahwa semua ular berbahaya, padahal ular memiliki ribuan spesies yang ada di dunia. Dari ribuan spesies tersebut hanya sebagian tergolong ular berbisa (*Venomous*) yang dapat membahayakan manusia, sebaliknya sebagian besar ular yang ditemukan di sekitar pemukiman merupakan jenis ular tidak berbisa (*Non-Venomous*) yang memiliki manfaat dalam pengendali ekosistem yaitu dengan memangsa hewan pengerat, serangga, dan hewan kecil lainnya.

### 2.1.1 Morfologi Ular

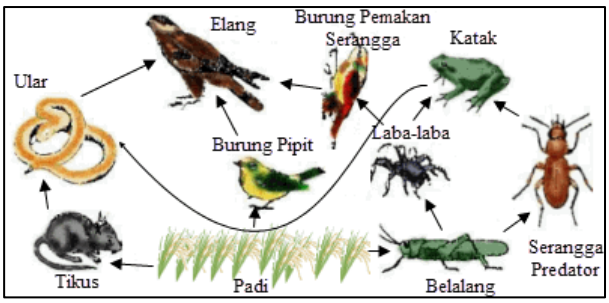



**Gambar 2. 1** Morfologi Ular

Diperkirakan ada sekitar 2800 spesies ular di dunia dan sekitar 320 di antaranya memiliki peran penting bagi manusia. Terdapat tiga famili ular berbisa seperti *Atractaspididae*, *Elapidae,* dan *Viperidae*. Sementara itu famili *Colubridae* merupakan yang terbesar namun sebagian besar anggotanya tidak berbahaya meskipun ada beberapa laporan kasus gigitan yang disebabkan. Morfologi ular juga dapat digunakan untuk membedakan antara ular berbisa dan tidak berbisa. Seperti yang ditunjukkan pada Gambar 2.1, terdapat beberapa ciri khas yang dapat diamati secara morfologis. Ular berbisa umumnya memiliki kepala berbentuk segitiga (triangular), pupil mata berbentuk elips, dan memiliki taring panjang yang berfungsi menyuntikkan bisa ke mangsanya. Selain itu, sisik subkaudal ular berbisa biasanya tersusun dalam satu baris. Sebaliknya, ular tidak berbisa memiliki kepala berbentuk oval, pupil mata bulat, serta tidak memiliki taring berbisa melainkan gigi-gigi kecil yang tidak berfungsi untuk menyuntikkan racun. Sisik subkaudal pada ular tidak berbisa tersusun dalam dua baris, yang menjadi salah satu indikator pembeda yang penting.

Pengetahuan mengenai perbedaan morfologi ini sangat penting terutama dalam situasi darurat seperti kasus gigitan ular, di mana identifikasi cepat dapat membantu menentukan penanganan medis yang tepat. Meskipun demikian, identifikasi berdasarkan morfologi tidak selalu akurat, sehingga kehati-hatian tetap diperlukan, dan sebisa mungkin identifikasi harus dikonfirmasi oleh ahli herpetologi atau tenaga medis terlatih.

### 2.1.2 Ekologi Ular




**Gambar 2. 2** Ekosistem

Ular dapat ditemukan hampir ada di seluruh benua kecuali Antartika, dengan kemampuan beradaptasi ular yang tinggi terhadap beragam kondisi ekologi. Kebanyakan

jenis ular lebih banyak di wilayah tropis dibandingkan dengan wilayah lain di bumi. Hal ini disebabkan oleh lingkungan dan mikrohabitat yang mendukung seperti iklim yang sesuai, melimpahnya jenis mangsa, serta faktor lain seperti sejarah geografis kawasan tersebut. (Fathoni, 2019). Kemampuan adaptasi yang dimiliki ular membuat spesies ular tersebar di berbagai habitat serta memiliki keanekaragaman spesies. Dengan memiliki tubuh yang lentur memungkinkan ular bergerak secara melata melalui otot-otot perut secara mudah di berbagai medan, mulai dari tanah, pohon, air.

Secara ekologis, ular menjadi perang penting di dalam ekosistem pada Gambar 2.2, yaitu sebagai pengendali populasi hewan seperti hama, hewan pengerat, katak, burung kecil, hingga serangga. Ular juga sebagai mangsa bagi beberapa spesies predator rantai makanan lain seperti elang, burung hantu, mamalia besar seperti musang, bahkan ular lain. Dalam rantai makanan ular berperan sebagai konsumen tingkat trofik tiga atau tingkat atas yang membantu mengendalikan populasi mangsanya. Jika jumlah ular menurun di suatu habitat maka populasi mangsanya bisa meningkat secara drastis dan mengganggu keseimbangan ekosistem. (Fathoni, 2019). Jika berkurangnya populasi ular dikarenakan pemburuan atau hilangnya habitat ular dapat berakibat fatal karena menyebabkan gangguan ekosistem, kondisi ini dapat memicu populasi mangsa ular seperti tikus, serangga meningkat yang dapat mengakibatkan penyebaran penyakit kepada manusia dan juga merusak tanaman pertanian.

Morfologi ular selain dipengaruhi oleh faktor genetik tetapi juga oleh lingkungan tempat mereka hidup. Kondisi geografis dan jenis habitat berperan besar dalam membentuk penampilan fisik ular di berbagai daerah. (Plateau et al., 2018). Habitat dari spesies ular memperhatikan keberadaan mangsanya. Ular juga termasuk hewan berdarah dingin, sehingga aktivitas dipengaruhi keadaan lingkungan, khususnya suhu dan kelembaban. Selain itu ular juga harus mengatur kondisi lingkungan termasuk bersembunyi untuk menghindari predator, mencerna makanan, reproduksi, dan tidur. Banyak masyarakat takut pada ular, karena tidak memahami perannya pada ekosistem. Ular yang datang ke pemukiman biasanya ular tidak berbisa dan justru membantu mengendalikan hama.

## 2.2 Citra Digital

Citra merujuk pada media yang berisi informasi tentang representasi atau imitasi visual suatu objek (Khairunnas et al., 2021). Citra digital dapat diibaratkan sebagai matriks yang memiliki baris dan kolom. Matriks tersebut menunjukkan setiap titik pada citra memiliki nilai elemen matriks yang menunjukkan warna pada titik-titik tersebut. Setiap kumpulan titik pada citra digital dikenal sebagai piksel (*pixel* atau "*picture element*" yaitu digambarkan sebagai kotak kecil. Penulisan citra digital sebagaimana fungsi intensitas f(x, y), yaitu nilai x(baris) dan y(kolom) menunjukkan koordinat piksel. Kemudian f(x, y) ialah nilai fungsi pada setiap titik (x, y) menyatakan nilai intensitas, keabuan, atau warna pada titik tersebut. Jenis citra berbeda-beda berdasarkan nilai piksel (Gustiawan et al., 2023).

1. Citra biner dengan setiap piksel hanya terdiri dari warna hitam atau putih. Citra ini sering disebut juga sebagai citra hitam putih (*black and white*) atau citra monokrom. maka hanya perlu 1 bit per piksel (0 dan 1) atau apabila dalam 8 bit ( 0 dan 255).
2. Citra *grayscale* setiap pikselnya menampilkan warna gradasi mulai dari putih sampai hitam, yang menghasilkan warna abu-abu. Nilai intensitas pada citra ini berkisar dari

0 hingga 255, di mana 0 mewakili hitam dan 255 mewakili putih. Jumlah gradasi warna yang ditampilkan tergantung pada jumlah bit yang tersedia di memori untuk menyimpan data warna. Pada citra ini, warna abu-abu menunjukkan tingkat intensitas sedang, hitam mewakili intensitas rendah, dan putih menunjukkan intensitas tinggi.

3. *Color Image* atau *RGB,* citra yang terbentuk dari kombinasi tiga warna dasar, yaitu merah (*Red*), hijau (*Green*), dan biru (*Blue*). Setiap warna dasar disimpan dalam 8 bit atau setara dengan 1 byte, yang memungkinkan setiap warna memiliki 256 tingkat gradasi, mulai dari 0 hingga 255. Dari kombinasi tersebut dalam satu piksel dapat menghasilkan lebih dari 16 juta warna (256 × 256 × 256). Jenis citra ini sering disebut *true color*, karena mewakili hampir seluruh warna yang ada di alam dengan jumlah warnanya sangat banyak. Penyimpanan citra ini juga membutuhkan ruang memori yang lebih besar dibandingkan jenis citra yang lainnya.

### 2.2.1 Pengolahan Citra

Pengolahan citra ialah tahapan memodifikasi gambar digital dengan menggunakan teknik tertentu untuk menghasilkan gambar baru atau memperoleh informasi (Saifullah, 2020). Dalam pengolahan citra, gambar diambil diambil pertama kali menggunakan alat *input* seperti kamera digital, lalu diubah menjadi format digital oleh sistem. Setelah citra diambil menggunakan dari perangkat, hasilnya berupa gambar digital kemudian disimpan di memori. Gambar hasil penyimpanan merupakan citra warna, akan diproses lebih lanjut melalui tahapan awal (preprocessing) hingga tahapan akhir, tergantung pada tujuan pengolahan.

Tahapan pengolahan citra, gambar pertama kali diambil melalui alat *input*, kemudian akan diolah citra sesuai fitur yang dipilih, lalu citra yang telah diolah akan ditampilkan. Pengolahan citra memiliki beberapa teknik utama yaitu *preprocessing*, kompresi gambar, dan *segmentasi*. Citra warna memiliki 3 komponen yiatu merah (red), hijau (green), biru (blue). Masing-masing komponen warna memiliki representasi nilai dengan range 0- 255 (Saifullah, 2020). Proses pengolahan citra bertujuan dalam memperbaiki kualitas gambar, mengekstrak informasi penting, atau mengubah citra menjadi bentuk yang lebih mudah dianalisis sesuai tujuan yang diinginkan.

### 2.2.2 Klasifikasi Citra

Klasifikasi ialah tahapan dalam mengelompokkan banyak data ke dalam label yang mewakili konsep atau jenis data tertentu. Dalam klasifikasi, pada metode *Supervised Learning* data akan dilatih oleh sistem yang telah dikelompok-kelompokkan sesuai dengan label yang sudah diberikan. Salah satu metode dalam *Supervised Learning* adalah *Neural Network* dimana pembelajaran ini terbuat dari kumpulan neuron sederhana yang disusun dalam beberapa lapisan dan saling terhubung melalui nilai-nilai tertentu. Hal tersebut meniru proses kerja biologis dari *neuron* atau cara kerja otak manusia (Jeremia Bu'ulölö et al., 2021). Hasil dari klasifikasi akan disimpan, sehingga sistem dapat memahami hubungan antara ciri-ciri dari citra dan label yang sesuai. Dengan adanya metode ini, sistem kemudian dapat langsung mengklasifikasikan data baru yang belum dikenali berdasarkan pola yang telah dipelajari sebelumnya.

### 2.2.3 *Augmentasi* Data Citra

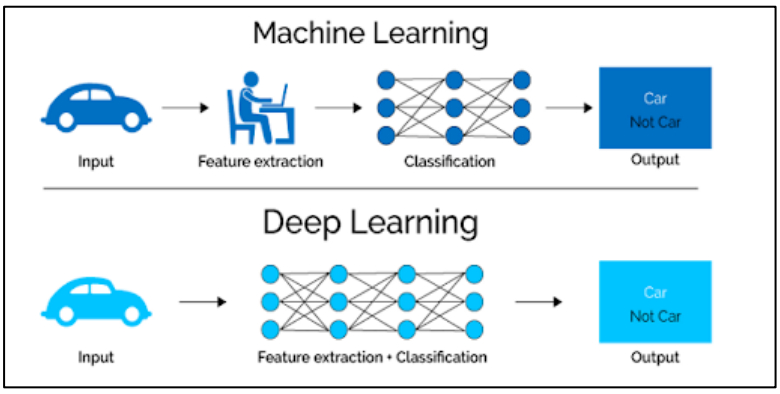
*Augmentasi* data pada citra merupakan tahapan proses untuk mengolah data gambar yang melibatkan modifikasi atau perubahan pada gambar tersebut sedemikian rupa. Kemudian dari hasil perubahan tersebut, sistem menganggapnya sebagai gambar yang berbeda, tetapi manusia masih dapat mengenali bahwa gambar tersebut sebenarnya sama. *Augmentasi* data bertujuan untuk meningkatkan variasi sampel data. Dengan teknik ini dapat mencegah situasi *overfitting* dalam proses pelatihan.

Teknik *augmentasi* data digunakan untuk meningkatkan kemampuan generalisasi model. Sehingga dapat menciptakan variasi data, model dapat lebih baik dalam menangkap pola penting dan lebih stabil terhadap variasi dalam data *testing* yang belum pernah dilihat sebelumnya (Shorten & Khoshgoftaar, 2019).

## 2.3 *Deep Learning*

*Deep learning* adalah cabang turunan dari bidang *machine learning* (pembelajaran mesin) yaitu dengan menggunakan *artificial neural network* (jaringan saraf tiruan) yang memiliki banyak lapisan (*neural networks*) yang meniru mekanisme otak manusia untuk menganalisis, memproses data dan membuat keputusan.

*Neural Network* dikembangkan agar dapat mensimulasikan perilaku otak manusia yang memungkinkannya untuk belajar dari sejumlah data besar. Dengan kemampuan *deep learning* dalam mengolah data tidak terstruktur seperti citra, audio, dan teks, deep learning mampu memberikan performa yang melampaui pendekatan pembelajaran konvensional. (Muhammadiyah Mataram et al., 2024).




**Gambar 2. 3** Arsitektur *Machine Learning* dan *Deep Learning*

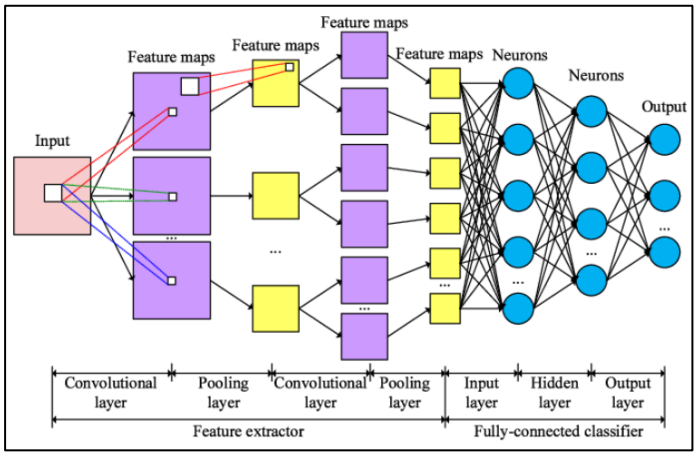
Dapat dilihat dari perbedaan Gambar 2.3 *arsitektur deep learning* berbeda dengan *machine learning* tradisional yang membutuhkan manusia untuk mengekstrak fitur penting dari data sebelum dilakukan klasifikasi, *deep learning* mampu melakukan ekstraksi fitur dan klasifikasi sekaligus dalam satu proses *end-to-end*. Kemampuan *deep learning* dalam menangani data yang lebih kompleks sangat efektif seperti gambar, suara, dan teks.

D*eep learning* berperan dalam mengatasi permasalahan data dengan skala besar sepeti seperti pada bidang *Computer Vision,* pengenalan suara *(Speech Recognition),* dan pemrosesan bahasa alami *(Natural Language Processing)*. *Deep learning* ialah cabang dari *Machine learning* yang menitu bagaima cara kerja *kortex* pada otak manusia, yaitu dengan menggunakan jaringan syaraf buatan yang memiliki banyak hiden layer (Khaeriyah, 2021).

## 2.4 *Convolutional Neural Network* (CNN)

*Convolutional Neural Network* adalah salah satu tipe *multi-layer neural network* dan juga arsitektur *deep learning* yang terinspirasi dari cara melihat makhluk hidup. Penemuan *Convolutional Neural Network* (CNN) awalnya dilakukan oleh Hubel dan Wiesel mengenai korteks visual pada kucing pada penelitian mereka. CNN merupakan sebuah struktur arsitektural yang dapat dilatih, yang memiliki beberapa tahapan yang meliputi *Input* dan *output*.

CNN adalah pengembangan dari *Multi-Layer Perceptron (MLP)* yang dioptimalkan digunakan dalam mengolah data dua dimensi, terutama pada pengolahan citra. CNN termasuk dalam kategori *Deep Neural Network* karena memiliki banyak lapisan *Deep Learning*, dan umumnya digunakan untuk tugas-tugas yang melibatkan data citra. *CNN* banyak digunakan dalam pengenalan gambar, deteksi objek, dan berbagai aplikasi pengolahan citra karena kemampuannya mengenali pola visual secara otomatis dan akurat. Arsitektur CNN dapat dilihat pada Gambar 2.4 berikut yang menggambarkan proses-proses yang terjadi pada.



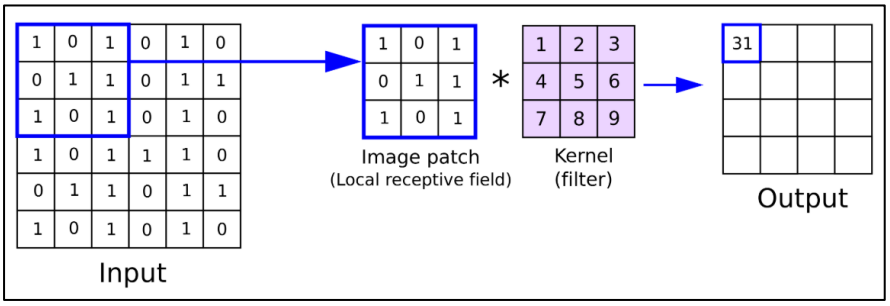

**Gambar 2. 4** Arsitektur *CNN*

Berdasarkan pada Gambar 2.4, struktur CNN Proses dimulai dari *input layer*, di mana citra dimasukkan dalam bentuk piksel digital. Selanjutnya, citra melewati beberapa *convolutional layer* yang berfungsi untuk mengekstraksi fitur penting dari gambar, seperti tepi, bentuk, dan tekstur, menggunakan *filter* atau kernel. Setelah itu, hasil dari konvolusi akan diproses oleh *pooling layer* untuk mereduksi dimensi data sambil tetap mempertahankan informasi utama. Tahapan konvolusi dan *pooling* ini biasanya dilakukan berulang kali untuk mendapatkan representasi fitur yang semakin kompleks dan abstrak. Setelah fitur diekstraksi sepenuhnya, hasilnya diratakan (*flattening*) dan dimasukkan ke dalam *fully-connected layers* yang terdiri dari *input layer*, *hidden layers*, dan *output layer*. Di tahap ini, jaringan akan melakukan proses klasifikasi berdasarkan fitur yang telah diperoleh dan menghasilkan output akhir berupa kelas atau label dari citra tersebut.

### 2.4.1 *Convolutional layer*

Pada lapisan konvolusi ini menggunakan *filter* atau kernel untuk mendeteksi karakteristik dari sebuah objek atau gambar, proses ini menghasilkan linear berdasarkan

gambar *input* yang sesuai. Dalam proses setiap lapisan konvolusi, terdapat beberapa parameter yang dapat disesuaikan untuk memodifikasi setiap lapisan diantaranya ada jumlah *filter*, ukuran langkah (*stride*), dan *padding*. *Stride* berfungsi untuk mengatur seberapa jauh *filter* bergerak melintasi gambar saat membaca data, *filter* tersebut akan bergerak berpindah posisi ke posisi lain di sepanjang ukuran piksel yang telah ditentukan. *Padding* digunakan untuk mengontrol ukuran *output* dan mencegah kehilangan informasi di tepi gambar. Pada lapisan ini, digunakan *filter* berukuran 3x3 dengan *Stride* = 1, yang akan digeser sebanyak 1 *Stride* di seluruh bagian, seperti contoh pada Gambar 2.5.

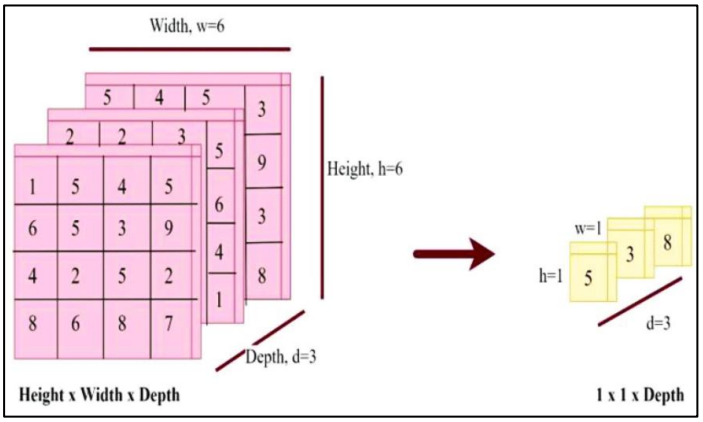



**Gambar 2. 5** *Convolution Layer*

### 2.4.2 *Pooling layer*

*Pooling layer* adalah untuk mengurangi jumlah perhitungan dalam jaringan saraf, sehingga meningkatkan efisiensi dan membantu mengurangi risiko *overfitting* (Teng et al., 2023). Proses ini membantu menyederhanakan data tanpa menghilangkan informasi penting, sehingga pemrosesan menjadi lebih cepat dan ringan.

*Pooling layer* mengambil mengambil output dari layer konvolusi sebagai *input*. Proses ini diterapkan pada *feature map* yang sudah melalui tahap konvolusi dan aktivasi. Hasil dari *pooling* berupa *feature map* dengan ukuran yang lebih kecil, karena hanya merangkum informasi penting dari *feature map* sebelumnya. Dengan menggeser filter pada gambar, *pooling* dilakukan untuk menerapkan operasi tertentu. Seperti di jelaskan Gambar 2.6. Beberapa jenis *pooling* yang umum digunakan antara lain *max pooling*, *average pooling*, dan *L2-norm pooling*. Masing-masing memiliki cara tersendiri dalam merangkum informasi, namun tujuannya tetap sama: menyederhanakan data sambil mempertahankan fitur penting.(Sarıgül et al., 2019).



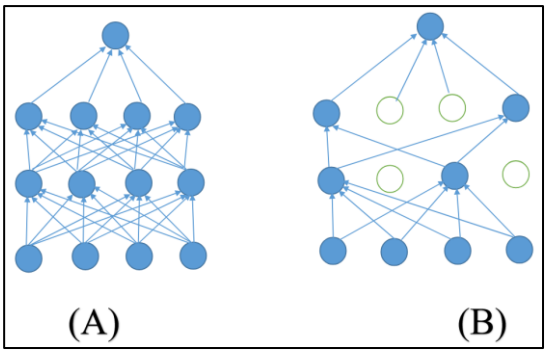

**Gambar 2. 6** *Global Average Pooling*

Pada Gambar 2.6 *Global Average Pooling* (GAP) adalah suatu teknik yang digunakan dalam arsitektur *Convolutional Neural Network* (CNN) di mana rata-rata dari semua nilai piksel di seluruh area *feature map* dihitung untuk setiap *channel*. Hasilnya adalah satu nilai rata-rata untuk setiap *channel*, mengurangi dimensi dari *feature map*. GAP sering dimanfaatkan untuk mengurangi dimensi data dalam arsitektur CNN, membantu mengurangi jumlah parameter dalam model dan mencegah *overfitting*. *GAP* memiliki kelebihan dalam hal ketahanan terhadap pergeseran dan perubahan bentuk pada data gambar, sehingga membuatnya lebih stabil dan mampu menghadapi variasi posisi fitur di dalam *feature map*. (Yenusi et al., 2023).

### 2.4.3 *Dropout*

Pada pelatihan *Convolutional Neural Networks* (CNN) untuk menghindari *overfitting* salah satu tekniknya yaitu dengan *DropOut*. *Overfitting* terjadi ketika jaringan saraf belajar untuk mengingat data pelatihan dengan sangat baik, tetapi tidak mampu menggeneralisasikan pengetahuan ini ke data yang belum pernah dilihat sebelumnya.

*DropOut* dilakukan dengan cara menghapus beberapa neuron secara acak saat pelatihan dari layer atau *hidden layer* sehingga jumlah parameter dan kompleksitasnya berkurang, sehingga mempercepat proses pelatihan. Dalam metode ini, aktivasi beberapa *neuron* yang dipilih secara acak untuk dinonaktifkan (diatur menjadi nol) selama proses pelatihan. *Neuron* yang dipilih berubah di setiap iterasi pelatihan. Teknik ini membantu membuat proses pelatihan menjadi lebih stabil dan mengurangi risiko *overfitting*. Istilah "*DropOut*" merupakan proses pemutusan sementara *neuron* (tersembunyi dan terlihat) dalam *neural network,* selama pelatihan berlangsung dan dipilih secara acak.




**Gambar 2. 7** Mekanisme *DropOut*
Sumber (Salehin & Kang, 2023)

Pada Gambar 2.7, menunjukkan perbedaan antara jaringan saraf standar (A) dan jaringan saraf yang menggunakan teknik *dropout* (B). Pada jaringan standar (A) menunjukkan semua neuron aktif dan saling terhubung penuh. Sementara itu, pada jaringan dengan *dropout* (B), beberapa neuron secara acak dinonaktifkan (ditandai dengan lingkaran kosong hijau) selama proses pelatihan untuk mencegah *overfitting* (Salehin & Kang, 2023).

### 2.4.4 *Batch Normalization*

*Batch Normalization* merupakan sebuah teknik yang digunakan untuk menormalkan nilai *input* pada setiap layer dalam jaringan saraf. Proses normalisasi ini bertujuan untuk

menyesuaikan skala nilai agar tetap berada dalam rentang yang konsisten. Teknik ini dapat mempercepat proses komputasi dan konvergensi saat pelatihan model.

Sehingga *batch normalization* dapat membantu mengurangi pergeseran distribusi data antar *layer*, sehingga model menjadi lebih efisien dan stabil dalam proses belajar. *Batch normalization* membantu menstabilkan dan mempercepat proses pelatihan jaringan saraf dengan menormalkan *output* dari setiap lapisan berdasarkan data dalam *mini-batch*. Proses ini membantu menjaga agar nilai output memiliki rata-rata yang mendekati nol dan varian yang tetap konsisten. Dengan demikian, batch normalization dibuat untuk mengatasi variabilitas dalam distribusi *input* antar layer, yang sering memperlambat konvergensi, dan memungkinkan penggunaan *learning rate* yang lebih tinggi untuk mempercepat proses pembelajaran (Garbin et al., 2020).

### 2.4.5 *Optimizer*

Optimasi merupakan bagian penting dari bagaimana proses jaringan syaraf belajar. Ini mengacu pada proses yang meminimalkan kesalahan (*loss*). Poses ini dilakukan dengan cara menyesuaikan bobot-bobot dalam jaringan secara bertahap, hingga model menemukan kombinasi yang menghasilkan kesalahan sekecil mungkin. Dengan pemilihan optimati yang tepat akan membantu model dapat belajar lebih baik.

Sebagai contoh, *TensorFlow* menyediakan berbagai pilihan optimasi untuk pengoptimalan model, seperti *Adam*, *RMSprop.* dan *Stochastic Gradient Descent* (SGD), serta lainnya. Setiap jenis optimasi tersebut memiliki karakteristik yang berbeda-beda. Dalam penggunaan optimasi tersebut dapat disesuaikan dengan model yang digunakan dan kecepatan model dalam mempelajari data yang baru.

*Optimasi* seperti *Adam* (*Adaptive Moment Estimation*) adalah metode optimasi adaptif yang menggabungkan konsep momentum dan *RMSProp* dengan ini menghitung gradien adaptif untuk setiap parameter dan mempertahankan momentum dari gradien sebelumnya. *RMSProp* (*Root Mean Square Propagation*) merupakan optimasi yang mengurangi ukuran langkah yang diambil untuk setiap parameter berdasarkan rata-rata kuadrat dari gradien sebelumnya. *Stochastic Gradient Descent* (SGD) adalah optimasi dengan memperbarui parameter dengan mengambil langkah berdasarkan gradien yang dihitung dari subset kecil (*batch*) data training pada setiap iterasi.

### 2.4.6 *Hyperparameter*

*Hyperparameter* merupakan salah satu jenis parameter yang perlu ditentukan sebelum melakukan proses training. Variabel-variabel dalam *hyperparameter* menentukan cara jaringan tersebut dilatih. Pengaturan *hyperparameter* bertujuan untuk mengoptimalkan bobot dan bias. Hyperparameter seperti *learning rate*, jumlah *epoch*, ukuran *batch*, dan arsitektur jaringan seperti jumlah lapisan dan neuron. Dalam pemilihan *hyperparameter* harus tepat karena sangat penting dapat memengaruhi kinerja akhir model secara signifikan. Beberapa contoh *hyperparameter* adalah sebagai berikut.

*Learning rate* adalah faktor yang mengatur seberapa besar model mengubah bobot dan biasnya selama *training* saat setiap kali proses pembaruan terjadi. Jika nilai pembelajaran (*learning rate*) terlalu tinggi dapat menyebabkan model tidak stabil,

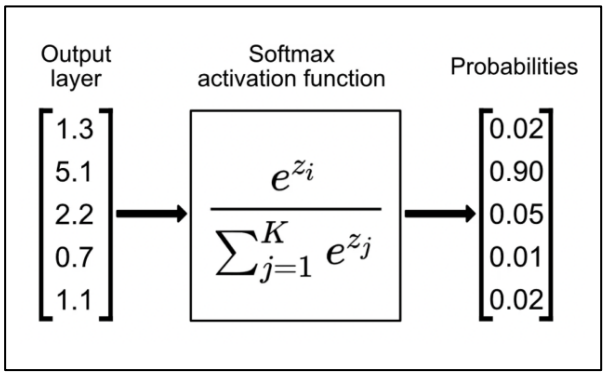
sedangkan tingkat pembelajaran yang terlalu rendah dapat membuat proses konvergensi menjadi sangat lambat. *Hyperparameter* ini mempengaruhi kecepatan konvergensi dan stabilitas *training*.

*Epoch* adalah jumlah iterasi yang dilakukan saat melatih model pada seluruh dataset *training*. Satu *epoch* berarti model telah melihat dan memproses seluruh dataset *training* satu kali. Jumlah *epoch* yang terlalu sedikit bisa menyebabkan *underfitting*, sedangkan jumlah *epoch* yang terlalu banyak bisa mengakibatkan *overvitting* (Xu et al., 2023). Dengan menentukan jumlah epoch yang digunakan akan mempengaruhi model dalam belajar.

### 2.4.7 *Softmax*

*Softmax classifier* adalah variasi dari *logistic regression* yang dirancang khusus untuk menangani klasifikasi dengan jumlah kelas lebih dari dua. Jika *logistic regression* biasanya digunakan untuk tugas klasifikasi biner yaitu memilih antara dua kemungkinan, sedangkan *softmax* memungkinkan model untuk memilih satu kelas dari banyak pilihan yang tersedia, dengan menghitung peluang masing-masing kelas secara proporsional (Li et al., 2020)

Fungsi *softmax* digunakan dalam menghitung distribusi probabilitas terhadap suatu vektor angka. Tujuan dari fungsi *softmax* adalah menghasilkan *output* dengan nilai-nilai dalam rentang 0 sampai 1, dengan total seluruh probabilitasnya berjumlah 1. *Softmax* dapat digunakan untuk model *multi classification*. Nilai kelas dihitung dengan menggunakan fungsi *softmax* pada Gambar 2.8, yang ditunjukkan oleh Persamaan 2.1.




**Gambar 2. 8** Fungsi *Softmax*

$$S(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{.....(2.1)}$$

Keterangan :
S : vektor dengan elemen bernilai di antara 0 hingga 1
e : konstanta matematika nilai mendekati 2,71828

## 2.5 *Transfer Learning*

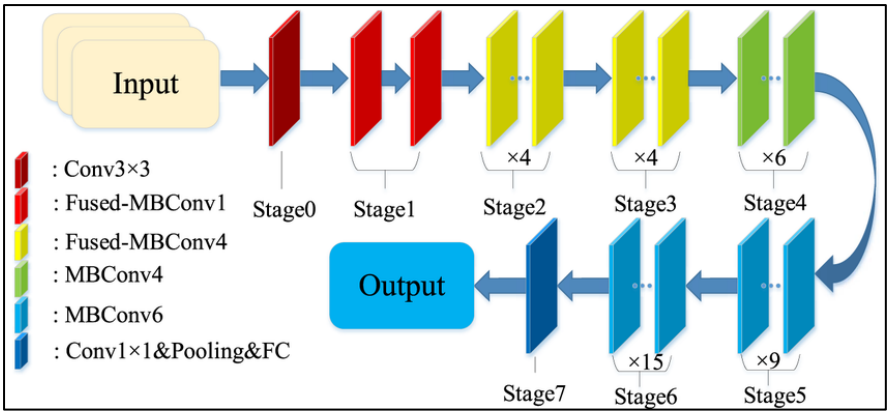
*Transfer learning* adalah metode yang memanfaatkan model sebelumnya yang telah dilatih dalam menyelesaikan suatu tugas, kemudian digunakan kembali untuk menyelesaikan tugas yang berbeda. Metode dengan *transfer learning* membatu pelatihan

data dengan jumlah data pelatihan yang terbatas, namun tetap ingin mencapai hasil akurasi yang tinggi (Salasta et al., 2024). *Transfer learning* ialah salah satu proses pada CNN dengan menggunakan model yang *pre- trained* atau sebelumnya telah melalui proses pelatihan oleh peneliti atau pihak lain dengan menggunakan dataset berukuran sangat besar, seperti *ImageNet*. Kemudian model akan digunakan kembali dalam menyelesaikan tugas baru, tanpa perlu melatih model dari awal lagi.

*ImageNet* adalah salah satu kumpulan dataset gambar berskala besar yang sering dimanfaatkan dalam pengembangan dan evaluasi model *deep learning* di bidang pengolahan citra. Model yang telah dilatih (*pre-trained*) dengan menggunakan dataset *ImageNet* umumnya memiliki kemampuan ekstraksi fitur yang lebih optimal, sehingga dapat digunakan kembali dalam berbagai tugas klasifikasi gambar dengan performa yang baik.

## 2.6 *EfficientNetV2*

Salah satu arsitektur *CNN* dalam bidang deep learning yang digunakan khusus untuk pemrosesan dan pengenalan gambar adalah *EfficientNet*. Dalam penelitian ini, digunakan arsitektur *EfficientNetV2S*, salah satu model *CNN* yang memiliki performa tinggi dalam klasifikasi gambar. *EfficientNetV2S* merupakan salah satu varian dari *EfficientNet* yang meemiliki kecepatan *training* lebih tinggi serta efisiensi parameter yang lebih optimal dibandingkan model sebelumnya, yaitu *EfficientNetV1*. Model ini dikembangkan menggunakan *training-aware neural architecture search* dan *scaling* (Sidik et al., 2023).




**Gambar 2. 9** Arsitektur *EfficeintNetV2S*
Sumber (Huang et al., 2023)

Gambar 2.9 merupakan arsitektur *EfficientNetV2S* dirancang untuk mencapai performa tinggi sekaligus menjaga efisiensi dalam penggunaan komputasi. Model ini terdiri dari beberapa blok utama, dimulai dengan *Batch Normalization* dan lapisan konvolusi awal 3x3 yang bertugas menangkap fitur-fitur dasar dari citra. Setelah itu menggunakan blok *Fused-MBConv* dan *MBConv* secara bertahap, yang merupakan jenis konvolusi efisien untuk memahami fitur-fitur dengan tingkat kompleksitas tinggi dengan beban komputasi yang lebih ringan. *Fused-MBConv* menggabungkan langkah konvolusi dan ekspansi dalam satu tahap untuk efisiensi, sedangkan *MBConv* menggunakan struktur *inverted residual* dengan *Squeeze-and-Excitation block* untuk memberi penekanan lebih pada fitur-fitur yang dianggap penting oleh model. Kemudian fitur yang dihasilkan dari semua blok ini kemudian

diringkas melalui *Global Average Pooling*, lalu dilanjutkan ke lapisan *dense* dan *dropout* untuk proses klasifikasi, dan akhirnya menghasilkan output melalui *fully connected layer* yang disesuaikan dengan jumlah kelas yang ingin diprediksi. Output akhir dihasilkan melalui *fully connected layer* yang disesuaikan dengan jumlah kelas. Secara keseluruhan, gambar ini merangkum alur pemrosesan fitur dalam *EfficientNetV2S* yang mendukung akurasi tinggi dan kecepatan inferensi.

## 2.7 *Evaluasi Matrix*

Penerapan Matrik evaluasi pada *Deep Learning* digunakan untuk mencapai pengklasifikasi yang optimal. *Evaluation Matrix* digunakan dalam prosedur klasifikasi data melalui dua tahap utama yaitu pelatihan dan pengujian. Pada tahap pelatihan model, matrik evaluasi digunakan untuk mengoptimalkan algoritma klasifikasi. *Confusion Matrix* merupakan sebuah metode evaluasi yang digunakan untuk menilai performa dari kinerja model. Terdapat memiliki 4 nilai *Confusion matrix* yang berbeda yaitu *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), *True Negative* (TN). Pengukuran performa model pada *confusion matrix* membutuhkan metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *f1-score* seperti tabel 2.1 berikut.

**Tabel 2. 1** *Confusion Matrix*

| | | ***Actual Values (Label Asil)*** | |
|---|---|---|---|
| | | ***True*** | ***False*** |
| ***Predeicted Values*** | ***True*** | *True Positive (TP)* | *False Positive (FP)* |
| | ***False*** | *False Negative (FN)* | *True Negative (TN)* |

Keterangan dari keempat nilai tersebut adalah sebagai berikut:

a. *True Positive* (TP) adalah data yang sebenarnya positif (*Actual Value = True*) dan diprediksi dengan benar sebagai positif oleh model (*Predicted Value = True*).
b. *True Negative* (TN) adalah data yang sebenarnya negatif (*Actual Value = False*) dan diprediksi dengan benar sebagai negatif oleh model (*Predicted Value= False*).
c. *False Positive* (FP) adalah data yang sebenarnya negatif (*Actual Value = False*), tetapi diprediksi salah sebagai positif oleh model (*Predicted Value = True*).
d. *False Negative* (FN) adalah data yang sebenarnya positif (*Actual Value = True*), tetapi diprediksi salah sebagai negatif oleh model (*Predicted Value = False*).

### 3.4.1 Akurasi

Akurasi adalah perbandingan antara jumlah prediksi yang benar dari total prediksi yang dibuat oleh model. Akurasi menjadi ukuran umum yang menunjukkan seberapa sering model membuat prediksi yang benar.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \quad \text{.....(2.2)}$$

### 3.4.2 Presisi

Presisi merupakan rasio antara jumlah prediksi positif yang benar dari semua prediksi positif yang dibuat oleh model. Digunakan untuk menilai ketepatan model dalam

membuat prediksi positif.

$$Presisi = \frac{True\ Positives\ (\mathrm{TP})}{True\ Positives\ (\mathrm{TP})\ +\ False\ Positives\ (\mathrm{FP})} \quad \text{.....(2.3)}$$

### 3.4.3 *Recall*

Recall ialah proporsi prediksi positif yang benar dari semua kasus positif sebenarnya dalam data. Ini mengukur kemampuan model untuk menemukan semua kasus positif.

$$Recall = \frac{True\ Positives\ (\mathrm{TP})}{True\ Positives\ (\mathrm{TP})\ +\ False\ Negative\ (\mathrm{FN})} \quad \text{.....(2.4)}$$

### 3.4.4 *F1 Score*

*F1 Score* merupakan rata-rata harmonik dari nilai presisi dan *recall*. *F1 Score* memberikan keseimbangan antara keduanya dan bermanfaat ketika diperlukan kompromi antara presisi dan recall.

$$F1\ Score = 2 \times \frac{\mathrm{Presisi} \times Recall}{\mathrm{Presisi} + Recall} \quad \text{.....(2.5)}$$

## 2.8 Penelitian Sebelumnya

Berdasarkan penelusuran yang telah dilakukan terhadap judul tugas akhir dari berbagai sumber yang akan dipaparkan berikut ini, bahwa penelitian yang berjudul "Klasifikasi Ular Berbisa Dan Tidak Berbisa Menggunakan Deep Learning Dengan Arsitektur *EfficientnetV2S*" dapat dipertanggungjawabkan keasliannya, adapun penelitian penelitian sebelumnya sebagai berikut:

**Tabel 2. 2** *State Of The Art*

| No | Judul Penelitian | Metode | Hasil Penelitian |
|---|---|---|---|
| 1. | Evaluasi Ensemble Stacking Arsitektur EfficientNetB3 dan EfficientNetV2S (Studi Kasus Klasifikasi Penyakit Alzheimer)<br><br>(Kasus et al., 2023) | Model EfficientNetB3 dan EfficientNetV2S, Keduanya menggunakan algoritma optimisasi Adam dengan learning rate sebesar 0,0001, selama 50 epoch, dan batch size sebanyak 32. | Model *EfficientNetV2S* mampu memberikan performa yang lebih baik. *EfficientNetV2S* 95,45% dengan nilai *loss* 0,2275 sedangkan *EfficientNetB3* 92,54% dengan nilai *loss* 0,3726. Arsitektur lebih akurat dalam mengenali pola dari data, tetapi juga lebih baik dalam menghindari *overfitting*.<br><br>Perbedaan : Terdapat pada objek dan dataset yang digunakan. |
| 2. | Analisis Klasifikasi Citra Penokohan Topeng Bali Menggunakan Model EfficientNetV2 dan Xception<br><br>(Yuniari et al., 2024) | Model yang diterapkan adalah EfficientNetV2 dan Xception. Menggunakan optimizer Adam dan binary crossentropy | Hasil pengujian model EfficientNetV2 berhasil mencapai akurasi 99%, lebih tinggi dibandingkan Xception yang mencapai 97%. Membuktikan bahwa efektif digunakan untuk mengidentifikasi berbagai jenis penokohan topeng Bali.<br><br>Perbedaan : Terdapat pada objek dan dataset yang digunakan. |

**Tabel 2. 3** *State Of The Art* (Lanjutan)

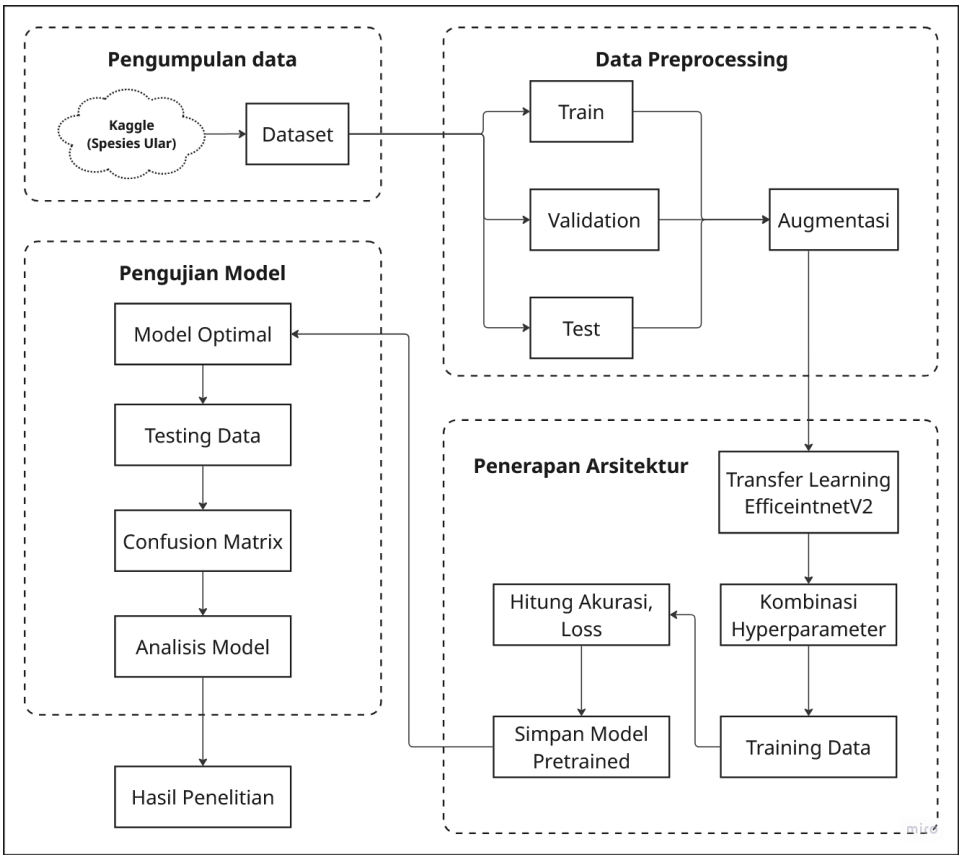
| | | | |
|---|---|---|---|
| 3. | *EfficientNets* and Vision Transformers for Snake Species Identification Using Image and Location Information<br><br>(Bloch & Friedrich, 2021) | EfficientNetB4, ViT-L (Vision Transformer Large), dan ViT-B (Vision Transformer Base), optimizer Adam | Model EfficientNet-B4 (S1) dengan F1-score antara 75,28% - 78,75%. ViT-L (S5) memiliki performa lebih rendah dibandingkan EfficientNet, dengan skor antara 63,20% - 78,75%. Ketika kedua model digabungkan dalam ensemble (S7), performa meningkat signifikan dengan F1-score mencapai 82,88%.<br><br>Perbedaan : Jumlah dataset |
| 4. | Analisis Performa Metode Convolutional Neural Network Dengan Arsitektur ConvNeXt Dalam Klasifikasi Spesies Ular Berbisa Dan Tidak Berbisa Di Indonesia<br><br>(Qisthan, 2023) | ConvNeXt, optimizer adam, learning rate 0,0001. | Arsitektur ConvNeXt. Model ConvNeXtTiny dan ConvNeXtSmall memiliki akurasi dan F1-score yang sama, yaitu 0,73. ConvNeXtBase dan ConvNeXtLarge, dengan F1-score masing-masing 0,85 dan 0,84. Model terbaik adalah ConvNeXtXLarge dengan skor tertinggi di semua metrik sebesar 0,92.<br><br>Perbedaan : Jumlah dataset dan arsitektur model yang digunakan. |
| 5. | Data augmentation and transfer learning efficientnetv2-s on rice leaf disease classification<br><br>(Nggego et al., 2025) | Arsitektur yang digunakan adalah EfficientNetV2-S. Optimizer Adam dengan learning rate 0,00001, batch size 32, dan jumlah epoch sebanyak 25. | EfficientNetV2-S secara performa klasifikasi penyakit daun padi. Model yang dilatih menggunakan teknik augmentasi mencapai akurasi sebesar 95% ± 2, yang merupakan peningkatan 29% ± 2 dari akurasi model yang dilatih tanpa metode augmentasi.<br><br>Perbedaan : Terdapat pada objek, learning rate. |
| 6. | Classication and Interpretation of Histopathology Images: Leveraging Ensemble of EcientNetV1 and EcientNetV2 Models<br><br>(Kalati et al., 2025) | Model CNN EfficientNetV1 dan EfficientNetV2. Optimizer Adam dengan parameter default (beta1=0.9, beta2=0.999, epsilon=1e-07) | Model EfficientNet V2-M (setara B2) mencatat akurasi tertinggi 99,15% dengan efisiensi pelatihan yang unggul. V1-B4 dan V2-L juga sangat akurat 99,07% dan performanya setara. V2-S sedikit di bawah dengan akurasi 98,48%, tetapi jauh lebih cepat dan hemat sumber daya. Secara keseluruhan, EfficientNetV2 lebih efisien saat pelatihan dibandingkan EfficientNetV1.<br><br>Perbedaan : Terdapat pada objek dan paramater. |
| 7. | Penerapan Arsitektur VGG-19 Untuk Menangani Overfitting Pada Identifikasi Ular Berbisa Menggunakan Metode Convolutional Neural Network (Indrawani, 2020) | VGG-19, Optimizer Adam, epoch sebanyak 15 | VGG-19 pre-trained dengan 15 epoch, menghasilkan akurasi tertinggi sebesar 89% pada data test dengan nilai loss 0,2 yang menunjukkan tingkat kesalahan (loss) yang rendah, dengan nilai sensitivity sebesar 87%, dan specificity sebesar 92%<br>Perbedaan : Terdapat pada dataset dan paramater. |

**Tabel 2. 4** *State Of The Art* (Lanjutan)

| | | | |
|---|---|---|---|
| 8. | Deep Learning-Based Snake Species Identification for Enhanced Snakebite Management<br><br>(Iguernane et al., 2025) | VGG-16, VGG-19, EfficientNet B0 menggunakan Adam dengan learning rate sebesar 1×10^-4. 224×224×3 sesuai input CNN. | Hasil pengujian menunjukkan bahwa VGG-16 mencapai sekitar 84% akurasi, VGG-19 sekitar 80%, dan EfficientNet B0 mencapai 92,23%. EfficientNet B0 menunjukkan performa terbaik dalam mengklasifikasi spesies ular secara akurat dan stabil.<br><br>Perbedaan : Jumlah dataset dan arsitektur model yang digunakan. |
| 9. | Snake species classification using deep learning techniques<br><br>(Ahmed et al., 2024) | Arsitektur VGG16, DenseNet121, ResNet50, dan MobileNetV2. Masing-masing arsitektur diuji pada jumlah kelas yang berbeda (5, 10, 16, 20, 22, dan 45 kelas). | Hasil ini menegaskan bahwa untuk dataset dengan 5 kelas, VGG16 paling unggul dalam hal akurasi dan stabilitas. VGG16 dapat akurasi tertinggi sebesar 99,32%, diikuti oleh DenseNet121 dengan akurasi 98,92%, serta MobileNetV2 yang mencapai 95,09%. Sementara itu, ResNet50 mencatatkan performa terendah dengan akurasi hanya 53,79%.<br><br>Perbedaan : Jumlah dataset dan arsitektur model yang digunakan. |
| 10. | Venomify: Automated Classification of Venomous and Non-Venomous Snake Species Using Deep Learning<br><br>(Anwarul & Tanwar, 2025) | Modified DenseNet169, Optimizer: Adam dengan lr 0,007, split data 80:20 | DenseNet169 mampu mengklasifikasikan ular berbisa dan tidak berbisa dengan akurasi tinggi sebesar 97,35% proses training 10 epoch. 97,35% accuracy, precision 97,35%, recall 97,37%, dan F1-score of 97,35%.<br><br>Perbedaan : Jumlah dataset dan arsitektur model yang digunakan |

# BAB III
# METODOLOGI PENELITIAN

Metodologi penelitian ini melibatkan serangkaian tahapan yang terstruktur, dimulai dari pengumpulan data hingga analisis hasil klasifikasi ular. Penelitian ini menerapkan metodologi kuantitatif dengan jenis eksperimental. Metodologi penelitian ini diterapkan untuk memodelkan penerapan arsitektur *EfficientNetV2S.*




**Gambar 3. 1** Tahapan Metodologi Penelitian

Tahapan penelitian yang diilustrasikan pada Gambar 3.1 diatas dimulai pengumpulan dataset citra ular dari sumber *kaggle* yang sesuai dengan kebutuhan penelitian dan melakukan penambahan data dengan *scrapping* melalui *website GBIF*. Setelah dataset citra dikumpulkan, dilanjutkan tahap *pre-processing* dengan *split dataset* dan dilakukan *augmentasi* data. Data yang telah melalui tahap *pre-processing* akan dilakukan proses penerapan arsitektur dengan *EfficientNetV2S*. Hasil dari model *pretrained EfficientNetV2S* akan digunakan untuk mengklasifikasikan spesies ular. Pada tahap akhir akan dilakukan pengujian dan didapatkan hasil dari pengujian tersebut. Penjabaran lanjut mengenai tahapan penelitian dibahas pada subbab-subbab setelah ini.

## 3.1 Pengumpulan Data

Pada penelitian ini menggunakan dataset yang bersumber dari *website Kaggle* yang bernama Dataset *Snakes species* "*https://www.kaggle.com/datasets/goelyash/165-different-snakes-species"* dengan nama author Yash Goel yang di update 3 tahun yang lalu dan

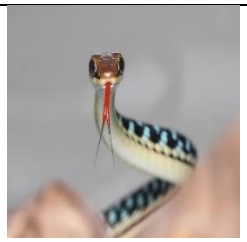
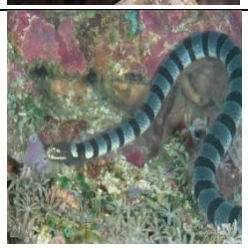
melakukan *scrapping* melalui *website GBIF*.

Pada proses ini data yang diambil yaitu 20 jenis ular yang beberapa tersebar di wilayah tropis, kemudian diklasifikasikan sebagai spesies ular tidak berbisa (*non-venomous*) dan ular berbisa (*venomous*). Tabel dibawah merupakan hasil dataset ular pada penelitian ini.

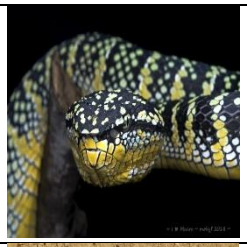
**Tabel 3. 1** Data Spesies Ular

| No | Nama Ilmiah Spesies | Gambar Ular | Famili Ular | Berbisa/Tidak |
|---|---|---|---|---|
| 1 | *Ahaetulla prasina* |  | *Ahaetulla* | Tidak |
| 2 | *Bitis gabonica* |  | *Bitis* | Berbisa |
| 3 | *Boiga irregularis* |  | *Boiga* | Berbisa |
| 4 | *Boiga kraepelini* |  | *Boiga* | Tidak |
| 5 | *Bungarus multicinctus* |  | *Bungarus* | Berbisa |
| 6 | *Chrysopelea ornata* |  | *Chrysopelea* | Tidak |

**Tabel 3. 2** Data Spesies Ular (lanjutan)

| No | Nama Ilmiah Spesies | Gambar Ular | Famili Ular | Berbisa/Tidak |
|---|---|---|---|---|
| 7 | *Daboia russelii* |  | *Daboia* | Berbisa |
| 8 | *Dendrelaphis pictus* |  | *Dendrelaphis* | Tidak |
| 9 | *Gonyosoma oxycephalum* |  | *Gonyosoma* | Tidak |
| 10 | *Laticauda colubrina* |  | *Laticauda* | Berbisa |
| 11 | *Lycodon capucinus* |  | *Lycodon* | Tidak |
| 12 | *Morelia viridis* |  | *Morelia* | Tidak |
| 13 | *Ophiophagus hannah* |  | *Ophiophagus* | Berbisa |

**Tabel 3. 3** Spesies Ular (lanjutan)

| No | Nama Ilmiah Spesies | Gambar Ular | Famili Ular | Berbisa/Tidak |
|---|---|---|---|---|
| 14 | *Protobothrops mucrosquamatus* |  | *Protobothrops* | Berbisa |
| 15 | *Psammodynastes pulverulentus* |  | *Psammodynastes* | Tidak |
| 16 | *Python molurus* |  | *Python* | Tidak |
| 17 | *Rhabdophis subminiatus* |  | *Rhabdophis* | Berbisa |
| 18 | *Tropidolaemus subannulatus* |  | *Tropidolaemus* | Berbisa |
| 19 | *Tropidolaemus wagleri* |  | *Tropidolaemus* | Berbisa |
| 20 | *Xenochrophis piscator* |  | *Xenochrophis* | Tidak |

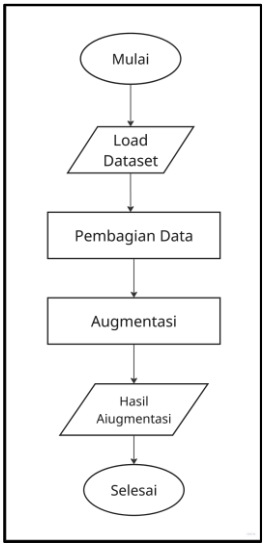
Hasil dari penentuan berbagai spesies ular yang akan dijadikan objek model klasifikasi gambar, terdapat pada Tabel 3.1 hingga Tabel 3.3. Pemilihan ini didasarkan pada

keberadaan spesies tersebut di lingkungan sekitar yaitu spesies ular yang memiliki habitat mencakup wilayah tropis khususnya 18 spesies terdapat di Indonesia yang memiliki jenis berbisa dan tidak berbisa.

Tabel tersebut mencakup informasi terkait nama spesies, gambar, *famili* ular, dan berbisa atau tidaknya ular tersebut. Pemberian kerangka tabel ini diharapkan dapat memberikan pandangan yang sistematik dan terstruktur terhadap kelompok spesies yang dipilih.

## 3.2 Data *Preprocessing*

Setelah tahapan pengumpulan data dan didapatkan 20 spesies ular yang akan diklasifikasi, langkah selanjutnya adalah pra-pemrosesan Data. Tujuan dari tahap ini adalah mempersiapkan data agar lebih optimal untuk proses pemodelan, dengan menerapkan berbagai teknik pemrosesan data.




**Gambar 3. 2** *Flowcart Preprocessing*

### 3.2.1 Pembagian Data

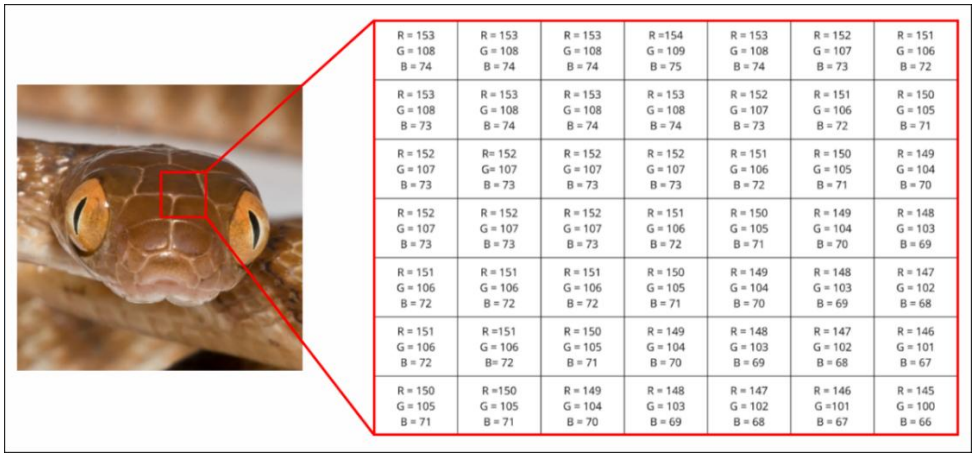
Dalam proses pembagian data atau sering disebut *split dataset*, proses tersebut menghasilkan data *training, validasi*, dan *testing*. Setelah dataset dibagi menjadi 3 dengan perbandingan data *training* sebesar 80%, agar model memahami dan dapat mempelajari pola yang lebih kompleks dan meningkatkan kemampuan generalisasi. Selanjutnya, dibagi 10% untuk data validasi yang digunakan untuk menilai kinerja model terhadap saat proses pelatihan. Sementara data testing sebesar 10% digunakan untuk mengevaluasi performa model dengan data baru, yang merepresentasikan kondisi dunia nyata. Hasil pembagian dari jumlah dataset yang belum di *augmentasi* pada penelitian ini dapat dilihat pada Tabel 3.4.

**Tabel 3. 4** Pembagian Data

| No | Pembagian Data | Jumlah |
|---|---|---|
| 1 | *Training* | 3680 |
| 2 | *Validation* | 480 |
| 3 | *Testing* | 480 |
| Total | | 4640 |

### 3.2.2 *Augmentasi*

*Augmentasi* data adalah proses yang bertujuan memperbanyak variasi data saat pelatihan agar menciptakan variasi baru dari dataset yang dimiliki. *Augmentasi* dilakukan dengan cara memodifikasi gambar asli tanpa mengubah label.




**Gambar 3. 3** Representasi Citra RGB

Gambar 3.3 menunjukkan citra ular yang digunakan sebagai sampel perhitungan dari *matrix* 384x384 piksel menjadi 7x7 piksel. Citra ular tersebut akan direpresentasikan sebagai sebuah matriks tiga dimensi yang berisi nilai-nilai intensitas warna untuk *channel* RGB (*Red, Green, Blue*). Setiap piksel dalam gambar memiliki nilai antara 0 hingga 255 pada ketiga *channel* tersebut. Kemudian gambar perlu melalui tahapan pra-pemrosesan (*preprocessing*) agar sesuai dengan format *input* pada model.

Pra-pemrosesan ini dilakukan menggunakan fungsi bawaan dari *TensorFlow*, yaitu "*preprocess_input"*, yang secara khusus disesuaikan untuk arsitektur *EfficientNetV2S*. Dengan fungsi citra akan dinormalisasi nilai piksel dari skala [0, 255] menjadi skala [-1, 1]. Tujuan normalisasi ini untuk menjaga kestabilan numerik saat data melewati layer. Oleh karena itu, *preprocess_input* melakukan normalisasi nilai piksel dari gambar menggunakan operasi matematis berikut:

$$\mathrm{x} = \frac{x}{127{,}5} - 1$$

Keterangan : x adalah nilai pixel (antara 0-255) :

Kemudian diambil dari hasil Rescale dengan menggunakan nilai lapisan RED sebagai sampel, seperti yang dijelaskan pada Gambar 3.3. Selanjutnya hasil perhitungan Rescale dapat dilihat pada Tabel 3.5 di bawah ini

**Tabel 3. 5** Nilai citra awal untuk proses normalisasi [-1,1]

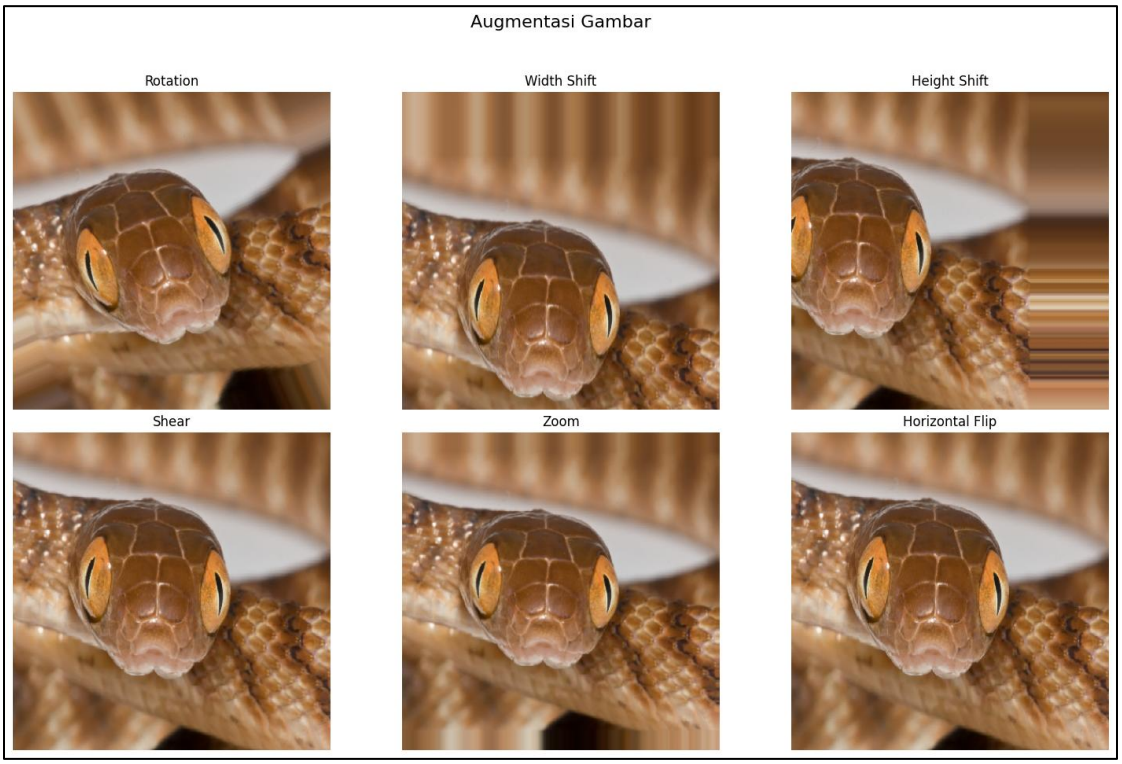
| | | | | | | |
|---|---|---|---|---|---|---|
| 0,20 | 0,20 | 0,20 | 0,21 | 0,20 | 0,19 | 0,18 |
| 0,20 | 0,20 | 0,20 | 0,20 | 0,19 | 0,18 | 0,18 |
| 0,19 | 0,19 | 0,19 | 0,19 | 0,18 | 0,18 | 0,17 |
| 0,19 | 0,19 | 0,19 | 0,18 | 0,18 | 0,17 | 0,16 |
| 0,18 | 0,18 | 0,18 | 0,18 | 0,17 | 0,16 | 0,15 |
| 0,18 | 0,18 | 0,18 | 0,17 | 0,16 | 0,15 | 0,15 |
| 0,18 | 0,18 | 0,17 | 0,16 | 0,15 | 0,15 | 0,14 |

Setelah proses normalisasi, proses citra ular berikutnya tahap *augmentasi* yaitu *rotation range*, *width*, *height*, *shere range*, *zoom range*, *horizontal flip*, terhadap gambar asli untuk menghasilkan versi baru yang berbeda namun masih relevan. Rincian lengkap mengenai proses augmentasi yang digunakan dapat dilihat pada. Pada Tabel 3.6 dapat dilihat detail augmentasi data yang dilakukan:

**Tabel 3. 6** Variasi *Parameter Augmentasi*

| No | Paramerter | Nilai | Keterangan |
|---|---|---|---|
| 1. | *Rotation Range* | 20 | Gambar akan diputar acak dalam rentang -20 hingga +20 derajat. Membantu mengenali objek dari sudut berbeda. |
| 2. | *Zoom* | 0,3 | Gambar akan diperbesar atau diperkecil secara acak hingga 30%. Membantu model mengenali objek dalam skala yang berbeda. |
| 3. | *Width* | 0,2 | Gambar akan digeser horizontal (kanan/kiri) sebanyak 20% dari lebar gambar. |
| 4. | *Height* | 0,2 | Gambar akan digeser vertikal (atas/bawah) sebanyak 20% dari tinggi gambar. |
| 5. | Shear | 0,15 | Distorsi gambar seperti memiringkan atau memotongnya secara diagonal. Berguna untuk meniru perspektif kamera miring |
| 6. | *Horizontal Flip* | True | Gambar dapat dibalik horizontal secara acak. Efektif untuk objek simetris. |

Tujuan dari *augmentasi* data adalah untuk meningkatkan keberagaman variasi dataset latih, sehingga model pembelajaran mesin dapat belajar dengan lebih baik. Selain meningkatkan kemampuan generalisasi model, augmentasi juga efektif dalam mengurangi risiko *overfitting*, karena model akan terbiasa mengenali pola dalam berbagai kondisi. Hasil *augmentasi* seperti pada Gambar 3.4 dibawah ini.



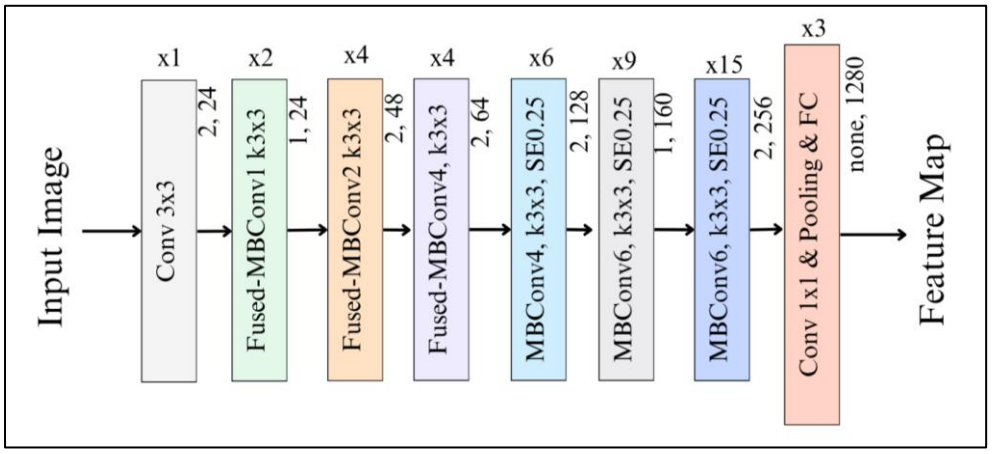

**Gambar 3. 4** Hasil *Augmentasi*

## 3.3 Penerapan Arsitektur

Langkah yang dilakukan setelah proses preprocessing data, augmentasi, dan pembagian data yaitu membangun model CNN dengan arsitektur *EfficientNetV2S*. Berikut merupakan alur arsitektur *EfficientNetV2S* dengan *transfer learning*:

### 3.3.1 Implementasi Arsitektur *EfficientNetV2S*

Penelitian ini menerapkan *Transfer Learning* dengan menggunakan arsitektur *EfficientNetV2S* sebagai *base model. Transfer learning* merupakan metode pemanfaatan pengetahuan dari model yang sebelumnya telah dilatih pada dataset berukuran besar seperti *ImageNet*, untuk digunakan kembali pada dataset yang berbeda namun memiliki karakteristik yang serupa (Umri et al., 2021). *ImageNet* adalah kumpulan data gambar berskala besar yang berisi lebih dari 14 juta gambar dengan masing-masing telah diberi label berdasarkan kategori objek yang ada di dalamnya. Pendekatan yang digunakan dalam *transfer learning* dilakukan dengan pendekatan *fine-tuning*, yaitu proses di mana model *pre-trained* disesuaikan ulang dengan melatih kembali beberapa lapisan menggunakan dataset baru dengan tujuan agar model lebih memahami data baru.

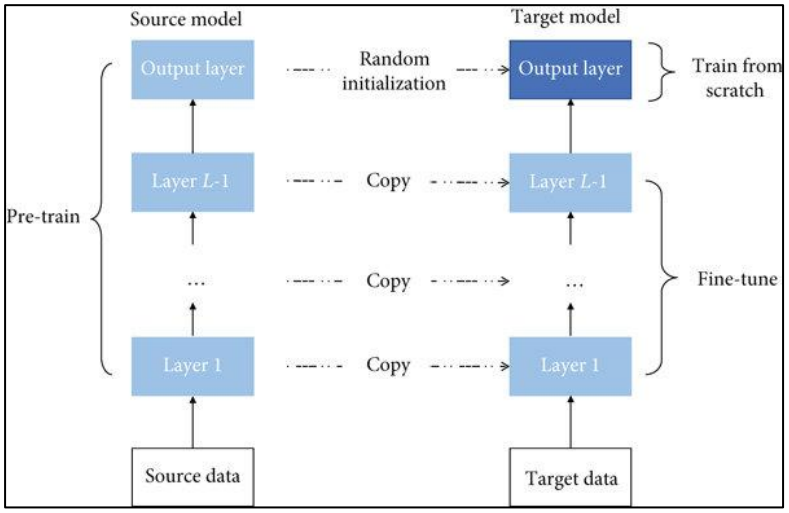



**Gambar 3. 5** Arsitektur *EfficientNetV2S*
Sumber (Aldakhil & Almutairi, 2024)

Pada Gambar 3.5 terlihat bahwa arsitektur *EfficientNetV2S* mengusung beberapa blok baru yang dirancang untuk meningkatkan efisiensi dan performa dibandingkan versi sebelumnya, *EfficientNet*. Beberapa di antaranya adalah blok *MBConv (Mobile Inverted Residual Bottleneck Convolution), Fused-MBConv*, serta *SE(Squeeze-and-Excitation) block.* Kombinasi blok-blok ini membantu model dalam mengekstraksi fitur dengan lebih baik serta mempercepat proses pelatihan. (Sidik et al., 2023).

Proses yang membedakan antara arsitektur *EfficientNetV1* dan *EfficientNetV2S* terletak pada penggunaan blok baru bernama Fused-MBConv pada EfficientNetV2S. Blok MBConv pada *EfficientNetV1* akan disederhakan, di mana proses ekspansi dan *depthwise convolution* digabung menjadi satu tahap menjadi *Fused-MBConv*. Perubahan ini menjadikan EfficientNetV2S lebih efisien dan tidak hanya mengurangi kompleksitas komputasi di tahap awal jaringan, tetapi juga secara signifikan mempercepat proses pelatihan, terutama pada citra resolusi rendah.

### 3.3.2 *Fine-Tuning*




**Gambar 3. 6** Proses *Fine-Tuning*

Gambar 3.6 menjelaskan konsep *fine-tuning* yang digunakan dalam penelitian ini. Dalam kode, model *EfficientNetV2S* dimuat dengan bobot *pre-trained* dari *ImageNet*, lalu menyesuaikannya dengan dataset baru yang lebih spesifik (*target model*). Proses *fine-tuning* dilakukan dengan hanya membuka sebagian lapisan akhir dari model, agar bisa menyesuaikan diri dengan target data, tanpa mengubah lapisan awal yang sudah belajar fitur umum. *Fine tuning* bertujuan untuk menyesuaikan model agar lebih memahami karakteristik unik dari data baru sambil tetap memanfaatkan pengetahuan umum yang telah dipelajari dari dataset besar sebelumnya seperti *ImageNet*.

### 3.3.3 *Input*

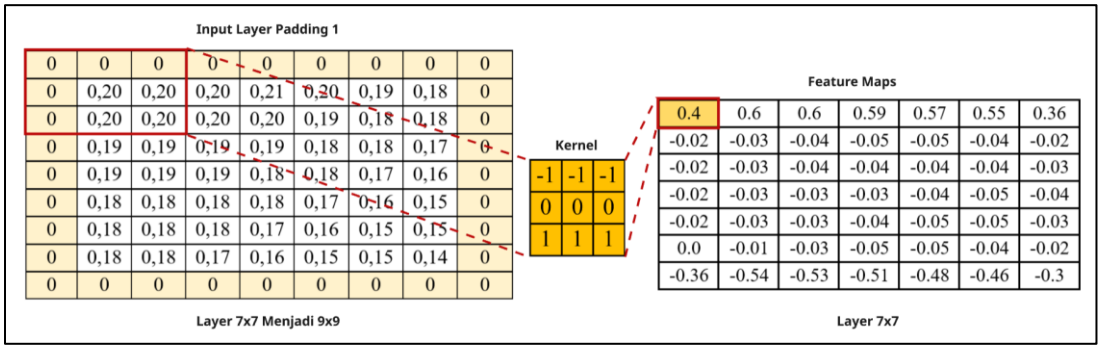
Tahap pertama dari penelitian ini yaitu *Imput* dari sebuah citra. *Input layer* menerima citra yang sudah dilakukan *preprocessing* sebelumnya. Citra dilakukan augmentasi, *resize* sebesar 384 x 384 piksel dengan 3 saluran atau channel *red*, *green*, *blue* (*RGB*), dan normalisasi *rescale* ini mengubah skala nilai piksel dari rentang [0, 255] menjadi [-1, 1]. Artinya, setiap nilai piksel yang semula merupakan angka antara 0 (hitam) hingga 255 (putih), akan disesuaikan ke dalam skala yang lebih kecil dan seimbang di sekitar nol. Proses ini penting karena model *EfficientNetV2S* dilatih pada data citra dengan skala piksel seperti ini, sehingga menjaga konsistensi dan meningkatkan akurasi prediksi saat digunakan pada data baru. Setelah memasuki input layer, citra inputan akan diolah pada layer konvolusi, seperti pada Tabel 3.7.

**Tabel 3. 7** Hasil normalisasi

| | | | | | | |
|---|---|---|---|---|---|---|
| 0,20 | 0,20 | 0,20 | 0,21 | 0,20 | 0,19 | 0,18 |
| 0,20 | 0,20 | 0,20 | 0,20 | 0,19 | 0,18 | 0,18 |
| 0,19 | 0,19 | 0,19 | 0,19 | 0,18 | 0,18 | 0,17 |
| 0,19 | 0,19 | 0,19 | 0,18 | 0,18 | 0,17 | 0,16 |
| 0,18 | 0,18 | 0,18 | 0,18 | 0,17 | 0,16 | 0,15 |
| 0,18 | 0,18 | 0,18 | 0,17 | 0,16 | 0,15 | 0,15 |
| 0,18 | 0,18 | 0,17 | 0,16 | 0,15 | 0,15 | 0,14 |

### 3.3.4 *Convolution Layer*

Proses konvolusi merupakan sebuah proses yang dilakukan dengan menggabungkan dua digit angka sehingga menghasilkan digit angka baru. Lapisan konvolusi terdiri dari serangkaian filter konvolusi atau disebut kernel.




**Gambar 3. 7** Proses Konvolusi

Berdasarkan contoh proses konvolusi pada Gambar 3.7, terdapat citra yang berukuran 5 x 5 yang kemudian dilakukan penambahan *padding* sebanyak 1 (yang ditulis sebagai 0 pada gambar). Sehingga, ukuran input citra menjadi 7 x 7 yang akan menghasilkan citra (*feature map*) berukuran 5 x 5. Kemudian hasil konvolusi di simpan dalam matrik yang baru, dengan proses perhitungan sebagai berikut :

**Tabel 3. 8** Perhitungan Konvolusi

| No | Perhitungan Konvolusi | Hasil |
|---|---|---|
| 1. | (0*−1)+(0*−1)+(0*−1)+(0*0)+(0,2*0)+(0,2*0)+(0*1)+(0,2*1)+(0,2*1) | 0,4 |
| 2. | (0*−1)+(0*−1)+(0*−1)+(0,2*0)+(0,2*0)+(0,2*0)+(0,2*1)+(0,2*1)+(0,2*1) | 0,6 |
| 3. | (0*−1)+(0*−1)+(0*−1)+(0,2*0)+(0,2*0)+(0,21*0)+(0,2*1)+(0,2*1)+(0,2*1) | 0,6 |
| ... | ... | ... |
| 49 | (0,15*−1)+(0,15*−1)+(0*−1)+(0,15*0)+(0,14*0)+(0*0)+(0*1)+(0*1)+(0*1) | -0,3 |

Langkah pertama pada peta aktivasi dimulai dari tanda kotak kuning pada Gambar 3.7 dihitung dengan cara mengonvolusi filter ke bagian gambar input yang juga ditandai kuning. Sehingga proses dapat dilihat pada Tabel 3.8. Proses konvolusi ini kemudian diulangi ke seluruh area gambar input hingga menghasilkan peta aktivasi secara menyeluruh. Pada peta aktivasi ditunjukkan bagaimana cara setiap filter merespon bagian-bagian kecil dari gambar input. Untuk setiap filter akan menghasilkan satu peta aktivasi, dan seluruh peta dari berbagai filter kemudian ditumpuk membentuk dimensi kedalaman (*depth*) dari volume *output* pada lapisan konvolusional. Tabal 3.9 akan menampilkan hasil dari *feature map*.

**Tabel 3. 9** Hasil Proses Konvolusi

| | | | | | | |
|---|---|---|---|---|---|---|
| 0,4 | 0,6 | 0,6 | 0,59 | 0,57 | 0,55 | 0,36 |
| -0,02 | -0,03 | -0,04 | -0,05 | -0,05 | -0,04 | -0,02 |
| -0,02 | -0,03 | -0,04 | -0,04 | -0,04 | -0,04 | -0,03 |
| -0,02 | -0,03 | -0,03 | -0,03 | -0,04 | -0,05 | -0,04 |
| -0,02 | -0,03 | -0,03 | -0,04 | -0,05 | -0,05 | -0,03 |
| 0,0 | -0,01 | -0,03 | -0,05 | -0,05 | -0,04 | -0,02 |
| -0,36 | -0,54 | -0,53 | -0,51 | -0,48 | -0,46 | -0,3 |

### 3.3.5 *Batchnormalization*

*Batchnormalization* bekerja dengan cara menormalkan aktivasi jaringan dalam setiap *mini-batch*, sehingga mengubah *output* dari suatu lapisan atau setara dengan input ke lapisan berikutnya, sehingga memiliki *mean* dan varians yang mendekati nol. Proses perhitungan *batchnormalization* menggunakan proses *ReLU* sebagai contoh yang dapat dilihat sebagai berikut.

1. Menghitung *Mini Batch Mean*

   Jumlah mini-batch ($n$) dan jumlah data dalam setiap *mini-batch* ($m$) dari matriks *input*. Dengan matriks diatas yang berukuran 7×7, dan diketahui bahwa jumlah data dalam setiap *mini-batch* ($m$) adalah 7 dan jumlah mini-batch ($n$) adalah 7 berikut perhitungan :

   $$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

   Keterangan :

   $\mu_B$ : nilai *batch mean*

   $m$ : jumlah sample pada *mini batch*

   $x_i$ : jumlah nilai dalam batch

   $$\mu_B = \frac{1}{49}\sum_{i=1}^{m} x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + \cdots + x_{49}$$

   $$\mu_B = \sum_{i=1}^{49} \frac{0{,}4 + 0{,}6 + 0{,}6 + 0{,}59 + 0{,}57 + 0{,}55 + \cdots + (-0{,}3)}{49}$$

   $$\mu_B = \sum_{i=1}^{49} \frac{0{,}69}{49} = -0{,}01408$$

2. Menghitung *mini batch* varian

   $$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

   $$\sigma_B^2 = \sum_{i=1}^{49} \frac{(0{,}4 - (-0{,}014)^2) + (0{,}6 - (-0{,}014)^2) + \cdots + ((-0{,}3) - (-0{,}014)^2)}{49}$$

   $$\sigma_B^2 = \sum_{i=1}^{49} \frac{3{,}5145}{49} = 0{,}07173$$

3. Menghitung normalisasi nilai *input*

$$\widehat{x_i} = \frac{x_i + \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Keterangan :

$\mu_B$ : nilai *batch mean*

$\sigma_B^2$ : nilai mini *batch varian*

$\epsilon$ : 1×10−3 atau 0,001

$$\widehat{x_1} = \frac{0{,}4 + (-0{,}01408)}{\sqrt{0{,}07173 + 0{,}001}} = 1{,}53546714$$

$$\widehat{x_2} = \frac{0{,}6 + (-0{,}01408)}{\sqrt{0{,}07173 + 0{,}001}} = 2{,}27709247$$

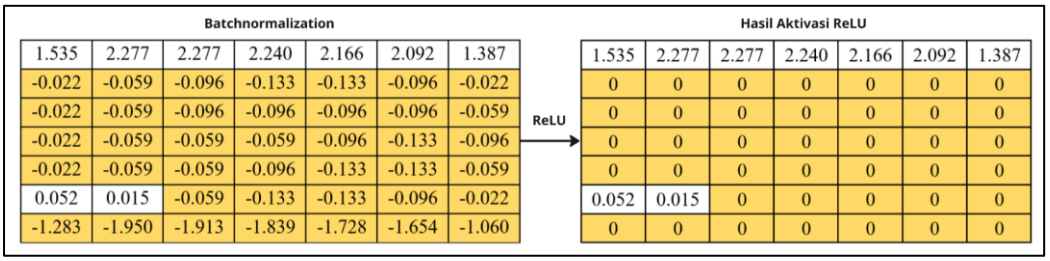
**Tabel 3. 10** Hasil *Batchnormalization*

| | | | | | | |
|---|---|---|---|---|---|---|
| 1,535 | 2,277 | 2,277 | 2,240 | 2,166 | 2,092 | 1,387 |
| -0,022 | -0,059 | -0,096 | -0,133 | -0,133 | -0,096 | -0,022 |
| -0,022 | -0,059 | -0,096 | -0,096 | -0,096 | -0,096 | -0,059 |
| -0,022 | -0,059 | -0,059 | -0,059 | -0,096 | -0,133 | -0,096 |
| -0,022 | -0,059 | -0,059 | -0,096 | -0,133 | -0,133 | -0,059 |
| 0,052 | 0,015 | -0,059 | -0,133 | -0,133 | -0,096 | -0,022 |
| -1,283 | -1,950 | -1,913 | -1,839 | -1,728 | -1,654 | -1,060 |

Pada Tabel 3.10 Merupakan hasil perhitungan keseluruhan dari proses *batch normalization* bertujuan untuk meningkatkan stabilitas dan performa saat pelatihan model.

### 3.3.6 *Activation Function*

Aktivasi *ReLU* (*Rectified Linear Unit*) adalah fungsi aktivasi yang digunakan pada model CNN, aktivasi tersebut dapat didefinisikan seperti berikut *"f(x)=max(0,x)"*. Fungsi ini menjalakan proses *thresholding* terhadap *input* pada citra dengan mempertahankan nilai yang lebih besar atau sama dengan nol. Sementara itu, semua nilai yang kurang dari nol akan diubah menjadi nol. *Aktivasi ReLU* bertujuan mempertahankan nilai positif dari hasil konvolusi dan menghilangkan nilai negatif.




**Gambar 3. 8** Aktivasi *ReLU*

Sebagai contoh perhatikan Gambar 3.8, yang terdapat sebuah citra (*feature map*) dengan dimensi 7 x 7. Beberapa sel di dalamnya memiliki nilai negatif atau kurang dari 0,

maka melalui fungsi *ReLU* akan diubah, nilai-nlai negatif tersebut diubah menjadi nilai 0, dan nilai-nilai yang positif akan tetap pada nilai aslinya.

### 3.3.7 *Pooling Layer*

*Pooling* adalah teknik untuk mengurangi ukuran matriks melalui operasi *pooling* memproses *Feature Map* sebagai input dengan berbagai operasi statistik berdasarkan nilai-nilai piksel terdekat. Dalam penelitian ini, menggunakan *Global Average Pooling*.

*Global Average Pooling (GAP)* adalah teknik yang mengambild dari perhitungan nilai rata-rata dari setiap *channel* pada *feature map*. Layer pada *GAP* digunakan untuk membantu mengatasi *overfitting* dengan mengurangi jumlah parameter pada model. Dengan mengurangi jumlah parameter proses pelatihan akan lebih cepat. *Global Average Pooling* juga mengubah matriks 3D menjadi vektor satu dimensi (1D). Pada penelitian ini, digunakan matriks 4 x 4 sebagai contoh dengan jumlah filter 1. Di dapatkan operasi GAP terhadap matriks hasil aktivasi *ReLu* ditunjukan pada Tabel 3.11 sebagai berikut.

**Tabel 3. 11** Hasil Aktivasi *ReLU*

| | | | | | | |
|---|---|---|---|---|---|---|
| 1,535 | 2,277 | 2,277 | 2,240 | 2,166 | 2,092 | 1,387 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0,052 | 0,015 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Hasil Perhitungan GAP :

$$GAP_1 = \frac{(1{,}53 + 2{,}27 + 2{,}27 + 2{,}24 + 2{,}16 + 2{,}09 + 1{,}38 + 0 + \cdots + 0)}{49} = 0{,}286$$

### 3.3.8 *Dropout*

*Dropout* digunakan untuk menghindari terjadinya *overfitting* dan dapat mempercepat proses pelatihan karena *dropout* bekerja dengan cara menghapus beberapa *neuron* secara acak saat pelatihan dari layer atau *hidden layer* sehingga jumlah parameter dan kompleksitasnya berkurang.

Cara kerja *dropout* adalah menonaktifan *neuron* selama fase pelatihan dalam suatu rangkaian *neuron* tertentu yang dipilih secara acak. Teknik *dropout* digunakan untuk mencegah *overfitting* dengan mengubah konsep dari pembelajaran seluruh bobot secara bersama-sama menjadi pembelajaran sebagian kecil bobot dalam jaringan pada setiap iterasi pelatihan, dan pada penelitian ini menggunakan *dropout layer* sebesar 0,3 atau 30%.

### 3.3.9 *Fully Connected Layer*

*Fully connected layer* adalah lapisan dalam jaringan saraf di mana setiap *neuron* terhubung langsung dengan seluruh *input* yang ada. Lapisan *Fully Connected* pada penelitian ini dilakukan setelah lapisan *Global Average Pooling 2D*, yaitu menggunakan *dense* (256) menandakan bahwa lapisan ini memiliki 256 neuron. Setiap neuron di lapisan ini terhubung

sepenuhnya (*fully connected*) dengan setiap output dari lapisan sebelumnya, yaitu lapisan *Global Average Pooling 2D*.

$$output_j = \sum_{i=1}^{n} x_i . w_i + b_j$$

Keterangan :

output = nilai aktivasi dari neuron.

n = 1280 fitur

j = neuron di layer Dense(256)

x = input vektor

w = matriks bobot

b = vektor bias

$output_j$ : j sebanyak neuron yang digunakan (256)

$$output_j = x_1 * w_{j1} + x_2 * w_{j2} + x_3 * w_{j3} + \cdots + x_{1280} * w_{j1280} + b_j$$

**Tabel 3. 12** Hasil Perhitungan *Dense* 256

| Activation | Perhitungan | Hasil |
|---|---|---|
| Output1 | (0,28*0,5)+(2,27*(−0,3))+(2,27*0,2)+...+(2,16*(−0,1))+0,05 | 0,12 |
| Output2 | (0,28*0,2)+(2,27*0,12)+(2,27*0,21)+...+(2,16*(−0,5))+0,05 | 0,02 |
| Output3 | (0,28*0,1)+(2,27*0,1)+(2,27*0,6)+...+(2,16*0,4)+0,05 | 0 |
| Output4 | (0,28*0,4)+(2,27*0,1)+(2,27*0,1)+...+(2,16*(−0,1))+0,05 | 0,087 |
| ... | ... | ... |
| Output256 | (0,28*0,2)+(2,27*0,11)+(2,27*(−0,2))+...+(2,16*0,4)+0,05 | 0,032 |

Tabel 3.12 tersebut menjelaskan proses perhitungan aktivasi yang terjadi pada layer *Dense*(256) pada jaringan saraf. Terdapat 1280 fitur *input* pada setiap proses perhitungan yang terjadi pada 256 *neuron*. Setiap neuron tersebut menerima seluruh input tersebut dan menghasilkan satu nilai *output* sebagai hasil aktivasi.

*ReLU* (*Rectified Linear Unit*) adalah fungsi aktivasi yang umum digunakan dan memetakan nilai input yang negatif menjadi 0, sementara membiarkan nilai positif tidak berubah. Dalam penelitian ini, lapisan *Fully Connected* pertama bertanggung jawab untuk memproses vektor hasil dari operasi *Global Average Pooling 2D*. Dengan 256 neuron, lapisan ini memiliki kapasitas untuk memahami relasi yang lebih kompleks antar fitur-fitur yang telah diekstrak dari lapisan-lapisan konvolusional sebelumnya. Hasil dari operasi *Output Fully Connected Layer* dapat dilihat pada Tabel 3.13.

**Tabel 3. 13** Output *Fully Connected Layer*

| Activation | Nilai | ReLU |
|---|---|---|
| Output1 | 0,12 | 0,12 |
| Output2 | 0,02 | 0,02 |
| Output3 | -0,027 | 0 |
| Output4 | 0,087 | 0,087 |
| ... | ... | ... |
| Output256 | 0,032 | 0,032 |

### 3.3.10 *Softmax*

*Softmax* adalah fungsi yang digunakan untuk menghitung distribusi probabilitas dari sebuah vektor angka. Lapisan ini terletak setelah lapisan *Fully Connected* dan memanfaatkan fungsi aktivasi *Softmax*. Lapisan dense layer (20, softmax) dalam model ini digunakan untuk mendapatkan nilai atau probabilitas klasifikasi dari 20 kelas (z) spesies ular. Untuk perhitungan dalam lapisan ini, dapat dilakukan sebagai berikut :

$$z_0 = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + \cdots + x_{256} * w_{256} + b$$
$$= (0{,}12*0{,}45)+(0{,}02*(-0{,}33))+(0*0{,}2)+\ldots+(0{,}032*(-0{,}5))+0{,}15 = 3{,}23$$

Berikut adalah seluruh nilai dari 20 node yang meginterpretasikan seluruh kelas spesies ular yang dapat dilihat pada Tabel 3.14. ini :

**Tabel 3. 14** Hasil Probabilitas 20 Spesies Ular

| Activation | z0 | z1 | z2 | z3 | ... | z19 |
|---|---|---|---|---|---|---|
| Nilai | 3,23 | 0,05 | - 0,68 | 0,23 | ... | 0,39 |

*Softmax* dapat digunakan untuk model *multi classification*. Kemudian, nilai probabilitas setiap kelas dilakukan dihitung dengan menggunakan fungsi *softmax*, berikut :

$$S(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$$S(z_0) = \frac{e^{3,23}}{\sum_{j=1}^{K} e^{3,23} + e^{0,05} + e^{-0,68} + \cdots + e^{0,39}}$$

$$S(z_0) = \frac{25{,}28}{25{,}28 + 1{,}051 + 0{,}506 + 1{,}259 + \cdots + 1{,}477} = \frac{25{,}28}{29{,}573} = 0{,}8548$$

Fungsi *softmax* bertujuan menghasilkan nilai *output* dengan rentang 0 hingga 1, kemudian memiliki jumlah probabilitas jika ditotal sama dengan 1. Fungsinya adalah untuk mengubah hasil klasifikasi gambar menjadi nilai probabilitas untuk masing kelas. Kelas dengan probabilitas tertinggi akan dianggap sebagai hasil prediksi akhir.

**Tabel 3. 15** Hasil Aktivasi dengan *Softmax*

| Spesies | 0 | 1 | 2 | 3 | ... | 19 |
|---|---|---|---|---|---|---|
| Hasil | 0,8548 | 0,0356 | 0,0171 | 0,0426 | ... | 0,0499 |

Dalam proses klasifikasi kelas dari Tabel 3.15 dengan probabilitas tertinggi akan dipilih sebagai hasil prediksi untuk gambar yang dianalisis. Ini memastikan bahwa prediksi yang dihasilkan benar-benar mencerminkan kelas yang paling sesuai dengan gambar yang diberikan. Setelah melakukan perhitungan dengan hasil 0,8548 dari citra ular yang telah diuji dari simulasi tersebut dikenali sebagai bagian dari kelas *Boiga irregularis (venomous)*. Hasil ini menunjukkan bahwa model berhasil mengidentifikasi ciri-ciri khas dari gambar tersebut yang cocok dengan kategori *Boiga irregularis (venomous)* berdasarkan probabilitas tertinggi yang telah dihitung. Kemudian nilai dari hasil pada contoh simulasi tersebut pada spesies 1 sampai 19 merupakan nilai kemiripan ciri-ciri spesies ular kepada kelas *Boiga irregularis.*

### 3.3.11 Uji *Hyperparameter*

Nilai *Hyperparameter* pada suatu model sangat penting karena mempengaruhi performa model tersebut. Dengan bereksperimen dengan berbagai kombinasi *Hyperparameter*, Anda dapat menemukan keseimbangan yang tepat antara kompeksitas model dan kemampuannya menggeneralisasi data, sehingga meningkatkan akurasi dan menghindari *Overfitting*.

*Optimizer* seperti *Adam*, *RMSprop*, dan *SGD* digunakan pada penelitian ini yang berfungsi untuk mempercepat dan menstabilkan proses pelatihan model. *Optimizer* bekerja dengan menyesuaikan bobot dalam jaringan saraf agar kesalahan prediksi model semakin kecil di setiap iterasi. Penggunaan *optimizer* ini untuk menemukan kombinasi pelatihan yang paling efektif dan efisien dalam mengoptimalkan akurasi model. *Learning rate* digunakan berdampingan dengan *optimizer* saat *compile* model. *Learning rate* berfungsi dalam menentukan besar perubahan bobot yang dilakukan pada setiap iterasi. Sehingga nilai *learning rate* tersebut harus disesuaikan pada bagian *optimizer* agar proses pembaruan bobot sesuai dengan laju yang diinginkan. *Learning rate* mempengaruhi kecepatan dan efektivitas model dalam belajar. Uji parameter ditampilkan pada tabel 3.16, seperti berikut.

**Tabel 3. 16** Rancangan Pengujian

| No | Arsitektur | Optimizer | Parameter 1<br>Epoch | Parameter 2<br>Batch Size | Parameter 3<br>Learning Rate |
|---|---|---|---|---|---|
| 1. | EfficientNetV2-S | Adam | 20 | 32 | 0,0001 |
| 2. | | | 20 | 32 | 0,00005 |
| 3. | | RMSprop | 20 | 32 | 0,0001 |
| 4. | | | 20 | 32 | 0,00005 |
| 5. | | SGD | 20 | 32 | 0,0001 |
| 6. | | | 20 | 32 | 0,00005 |

### 3.3.12 *Training* Dan *Validation*

Tahap ini merupakan proses pelatihan dan validasi yang diterapkan pada model setelah inisiasi model dan parameter pengujian ditentukan. Data pelatihan akan melalui model dalam *batch-batch* dimana prediksi model dibandingkan dengan label asli. Proses ini berlanjut selama beberapa epoch dan pada akhir tiap *epoch* performa model diuji pada data *validasi*. Data validasi digunakan untuk memastikan agar model tidak hanya belajar pada data *training* tetapi dapat bekerja dengan baik pada data yang belum dilihat sebelumnya selama proses pengembangan dan penyempurnaan model.

Dalam proses pelatihan model *deep learning*, meggunkan strategi untuk menghindari *overfitting* sekaligus memastikan efisiensi waktu komputasi. Teknik *Early Stopping* ini termasuk dalam kategori *callback* yang bekerja secara otomatis selama proses pelatihan berlangsung. *Early Stopping* berfungsi untuk menghentikan pelatihan lebih awal apabila model tidak menunjukkan peningkatan pada data validasi setelah beberapa *epoch*, sehingga dapat mencegah pemborosan sumber daya dan *overtraining*.

## 3.4 Evaluasi Dan Pengujian Model

*Confusion matrix* digunakan dalam pengujian model pada penelitian ini, berfungsi untuk mengukur performa model, yang meliputi nilai akurasi (*accuracy*), presisi (*precision*), dan *recall*. *Confusion matrix* digunakan untuk mengevaluasi kinerja sistem dalam klasifikasi. Dengan hasil evaluasi dapat diketahui sejauh mana model mampu membedakan tiap kelas yang akan diklasifikasikan. Dari hasil evaluasi menjadi acuan dalam menentukan apakah model sudah layak digunakan atau perlu dilakukan perbaikan lebih lanjut.

### 3.4.1 Akurasi

Akurasi ialah perbandingan antara jumlah prediksi yang benar dari total prediksi yang dibuat oleh model. Akurasi menjadi ukuran umum yang menunjukkan seberapa sering model membuat prediksi yang benar.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

### 3.4.2 Presisi

Presisi ialah proporsi prediksi positif yang benar dari semua prediksi positif yang dibuat oleh model. Ini mengukur akurasi prediksi positif.

$$Presisi = \frac{True\ Positives\ (\mathrm{TP})}{True\ Positives\ (\mathrm{TP})\ +\ False\ Positives\ (\mathrm{FP})}$$

### 3.4.3 *Recall*

Recall ialah proporsi prediksi positif yang benar dari semua kasus positif sebenarnya dalam data. Ini mengukur kemampuan model untuk menemukan semua kasus positif.

$$Recall = \frac{True\ Positives\ (\mathrm{TP})}{True\ Positives\ (\mathrm{TP})\ +\ False\ Negative\ (\mathrm{FN})}$$

### 3.4.4 *F1 Score*

*F1 Score* ialah harmonik rata-rata dari presisi dan *recall*. Ini memberikan keseimbangan antara keduanya dan berguna ketika Anda membutuhkan keseimbangan antara presisi dan *recall*.

$$F1\ Score = \mathbf{2} \times \frac{\mathrm{Presisi} \times Recall}{\mathrm{Presisi} + Recall}$$

Untuk menilai apakah performa baik atau tidaknya pada suatu model klasifikasi dapat dilihat dari parameter pengukuran performanya, yaitu tingkat akurasi, recall, dan presisi. Dalam proses evaluasi ini, digunakan nilai-nilai dari confusion matrix seperti "*True Positive*" (TP), "*True Negative*" (TN), "*False Positive*" (FP), dan "*False Negative*" (FN). Dari nilai tersebut, kita bisa menghitung akurasi dan mengevaluasi seberapa baik model melakukan klasifikasi. Perhitungan nilai *Accuracy*, *Precision*, *Recall*, *F1-Score*, dan support. Tabel 3.17 menunjukkan rencana pengujian dengan *confusion matrix*.

Pada Tabel 3.17 merupakan *Confusion Matrix* yang menunjukkan hasil evaluasi model klasifikasi terhadap 20 label spesies ular yang berbeda. Pada tabel ini, sumbu vertikal mewakili label aktual (Aktual), sedangkan sumbu horizontal menunjukkan label yang

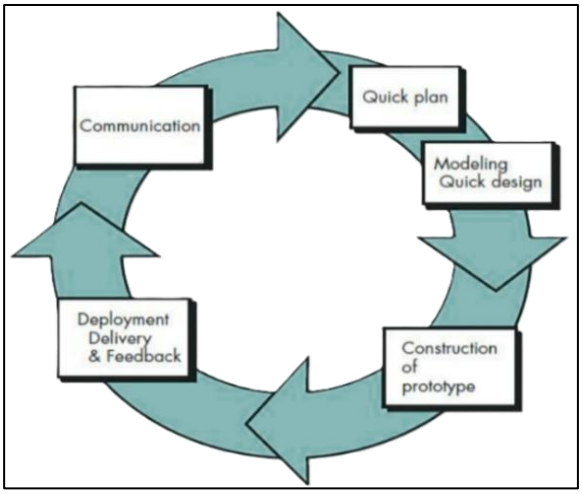
diprediksi oleh model (Prediksi). Sel diagonal yang berwarna kuning menunjukkan jumlah prediksi yang benar, yaitu ketika label aktual sama dengan label prediksi. Jika pada tabel *Confusion Matrix* berhasil mengklasifikasikan setiap spesies dengan benar tanpa adanya kesalahan klasifikasi, akan ditunjukkan dengan hanya adanya warna kuning pada diagonal dan tidak ada warna di luar diagonal. Hal ini menandakan bahwa model memiliki performa yang sangat baik atau bahkan sempurna dalam mengklasifikasikan data spesies yang diuji.

Tabel rancangan confusion matrix dapat dilihat pada Tabel 3.17, pengujian dilakukan terhadap masing-masing kelas ular yang digunakan kemudian akan diketahui *accuracy*, *precission*, dan *recall* rata-rata.

**Tabel 3. 17** Rancangan *Confusion Matrix*

| | | Actual | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Label Spesies** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **...** | **18** | **19** | **20** |
| **Prediksi** | **1** | | | | | | | | | | | | | | |
| | **2** | | | | | | | | | | | | | | |
| | **3** | | | | | | | | | | | | | | |
| | **4** | | | | | | | | | | | | | | |
| | **5** | | | | | | | | | | | | | | |
| | **6** | | | | | | | | | | | | | | |
| | **7** | | | | | | | | | | | | | | |
| | **8** | | | | | | | | | | | | | | |
| | **9** | | | | | | | | | | | | | | |
| | **10** | | | | | | | | | | | | | | |
| | **...** | | | | | | | | | | | | | | |
| | **18** | | | | | | | | | | | | | | |
| | **19** | | | | | | | | | | | | | | |
| | **20** | | | | | | | | | | | | | | |

## 3.5 Pengembangan Sistem




**Gambar 3. 9** Metodologi *Prototyping*
Sumber (Kurniati, 2021)

Tahap pengembangan sistem dapat dilihat pada Gambar 3.9 yang merupakan tahap metode *Prototyping* yang dilakukan untuk merancang sistem secara keseluruhan hingga

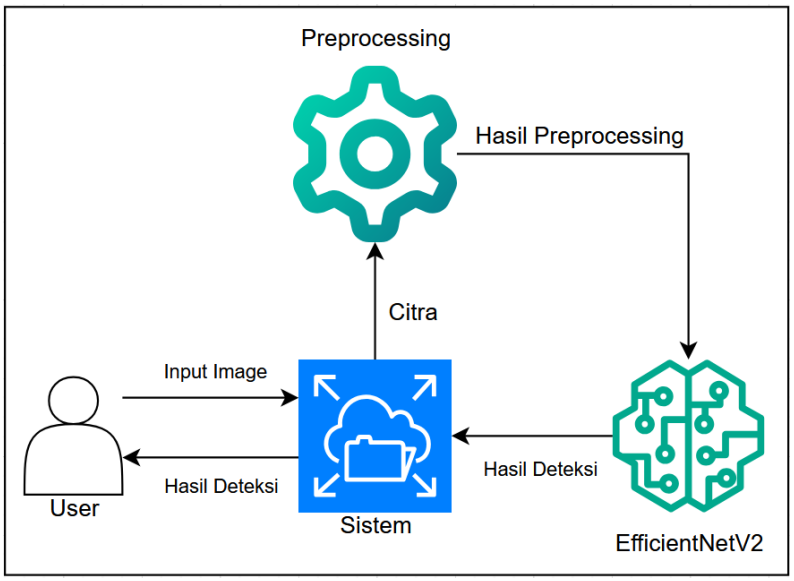
memasuki proses pengkodean hingga pengujian, dimana hasil dari pengembangan ini berupa platform klasifikasi ular. Pengembangan sistem secara rinci ditunjukkan seperti dibawah ini.

### 3.5.1 Pengumpulan Kebutuhan (*Communication*)

Pada tahapan awal ini pengembangan sistem, langkah pertama yang dilakukan adalah membuat analisis kebutuhan sistem. Identifikasi dan pengumpulan data terkait kebutuhan sistem, baik dari sisi data maupun proses yang akan dilakukan. *Dataset* citra ular dari sumber *kaggle* yang sesuai dengan kebutuhan penelitian. Data diambil dari *website Kaggle* yang bernama Dataset *Snakes species* "https://www.kaggle.com/datasets/goelyash/165-different-snakes-species" dengan nama author Yash Goel yang di update 3 tahun yang lalu dan melakukan *web scrapping* untuk menambah data citra ular. Kebutuhan mencakup fitur dan fungsi yang harus dimiliki oleh sistem agar dapat menyelesaikan tugas-tugas utamanya, yaitu melakukan klasifikasi gambar spesies ular. Sistem dapat menerima gambar ular yang akan diklasifikasi sesuai spesies, kemudian sistem akan menampilkan hasil prediksi yang telah meggunakan model *transfer learning* dengan arsitektur *EfficientNetV2S* karena efisiensi dan performanya yang baik dalam klasifikasi citra.

### 3.5.2 Perancangan dan pemodelan sistem (*Quick Plan*)

Tahapan ini mencakup perancangan arsitektur sistem, proses aliran data, serta logika pemrosesan yang akan digunakan. Dalam perencanaan pemodelan sistem ini, bertujuan mendefinisikan kebutuhan dasar dari pengguna serta merumuskan alur kerja sistem secara menyeluruh sebelum diimplementasikan.

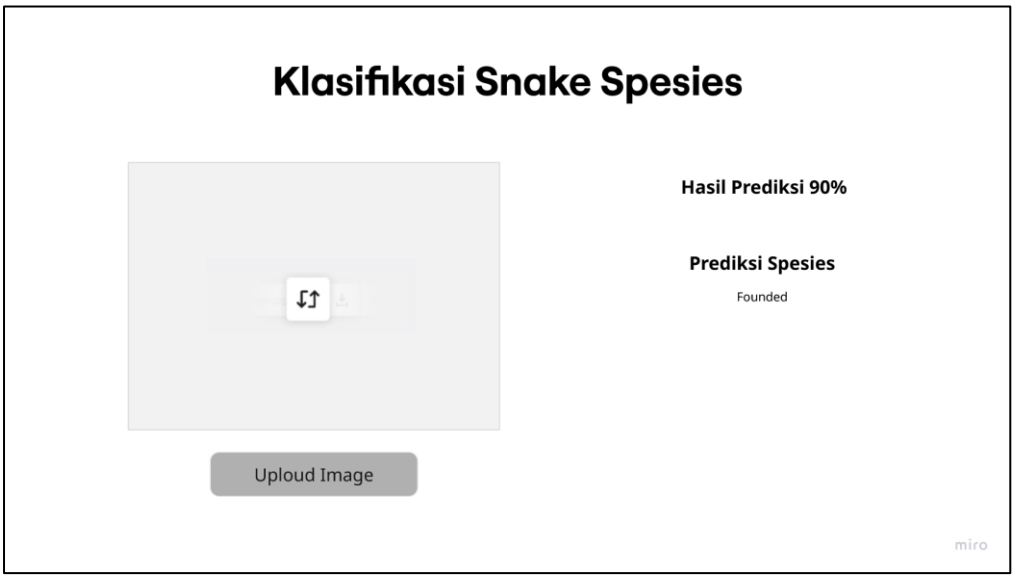



**Gambar 3. 10** Perancagan dan Pemodelan Sistem

Arsitektur sistem pada Gambar 3.10 diawali dengan *user* menginput citra ular kedalam platform klasifikasi spesies ular, kemudian sistem akan melakukan *pre-processing* pada citra tersebut. Setelah citra mengalami proses *pre-processing*, citra akan memasuki proses deteksi dengan menggunakan model *EfficienNetV2-S*. Setelah seluruh proses selesai dilakukan, sistem akan memberikan hasil deteksi pada pengguna.

### 3.5.3 Perancangan Antarmuka Pengguna (*Quick Design*)

Analisis fitur dalam sistem ini berfokus pada identifikasi fungsi-fungsi utama yang harus tersedia untuk mendukung proses klasifikasi. Desain sistem pada penelitian ini dilakukan sebagai rancangan antar muka aplikasi, data input, prosedur preprocessing, dan hasil dari prediksi klasifikasi.




**Gambar 3. 11** Halaman Utama Klasifikasi Spesies Ular

Gambar 3.11 menampilkan rancangan desain antarmuka sistem klasifikasi spesies ular. Dalam tampilan ini, pengguna diberikan opsi untuk memasukkan gambar yang akan dianalisis. Pertama, pengguna dapat menekan tombol "Uploud Image" untuk mengambil/ mengunggah gambar yang telah tersimpan di perangkat mereka.

Setelah gambar berhasil dimasukkan, sistem akan secara otomatis menjalankan proses deteksi dan klasifikasi objek pada gambar tersebut. Hasil dari proses ini akan ditampilkan dalam bentuk label spesies ular, tingkat akurasi prediksi (misalnya 96%). Tampilan hasil ini dirancang agar informatif dan mudah dipahami oleh pengguna.

### 3.5.4 Implementasi (*Construction of Prototype*)

Setelah tahap perancangan antarmuka pengguna selesai dilakukan, proses pengembangan sistem dilanjutkan dengan mengintegrasikan model *machine learning* ke dalam sistem *website*. Implementasi merupakan tahap pengembangan sistem untuk merancang sistem dari hasil desain dan merealisasikan secara keseluruhan hingga memasuki proses pengkodean, dimana pengkodean menghasilkan hasil akhir berupa platform klasifikasi ular. Pada tahap perancangan aplikasi ini dilakukan dengan bahasa pemrograman *phyton* dan *library* tensorflow/*keras* yang digunakan.

Model dalam penelitian ini dibangun menggunakan pendekatan *transfer learning* dengan arsitektur *EfficientNetV2S*, yang dikenal efektif dalam klasifikasi gambar. Proses pelatihan dilakukan di *Jupyter Notebook*, di mana berbagai eksperimen dilakukan untuk mencapai akurasi terbaik dalam mengenali 20 spesies ular. Setelah model dilatih dan hasilnya memuaskan, model disimpan dalam format ".h5" agar bisa digunakan kembali di

sistem *website*. Model yang sudah jadi ini kemudian diintegrasikan ke dalam sistem berbasis *website*. Nantinya, setiap kali pengguna mengunggah gambar ular, sistem akan memproses gambar tersebut, mengirimkannya ke model, dan menampilkan hasil prediksi berupa nama spesies serta informasi apakah ular tersebut berbisa atau tidak. Semua ini berlangsung secara otomatis dan ditampilkan langsung di halaman *website*, sehingga mudah diakses oleh pengguna.

### 3.5.5 Pengujian (*Deployment Delivery & Feedback*)

Tahap pengujian merupakan tahap *website* sudah dapat digunakan oleh pengguna. Pengguna dapat menggunakan sistem dalam bentuk *website* yang sebelumnya dirancang dalam bentuk *wireframe* pada proses desain. Setelah aplikasi selesai dikembangkan, kemudian dilakukan evaluasi sistem dengan cara metode pengujian *black-box*. Metode *black-box* adalah jenis pengujian yang menilai sistem berdasarkan hasil *output*, tanpa melihat bagaimana logika atau struktur kode di dalamnya bekerja. Fokus utama dari metode ini adalah memastikan bahwa sistem merespons dengan benar terhadap setiap *input* yang diberikan pengguna. Pendekatan ini sangat tepat digunakan pada sistem berbasis *website* seperti yang dikembangkan dalam penelitian ini, karena pengguna hanya berinteraksi melalui antarmuka dan tidak perlu memahami proses teknis di balik layar.

Metode *black-box* testing sangat cocok digunakan untuk menemukan kesalahan pada fungsi-fungsi utama sistem, khususnya di bagian antarmuka pengguna, tempat di mana pengguna berinteraksi langsung dengan aplikasi. Pendekatan ini membantu memastikan bahwa setiap tombol, fitur, dan respons sistem bekerja sesuai yang diharapkan saat digunakan oleh pengguna secara nyata. Berikut adalah tabel rincian pengujian sistem yang akan dilakukan dengan metode *black box*, yang mencakup berbagai skenario pengujian, input yang digunakan, serta *output* yang diharapkan dari sistem.

**Tabel 3. 18** Pengujian Sistem

| No | Pengujian | Ekspektasi Hasil | Hasil |
|---|---|---|---|
| 1 | Pengguna dapat mengambil atau mengunggah gambar ular yang telah tersimpan di perangkat mereka. | Pengguna berhasil mengunggah gambar. | |
| 2 | Pengguna dapat mengunggah gambar melalui tombol “Upload Image” untuk melakukan klasifikasi ular. | Pengguna berhasil mengunggah gambar melalui tombol “Upload Image” untuk melakukan klasifikasi ular. | |
| 3 | Sistem dapat menampilkan hasil prediksi spesies ular. | Sistem berhasil menampilkan hasil hasil prediksi spesies ular. | |

Pada Tabel 3.18 berisikan tiga skenario pengujian *black-box* yang dirancang untuk menguji fungsi utama sistem. Jika hasil dari ketiga skenario ini menunjukkan *output* yang sesuai dengan harapan, maka dapat disimpulkan bahwa sistem telah berhasil melewati tahap pengujian dan berfungsi sebagaimana mestinya.

# BAB IV
# HASIL PENGUJIAN DAN PEMBAHASAN

## 4.1 Implementasi

Pada tahap implementasi akan menjelaskan bagaimana rancangan yang telah disampaikan pada bab Metodologi Penelitian. Tahap implementasi menunjukkan tahapan penelitian menjadi produk akhir dan menyampaikan hasil dari dilakukannya rancangan ini.

### 4.1.1 Pengumpulan Data

Gambar spesies ular diambil dari website *Kaggle* yang bernama Dataset *Snakes species* "*https://www.kaggle.com/datasets/goelyash/165-different-snakes-species*" dengan nama author Yash Goel yang di update 3 tahun yang lalu dan untuk menambah jumlah data dilakukan juga sccrapping dari website *Gbif*.

Pada proses ini data disimpan pada folder kemudian diambil yaitu 20 jenis ular, kemudian diklasifikasikan sebagai spesies ular tidak berbisa (*non-venomous*) dan ular berbisa (*venomous*).

**Source Code 1: Unggah Data Dari Folder**

```
import os
DATASET_PATH = "dataset/"

def load_dataset():
if not os.path.exists(DATASET_PATH):
   print(f"[!] Folder '{DATASET_PATH}' tidak ditemukan.")
   return []

   all_classes = [d for d in os.listdir(DATASET_PATH)
               if os.path.isdir(os.path.join(DATASET_PATH, d))]
   if not all_classes:
     print("Tidak ada subfolder kelas yang ditemukan di dalam dataset.")
   else:
     print("Dataset ditemukan dengan kelas Ular :")
     for cls in all_classes:
        class_path = os.path.join(DATASET_PATH, cls)
        img_files = [f for f in os.listdir(class_path)
        if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
        print(f" - {cls} ({len(img_files)} gambar)")
   return all_classes

```

### 4.1.2 *Preprocessing* Data

Tahap ini adalah mempersiapkan data agar lebih optimal untuk proses pemodelan, dengan menerapkan berbagai teknik pemrosesan data. Setelah unggah data akan dilakukan proses pembagian data kemudian dilakukan augmentasi.

1. Pembagian Data

   Dataset dibagi menjadi 3 dengan perbandingan data *training* yang besar 80%, untuk mempelajari pola lebih kompleks dari dataset. Data *validation* 10% digunakan untuk mengukur kinerja model pada data yang belum pernah dilihat sebelumnya. Sedangkan

data *testing* 10% digunakan untuk menguji kinerja model pada data yang benar-benar baru dan independen yang mempresentasikan dunia nyata.

**Source Code 2: Pembagian Data**

```
import os
import shutil
from sklearn.model_selection import train_test_split

train_ratio = 0.8
val_ratio = 0.1
test_ratio = 0.1

DATASET_PATH = "dataset/"
OUTPUT_PATH = "split_data_ular/"

def split_dataset_multiclass():
all_classes = [d for d in os.listdir(DATASET_PATH)
               if os.path.isdir(os.path.join(DATASET_PATH, d))]
    result_dict = {}

    for split in ['train', 'val', 'test']:
        for class_name in all_classes:
            split_path = os.path.join(OUTPUT_PATH, split, class_name)
            os.makedirs(split_path, exist_ok=True)

    for class_name in all_classes:
        class_path = os.path.join(DATASET_PATH, class_name)
        all_imgs = [os.path.join(class_path, f)
                    for f in os.listdir(class_path)
                    if f.lower().endswith(('.jpg', '.jpeg', '.png'))]

        if not all_imgs:
            print(f"Folder kosong: {class_name}")
            continue
train_val_imgs, test_imgs =
train_test_split(all_imgs,test_size=test_ratio, random_state=42)

val_ratio_adj = val_ratio / (train_ratio + val_ratio)

train_imgs, val_imgs = train_test_split(
train_val_imgs, test_size=val_ratio_adj, random_state=42)

        result_dict[class_name] = {
            'train': len(train_imgs),
            'val': len(val_imgs),
            'test': len(test_imgs)}
```

*2. Augmentasi*

*Augmentasi* data adalah proses yang digunakan untuk menambah variasi data dalam proses pelatihan dengan menciptakan memodifikasi data yang sudah ada tanpa mengubah

data label. Pada tahap ini ukuran gambar diubah menjadi 384 x 384 piksel. Selanjutnya gambar akan dilakukan *Rescale* menggunakan fungsi bawaan dari *TensorFlow*, yaitu "*preprocess_input*", yang secara khusus disesuaikan untuk arsitektur *EfficientNetV2S*. Kemudian pada penelitian ini. Ukuran ini dipilih karena menyesuaikan arsitektur E*fficientNetV2S* saat dilatih sebelumnya dengan *ImageNet*.

**Source Code 3: Augmentasi Data**

```
IMG_SIZE = (384, 384)

train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    zoom_range=0.3,
    horizontal_flip=True
)

val_test_datagen =
ImageDataGenerator(preprocessing_function=preprocess_input)
```

### 4.1.3 Implementasi *EfficinetNetV2S*

1. Implementasi *EfficientNetV2S*

Membangun model CNN dengan arsitektur *EfficientNetV2S* yang telah dilatih dengan *ImageNet* adalah kumpulan data gambar berskala besar yang berisi lebih dari 14 juta gambar dengan masing-masing telah diberi label berdasarkan kategori objek yang ada di dalamnya. Dengan menggunakan model *EfficientNetV2S* yang telah terlatih dapat menghemat waktu karena tidak perlu melatih ulang seperti bentuk visual ular (seperti bentuk, pola, tekstur).

**Source Code 4: Implementasi EfficientNetV2S**

```
base_model = EfficientNetV2S(weights="imagenet", include_top=False,
input_shape=(384, 384, 3))
```

*2. Fine Tuning*

Model *EfficientNetV2S* dimuat dengan bobot *pre-trained* dari *ImageNet*, lalu menyesuaikannya dengan dataset baru yang lebih spesifik. Dengan teknik *fine-tuning*, yaitu pelatihan lanjutan pada model *pre-trained* agar mampu menangkap fitur yang lebih relevan terhadap dataset baru secara lebih akurat. *Fine tuning* hanya mengambil 100 lapisan terakhir dari model untuk dilatih ulang. Sedangkan lapisan-lapisan awal diabaikan karena sudah belajar fitur visual dasar seperti tepi, bentuk, dan tekstur.

**Source Code 5: Fine Tuning**

```
base_model.trainable = True
# Fine-tuning: buka 100 layer terakhir
for layer in base_model.layers[:-100]:
    layer.trainable = False
```

*3. Top Head Layer*

Pada bagian *Top Head Layer*, lapisan akhir dari model disesuaikan agar *output* sesuai dengan jumlah kelas spesies ular dalam dataset, sehingga model bisa belajar mengenali kategori yang lebih spesifik. Fungsi *softmax* kemudian diterapkan untuk mengubah *output* tersebut menjadi probabilitas yang menjumlah hingga 1, sehingga memudahkan model dalam menentukan kelas dengan tingkat keyakinan tertinggi.

**Source Code 6: Top Head Layer**

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(num_classes, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=output)
```

*4. Optimizer Adam*

Proses pelatihan menggunakan *optimizer Adam* karena adaptif dalam menyesuaikan laju pembelajaran, membuatnya efisien dan stabil dalam menangani data yang kompleks. Dengan nilai *learning rate* sebesar 0.0001, yang berarti model belajar secara perlahan namun lebih stabil. *Learning rate* yang kecil membantu model untuk menyesuaikan bobot secara hati-hati meskipun proses *training* menjadi lebih lambat.

**Source Code 7: Optimizer**

```
model.compile(optimizer=Adam(
              learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

### 4.1.4 *Training Data*

Proses pelatihan dan validasi yang diterapkan pada model setelah inisiasi model dan parameter pengujian ditentukan. Data pelatihan akan melalui model dalam batch-batch dimana prediksi model dibandingkan dengan label asli. Model dilatih menggunakan data latih (*training* data) untuk mengenali pola-pola visual yang ada pada gambar ular, sekaligus selalu dievaluasi menggunakan data validasi untuk mengukur kinerja secara berkala.

**Source Code 8: Training Data**

```
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=20,)
```

### 4.1.5 Implementasi Sistem

Sistem dibuat setelah model yang telah disimpan, selanjutnya akan diintegrasikan ke sebuah website. website yang dikembangkan akan menggunakan bahasa pemrograman *html* dan *java script*. Hasil dari integrasi ini adalah sebuah *website* yang memungkinkan pengguna dapat memasukkan gambar yang kemudian akan diklasifikasikan oleh sistem untuk mengidentifikasi spesies ular di dalam gambar tersebut. Dalam sistem ini, gambar ular yang diunggah oleh pengguna akan diproses oleh model. Hasil akhir akan berupa klasifikasi nama spesies ular, kategori apakah ular tersebut berbisa atau tidak, serta jenis racun yang dimiliki (jika berbisa).

**Source Code 9: Impementasi Sistem**

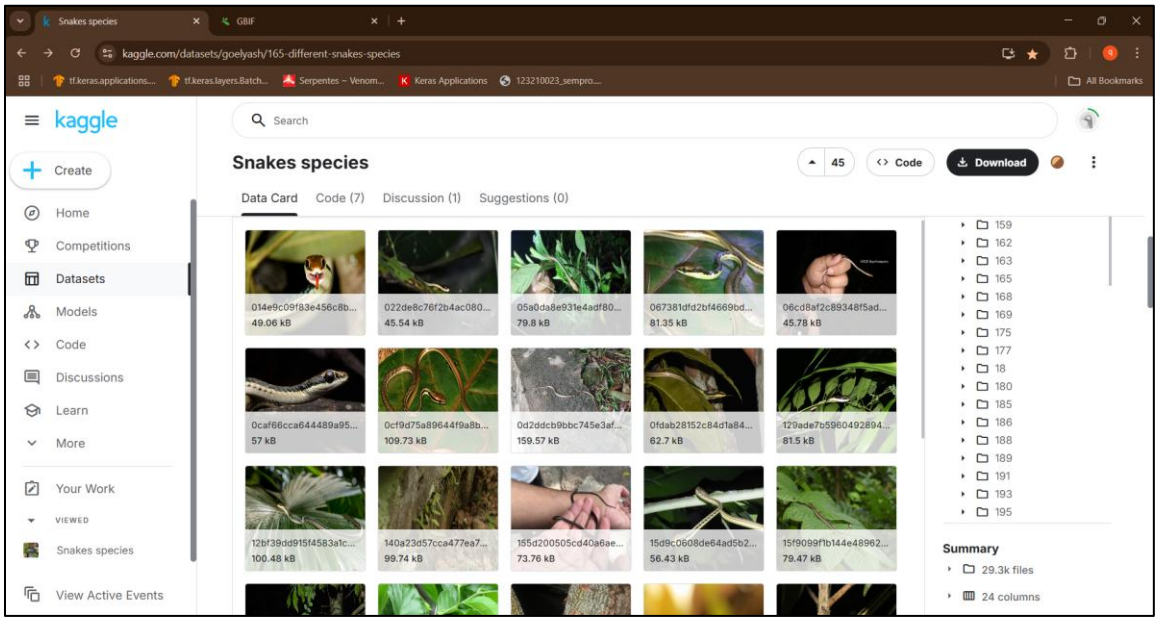
```
app = Flask(__name__)
UPLOAD_FOLDER = 'static/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

os.makedirs(UPLOAD_FOLDER, exist_ok=True)
model = load_model('model/spesies_ular_92.83.h5')

def model_predict(img_path):
    img = image.load_img(img_path, target_size=(384, 384))
    img_array = image.img_to_array(img)
    img_array = preprocess_input(img_array)
    img_array = np.expand_dims(img_array, axis=0)

    predictions = model.predict(img_array)[0]
    pred_index = np.argmax(predictions)
    confidence = predictions[pred_index]

    pred_label =
                next(name for name, info in class_info.items()
                if info["index"] == pred_index)
    venom_label = "Berbisa"
                if class_info[pred_label]["venom"] == 1 else "Tidak Berbisa"
    toxin_type = class_info[pred_label]["toxin_type"]

    all_predictions = [
        (name, float(predictions[info["index"]]))
        for name, info in class_info.items()
    ]
    all_predictions.sort(key=lambda x: x[1], reverse=True)

    return pred_label, confidence, venom_label, toxin_type, all_predictions

```

## 4.2 Hasil

Pada tahap ini akan dijelaskan hasil penelitian dari implementasi model *Convolutional Neural Network* (CNN) dengan menggunakan arsitektur *EfficientNetV2S* untuk klasifikasi spesies ular. Pada bab hasil akan menunjukkan bagaimana model dapat membedakan spesies ular dan seberapa baik performa model dalam melakukan klasifikasi berdasarkan data yang tersedia. Hasil akan menjelaskan dimulai dari tahap pertama yaitu hasil pengumpulan data, hingga hasil dari proses pengujian sistem yang dilakukan pada penelitian.

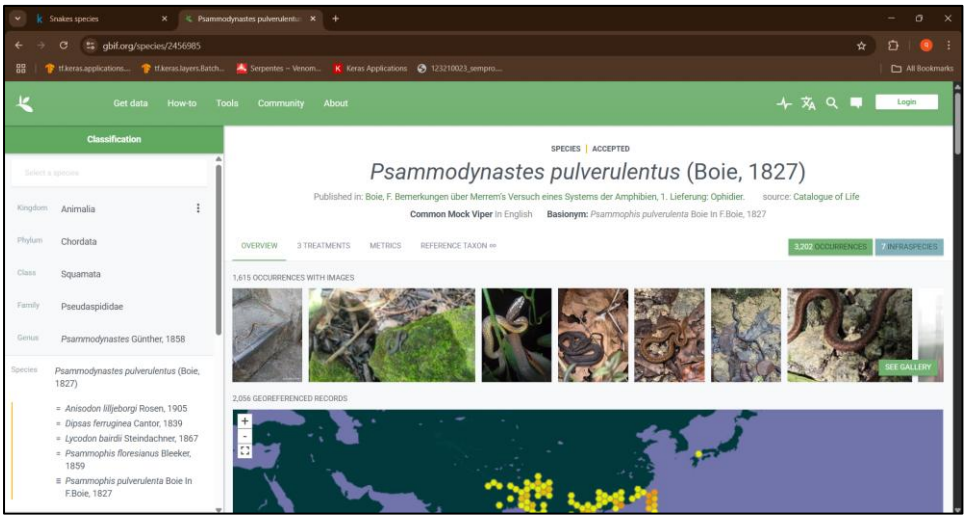
### 4.2.1 Hasil Pengumpulan Data

Data ular didapatkan melalui pengumpulan dataset dari *kaggle* dan untuk penambahan data dilakukan dengan *web scrapping* melalui *website Global Biodiversity Information Facility* (GBIF). GBIF merupakan organisasi internasional yang menyediakan akses terbuka ke data tentang keanekaragaman hayati di seluruh dunia. Hasil dari pengumpulan data dari *Kaggle* yang bernama *Dataset Snakes species "https://www.kaggle.com/datasets/goelyash/165-different-snakes-species"* dengan nama author Yash Goel yang di update 3 tahun yang lalu. Kemudian diambil 20 spesies yang akan digunakan pada penelitian. Gambar 4.1 ditunjukkan hasil dalam pengumpulan data melalui *kanggle*.




**Gambar 4. 1** *Kaggle Dataset Snakes species*

Selanjutnya dilakukan pemilihan data gambar ular, karena terdapat kecacatan atau terdapat gambar yang tidak jelas yang dilakukan penghapusan secara manual. Kemudian agar data gambar untuk setiap spesies ular agar seimbang dilakukan penambahan data melalui *web scrapping* melalui *website GBIF*. Setiap gambar pada 20 spesies ular yang digunakan disama ratakan yang berjumlah setiap spesies memiliki 232 untuk setiap spesies. Pada Gambar 4.2 merupakan penampilan dari website GBIF saat mencari data gambar untuk penambahan pada dataset yang digunakan.




**Gambar 4. 2** *Web Scrapping Gbif*

### 4.2.2 Hasil *Preprocessing* Data

Hasil Data *pre-processing* dalam penelitian menghasilkan dua tahap, yaitu saat proses pembagian dataset dan proses *augmentasi* data untuk memperkaya data latih. Berikut adalah beberapa hasil tahapan *preprocessing* data :

1. Hasil Pembagian Dataset

   Terdapat tiga pembagian data yang dilakukan yaitu *training*, *validation*, dan *testing*. Jumlah dataset yang sebelumnya 4640 akan dibagi menjadi 80% untuk *training*, 10% untuk *validation*, dan *testing* 10% yang mempresentasikan data diluar data saat pelatihan.

   Pemilihan dengan proporsi 80:10:10 dilakukan setelah melakukan berbagai percobaan lain. Seperti pembagian dengan 70:20:10 yang menunjukkan hasil pelatihan sedikit lebih rendah, hal ini diduga karena pelatihan yang lebih kecil sebesar 70% membuat model kurang optimal dalam mempelajari pola dari setiap spesies ular untuk 20 kelas.

   Hasil pemilihan dengan menggunakan proporsi 80:10:10 pada penelitian ini karena memberikan keseimbangan yang lebih baik antara kebutuhan pelatihan dan evaluasi, sekaligus menjaga efisiensi eksperimen. Dengan jumlah total data sebanyak 4640 citra gambar spesies ular, dilakukan pembagian menjadi tiga bagian sebagaimana ditampilkan pada Tabel 4.1. Data *training* terdiri dari 3680 citra (80%) digunakan model untuk mempelajari pola ular. Selanjutnya, 480 citra (10%) untuk data validasi berfungsi memantau performa model selama pelatihan dan mendeteksi potensi *overfitting*. Kemudian, 480 citra (10%) digunakan sebagai data pengujian (testing), yaitu data yang tidak dilibatkan dalam proses pelatihan maupun validasi dan digunakan untuk menilai kinerja akhir model secara objektif.

**Tabel 4. 1** Hasil Pembagian Data

| No | Pembagian Data | Jumlah |
|---|---|---|
| 1 | *Training* | 3680 |
| 2 | *Validation* | 480 |
| 3 | *Testing* | 480 |
| Total | | 4640 |

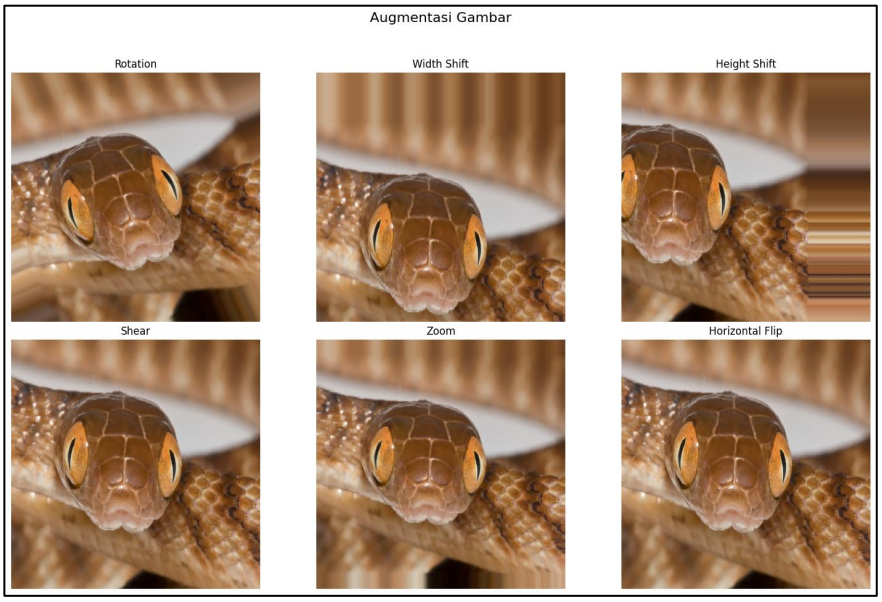
Dataset multi-klasifikasi berhasil di-split!

| Class | Train | Val | Test |
|---|---|---|---|
| Ahaetulla prasina (0) | 184 | 24 | 24 |
| Bitis gabonica (1) | 184 | 24 | 24 |
| Boiga irregularis (1) | 184 | 24 | 24 |
| Boiga kraepelini (0) | 184 | 24 | 24 |
| Bungarus multicinctus (1) | 184 | 24 | 24 |
| Chrysopelea ornata (0) | 184 | 24 | 24 |
| Daboia russelii (1) | 184 | 24 | 24 |
| Dendrelaphis pictus (0) | 184 | 24 | 24 |
| Gonyosoma oxycephalum (0) | 184 | 24 | 24 |
| Laticauda colubrina (1) | 184 | 24 | 24 |
| Lycodon capucinus (0) | 184 | 24 | 24 |
| Morelia viridis (0) | 184 | 24 | 24 |
| Ophiophagus hannah (1) | 184 | 24 | 24 |
| Protobothrops mucrosquamatus (1) | 184 | 24 | 24 |
| Psammodynastes pulverulentus (0) | 184 | 24 | 24 |
| Python molurus (0) | 184 | 24 | 24 |
| Rhabdophis subminiatus (1) | 184 | 24 | 24 |
| Tropidolaemus subannulatus (1) | 184 | 24 | 24 |
| Tropidolaemus wagleri (1) | 184 | 24 | 24 |
| Xenochrophis piscator (0) | 184 | 24 | 24 |

**Gambar 4. 3** Data Ular yang digunakan

2. *Augmentasi* Data

Hasil dari *augmentasi* data yang dilakukan pada penelitian ini yaitu yang pertama yaitu melakukan *rescale* data. Kemudian hasil tersebut akan dilakukan beberapa proses *augmentasi* yaitu *rotation range, width, height, shear range, zoom range, horizontal flip,* terhadap gambar asli untuk menghasilkan versi baru yang berbeda pada data latih. *Augmentasi* gambar akan dilakukan berbeda-beda tiap *epoch* pada setiap pelatihan yang digunakan untuk memperkaya variasi data *training* dari kombinasi beberapa proses *augmentasi*. Hasil augmentasi dapat dilihat pada Gambar 4.4.




**Gambar 4. 4** Hasil *Augmentasi*

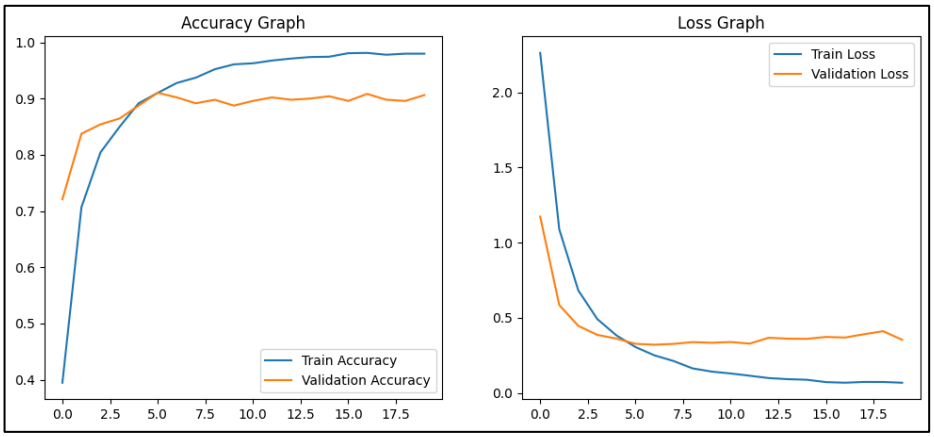
### 4.2.3 Hasil *Training EfficietNetV2S*

Hasil model pada penelitian ini dilakukan dengan menggunakan metode CNN dengan arsitektur *EfficietNetV2S*. Pelatihan yang dilakukan menghabiskan waktu rata-rata

untuk semua percobaan 28 menit. Pelatihan model dilakukan percobaan dengan tiga jenis *optimizer* yang berbeda yaitu *Adam, RMSprop, dan SGD*. Masing-masing pelatihan dijalankan selama 20 *epoch* dan menggunakan variasi nilai *learning rate*. Hasil pengukuran waktu menunjukkan bahwa proses *training* tercepat diperoleh oleh *optimizer Adam* dengan waktu sekitar 27 menit, diikuti oleh *SGD* yang memiliki durasi pelatihan yang hampir sama. Sementara itu, *RMSprop* membutuhkan waktu pelatihan paling lama yaitu sekitar 35 menit. Perbedaan waktu ini menunjukkan bahwa pemilihan *optimizer* dan *learning rate* sanagt berpengaruh ,tidak hanya terhadap akurasi tetapi juga efisiensi waktu dalam proses pelatihan model.

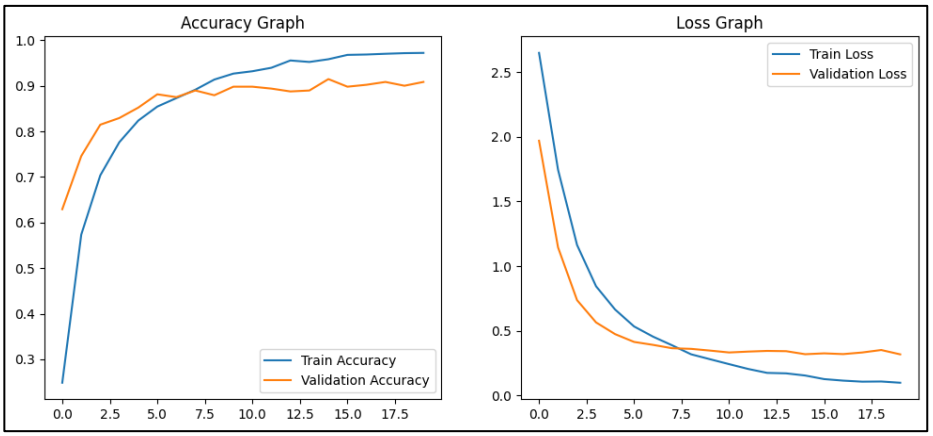
Hasil dari perbandingan percobaan untuk mencari akurasi dan efisiensi waktu pelatihan dilakukan untuk masing-masing *optimizer Adam, RMSprop, dan SGD* menggunakan nilai *learning rate* yang berbeda selama 20 epoch. Setiap optimizer yaitu *Adam, RMSprop, dan SGD* menunjukkan karakteristik performa yang unik terhadap perubahan *learning rate* yang digunakan yaitu 0,0001 dan 0,0005 pada saat percobaan *training*. Dari setiap percobaan akan menampilkan grafik akurasi dan validasi, selanjutnya dilakukan juga evaluasi dengan menggunakan *confusion matrix* untuk menilai setiap percobaan yang dilakukan. Hasil dari percobaan ini ditampilkan Sebagai berikut:

1. *Optimizer Adam*




**Gambar 4. 5** *Optimizer Adam Learning Rate* 0,0001

Pelatihan dengan *optimizer Adam learning rate* 0,0001 pada Gambar 4.5 grafik pelatihan dan validasi terlihat bahwa akurasi pelatihan meningkat secara konsisten hingga mendekati 1 menandakan model mampu belajar dengan baik. Namun, akurasi validasi cenderung stabil di sekitar 90% setelah *epoch* ke-5, yang bisa menjadi indikasi bahwa model mulai mengalami *overfitting*. Grafik sebelah kanan menunjukkan nilai *loss* untuk data pelatihan dan validasi. Nilai *loss* pelatihan menurun drastis hingga mendekati nol, menandakan bahwa model berhasil meminimalkan kesalahan pada data pelatihan. Sebaliknya, loss pada data validasi menurun pada awal pelatihan namun mulai meningkat tipis setelah *epoch* ke-7. Ini memperkuat dugaan bahwa model mengalami *overfitting*, yaitu terlalu menyesuaikan diri terhadap data pelatihan dan kehilangan generalisasi pada data baru.

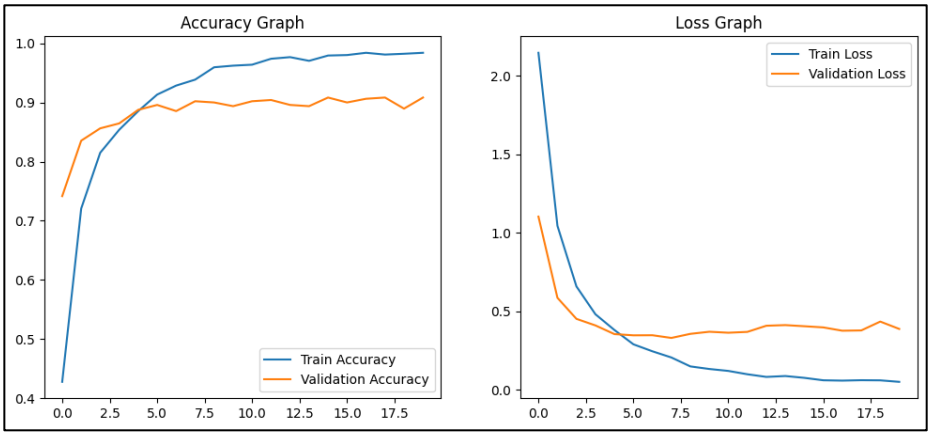



**Gambar 4. 6** *Optimizer Adam Learning Rate* 0,00005

Gambar 4.6 menunjukkan hasil pelatihan model menggunakan *optimizer Adam learning rate* 0,00005. Dari grafik akurasi, terlihat bahwa baik akurasi pelatihan maupun validasi meningkat secara stabil hingga di atas 90%, tanpa adanya penurunan signifikan pada akurasi validasi, yang menandakan proses pelatihan berlangsung cukup baik. Grafik loss juga menunjukkan penurunan yang konsisten pada kedua data, dengan nilai loss validasi yang lebih rendah dari loss pelatihan pada beberapa titik, mengindikasikan bahwa model memiliki kemampuan generalisasi yang baik. Dengan learning rate yang lebih kecil, pelatihan berjalan lebih lambat namun lebih stabil, dan cenderung menghindari overfitting dibandingkan konfigurasi sebelumnya.

Hasil pelatihan Gambar 4.5 dan Gambar 4.6 dengan *optimizer Adam*, akurasi tertinggi diperoleh saat menggunakan *learning rate* sebesar 0,0001 mencapai 93,12%. Namun, *learning rate* 0,00005, akurasi sedikit lebih rendah yaitu 91,46%.

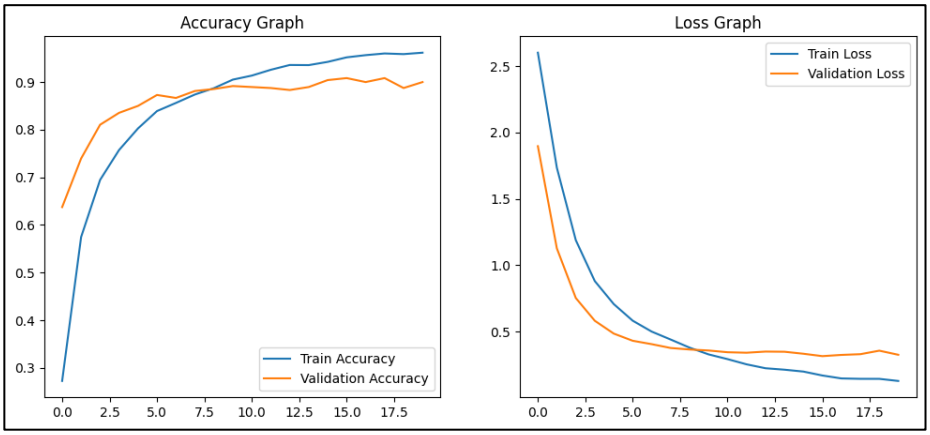
2. *Optimizer RMSprop*




**Gambar 4. 7** *Optimizer RMSprop Learning Rate* 0,0001

Gambar 4.7 menunjukkan hasil pelatihan model menggunakan *optimizer RMSprop* dengan *learning rate* 0,0001. Grafik akurasi memperlihatkan bahwa akurasi pelatihan meningkat pesat dan stabil di atas 90%, sedangkan akurasi validasi bertahan di kisaran 90%, yang menandakan performa cukup baik namun dengan

indikasi awal *overfitting*. Grafik *loss* menunjukkan bahwa *loss* pelatihan turun drastis mendekati nol, sedangkan *loss* validasi mulai stagnan dan sedikit meningkat setelah *epoch* ke-7. Hal ini mengindikasikan bahwa model mulai terlalu menyesuaikan data pelatihan dan kehilangan sedikit kemampuan generalisasi.

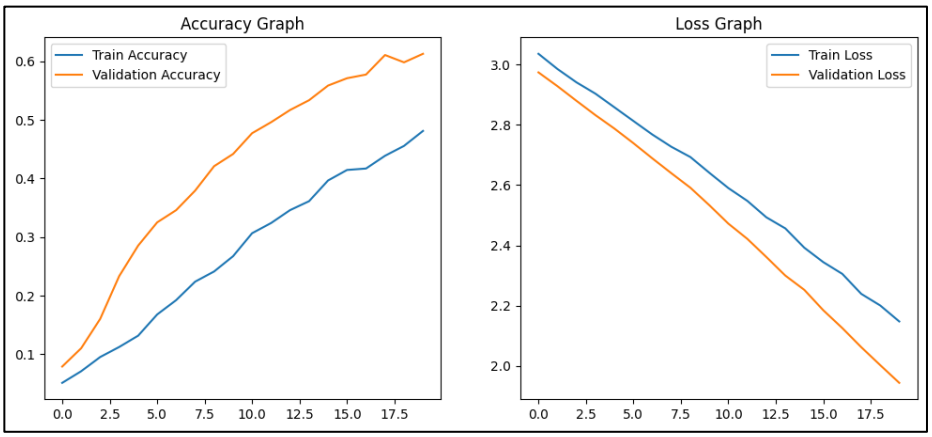



**Gambar 4. 8** *Optimizer RMSprop Learning Rate* 0,00005

Gambar 4.8menunjukkan hasil pelatihan model dengan *optimizer RMSprop* dan *learning rate* 0,00005. Grafik akurasi menunjukkan tren positif yang stabil, di mana akurasi pelatihan meningkat hingga mendekati 95%, dan akurasi validasi tetap tinggi di sekitar 90%, tanpa penurunan berarti. Sementara itu, grafik *loss* juga menunjukkan penurunan konsisten baik pada data pelatihan maupun validasi, bahkan loss validasi sempat lebih rendah dari *loss* pelatihan, yang mengindikasikan generalisasi model yang baik. Dengan *learning rate* yang kecil, proses pelatihan berjalan lebih lambat namun lebih stabil dan minim *overfitting*.

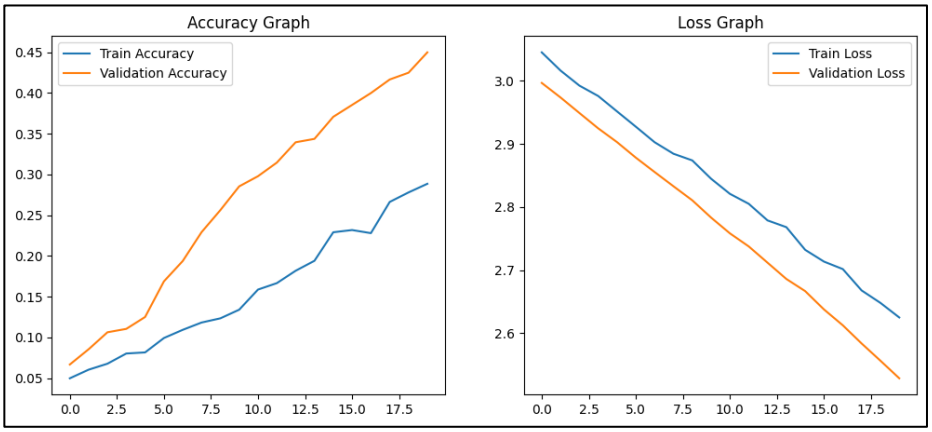
Hasil pelatihan Gambar 4.7 dan Gambar 4.8 dengan *optimizer RMSprop*, hasil terbaik didapatkan dengan *learning rate* 0,0001, yang memberikan akurasi tertinggi sebesar 91,87%. Sementara itu, *learning rate* 0,00005 menghasilkan akurasi sedikit lebih rendah yaitu 90,62%, namun dengan waktu pelatihan yang lebih lama.

3. *Optimizer SGD*




**Gambar 4. 9** *Optimizer SGD Learning Rate* 0,0001

Gambar 4.9 menunjukkan hasil pelatihan model menggunakan *optimizer SGD* dengan *learning rate* 0,0001. Pada grafik akurasi, baik akurasi pelatihan maupun validasi mengalami peningkatan secara bertahap, namun masih tergolong rendah dengan akurasi pelatihan belum mencapai 0,6 hingga akhir *epoch*. Grafik *loss* juga menunjukkan penurunan yang lambat dan linier, menandakan proses pelatihan berlangsung sangat lambat. Hal ini mengindikasikan bahwa *learning rate* yang digunakan terlalu kecil untuk SGD, sehingga model kesulitan melakukan pembaruan bobot secara efektif dan membutuhkan lebih banyak *epoch*.




**Gambar 4. 10** *Optimizer SGD Learning Rate* 0,00005

Gambar 4.10 menunjukkan hasil pelatihan model menggunakan *optimizer SGD* dengan *learning rate* 0,00005. Grafik akurasi menunjukkan peningkatan yang sangat lambat, baik pada data pelatihan maupun validasi, dengan akurasi pelatihan bahkan belum mencapai 0,3 setelah 20 epoch. Demikian pula, grafik *loss* memperlihatkan penurunan yang konsisten namun sangat pelan. Hal ini mengindikasikan bahwa *learning rate* yang digunakan terlalu kecil, sehingga model belajar sangat lambat dan belum mampu mencapai performa optimal dalam jumlah *epoch* yang tersedia. Diperlukan peningkatan learning rate atau jumlah epoch lebih banyak.

Hasil pelatihan Gambar 4.9 dan Gambar 4.10 dengan *optimizer SGD* dengan *momentum* 0,9, pada *learning rate* 0,0001, yaitu sebesar 63,75%. Sementara itu, *learning rate* 0,00005 menghasilkan akurasi yang jauh lebih rendah sebesar 45,21%.

**Tabel 4. 2** Hasil Dari Pelatihan Percobaan

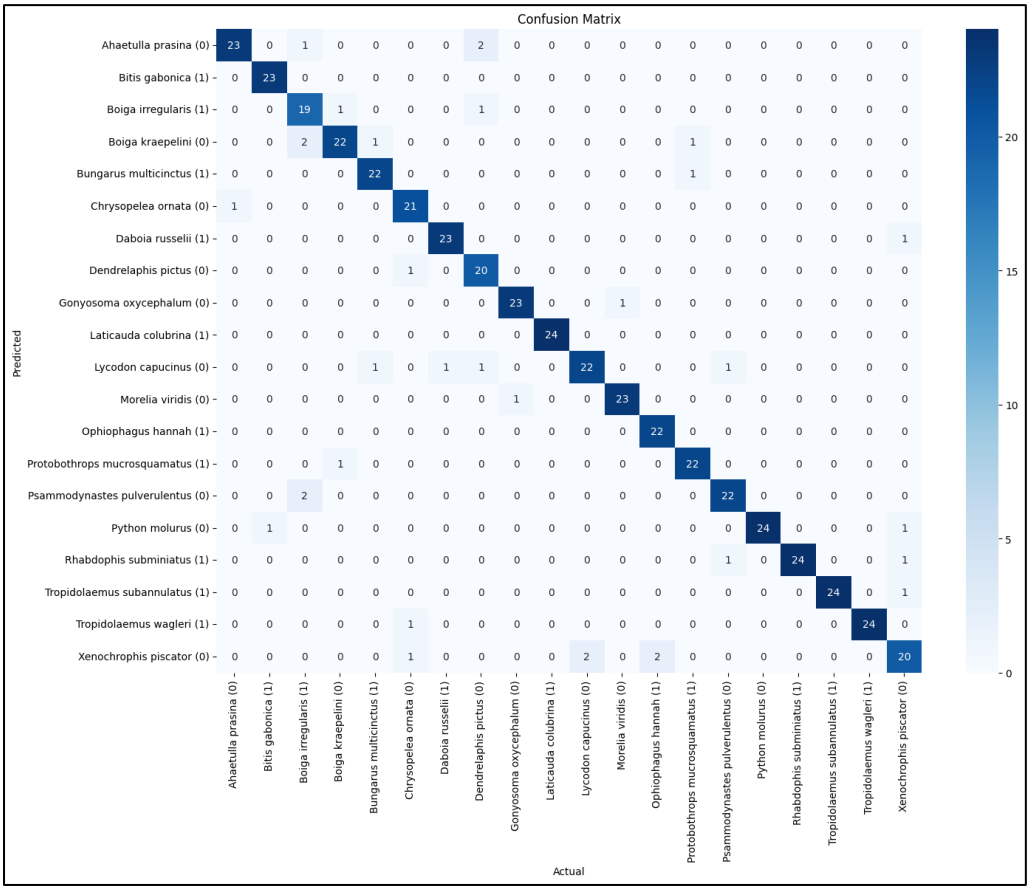
| No | Arsitektur | Optimizer | Parameter 1 Epoch | Parameter 2 Batch Size | Parameter 3 Learning Rate | Hasil Akurasi |
|---|---|---|---|---|---|---|
| 1. | EfficientNetV2-S | Adam | 20 | 32 | 0,0001 | 0,9312 |
| 2. | | | 20 | 32 | 0,00005 | 0,9146 |
| 3. | | RMSprop | 20 | 32 | 0,0001 | 0,9187 |
| 4. | | | 20 | 32 | 0,00005 | 0,9062 |
| 5. | | SGD | 20 | 32 | 0,0001 | 0,6375 |
| 6. | | | 20 | 32 | 0,00005 | 0,4521 |

Dari keseluruhan hasil pelatihan pada percobaan tersebut, didapatkan performa model yang cukup baik dengan akurasi yang terus meningkat seiring bertambahnya *epoch*.

Pelatihan model dengan *optimizer Adam* pada Gambar 4.5 mampu mencapai akurasi *training* hingga 98,12% dan akurasi validasi tertinggi sebesar 91,04%. Selain itu, nilai *loss* pada data *training* dan validasi menunjukkan penurunan yang konsisten, menandakan bahwa model belajar secara efektif tanpa mengalami *overfitting* yang signifikan. Kemudian dari hasil tersebut dilakukan testing dengan data testing dan menghasilkan akurasi sebesar 93,12%. Hasil ini dapat dilihat pada Tabel 4.2 bahwa model memiliki kemampuan generalisasi yang baik terhadap data baru.

### 4.2.4 Hasil Evaluasi

Model yang telah dilatih dan menunjukkan performa yang stabil pada data *training* dan validasi. Pada Gambar 4.11, merupakan hasil *Confusion Matrix* data *testing* yang berjumlah 480 gambar. Setiap spesies memiliki jumlah gambar 24, jika sel diagonal berjumlah 24 yang berwarna biru menunjukkan jumlah prediksi yang benar, yaitu ketika label aktual sama dengan label prediksi.




**Gambar 4. 11** Hasil *Confusion Matrix* data *testing*

Setelah mengetahui hasil *Confusion Matrix* menunjukkan beberapa spesies ular mendapatkan 24 untuk sel diagonal menandakan model tersebut memprediksi benar sesuai kelas ular. Kemudian dilakukan perhitungan evaluasi untuk menilai seberapa baik model dalam mempelajari data ular.

Evaluasi berfungsi untuk mengukur performa model, yang meliputi nilai akurasi (*accuracy*), presisi (*precision*), dan *recall*. Perhitungan evaluasi dilakukan dibawah ini

dalam mencari nilai untuk salah satu kelas *Ahaetulla prasina* untuk melihat seberapa akurat model dalam memprediksi suatu kelas, sebagai berikut :

1. *Precision* untuk kelas *Ahaetulla prasina*

$$Presisi = \frac{23}{23 + 3} = 0{,}8846$$

2. *Recall* untuk kelas *Ahaetulla prasina*

$$Recall = \frac{23}{23 + 1} = 0{,}9583$$

3. *F1 Score* untuk kelas *Ahaetulla prasina*

$$F1\ Score = 2 \times \frac{0{,}88 \times 0{,}96}{0{,}88 + 0{,}96} = 0{,}919$$

4. Akurasi untuk 20 kelas dalam klasifikasi spesies ular

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} = \frac{447}{480} = 0{,}93125$$

Dari hasil dari tabel *confusion matrix* kemudian dilakukan perhitungan untuk mengetahui nilai akurasi, *presisi, recall* dan *F1score* dengan membandingkan nilai prediksi dengan label 20 kelas yang dilatih. Hasil dari perhitungan tersebut diperlihatkan pada Gambar 4.12 seperti berikut:

Akurasi per kelas:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Ahaetulla prasina (0) | 0.88 | 0.96 | 0.92 | 24 |
| Bitis gabonica (1) | 1.00 | 0.96 | 0.98 | 24 |
| Boiga irregularis (1) | 0.90 | 0.79 | 0.84 | 24 |
| Boiga kraepelini (0) | 0.85 | 0.92 | 0.88 | 24 |
| Bungarus multicinctus (1) | 0.96 | 0.92 | 0.94 | 24 |
| Chrysopelea ornata (0) | 0.95 | 0.88 | 0.91 | 24 |
| Daboia russelii (1) | 0.96 | 0.96 | 0.96 | 24 |
| Dendrelaphis pictus (0) | 0.95 | 0.83 | 0.89 | 24 |
| Gonyosoma oxycephalum (0) | 0.96 | 0.96 | 0.96 | 24 |
| Laticauda colubrina (1) | 1.00 | 1.00 | 1.00 | 24 |
| Lycodon capucinus (0) | 0.85 | 0.92 | 0.88 | 24 |
| Morelia viridis (0) | 0.96 | 0.96 | 0.96 | 24 |
| Ophiophagus hannah (1) | 1.00 | 0.92 | 0.96 | 24 |
| Protobothrops mucrosquamatus (1) | 0.96 | 0.92 | 0.94 | 24 |
| Psammodynastes pulverulentus (0) | 0.92 | 0.92 | 0.92 | 24 |
| Python molurus (0) | 0.92 | 1.00 | 0.96 | 24 |
| Rhabdophis subminiatus (1) | 0.92 | 1.00 | 0.96 | 24 |
| Tropidolaemus subannulatus (1) | 0.96 | 1.00 | 0.98 | 24 |
| Tropidolaemus wagleri (1) | 0.96 | 1.00 | 0.98 | 24 |
| Xenochrophis piscator (0) | 0.80 | 0.83 | 0.82 | 24 |
| accuracy | | | 0.93 | 480 |
| macro avg | 0.93 | 0.93 | 0.93 | 480 |
| weighted avg | 0.93 | 0.93 | 0.93 | 480 |

**Gambar 4. 12** Hasil Perhitungan Evaluasi Model

Berdasarkan hasil perhitungan dari *confusion matrix* pada model yang menggunakan *optimizer Adam* dengan *learning rate* 0,0001, dapat disimpulkan bahwa model bekerja

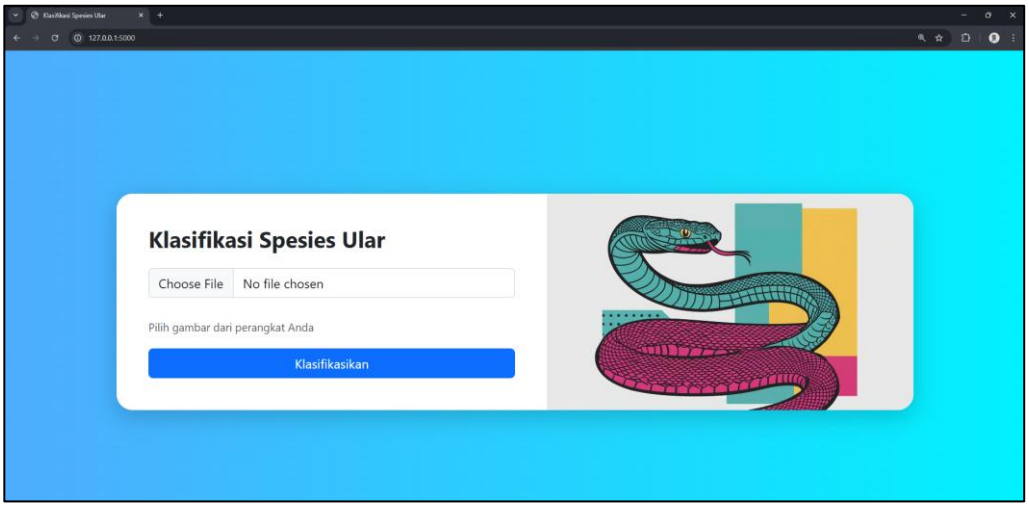
sangat baik dalam mengklasifikasikan 20 kelas spesies ular. Model berhasil mencapai akurasi keseluruhan sebesar 93%, yang mencerminkan tingkat ketepatan prediksi yang tinggi. Nilai rata-rata *precision, recall,* dan *f1-score* juga konsisten di angka 0,93, menandakan keseimbangan antara kemampuan model dalam mengenali kelas yang benar (*recall*) dan menghindari prediksi yang salah (*precision*). Hampir semua kelas memiliki *f1-score* di atas 0,90, bahkan beberapa kelas seperti *Laticuada colubrina*, *Protobothrops mucrosquamatus*, dan *Tropidolaemus subannulatus* menunjukkan performa sempurna. Hasil ini mengindikasikan bahwa model tidak hanya andal secara keseluruhan, tetapi juga mampu mengenali tiap kelas dengan sangat baik, membuatnya layak digunakan untuk tugas klasifikasi ular dengan kompleksitas tinggi.

### 4.2.5 Hasil Implementasi Pengembangan Sistem

Pengembangan sistem terdiri atas beberapa halaman antarmuka pengguna yang dirancang untuk merepresentasikan fungsionalitas utama digunakan untuk memilih gambar spesies ular. Kemudian halaman berikutnya merupakan hasil klasifikasi spesies yang akan menampilkan nama spesies ular.

1. Halaman Beranda

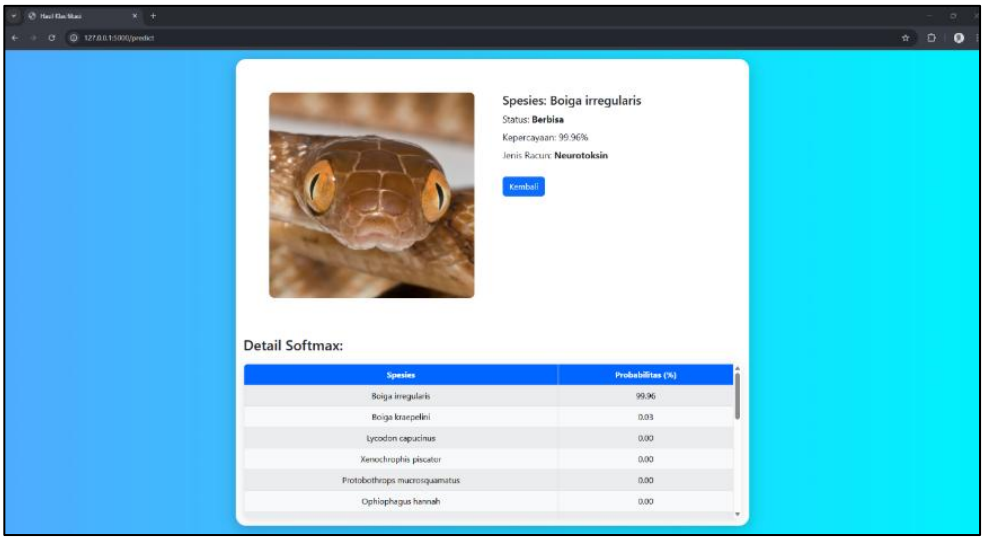
   Pada halaman utama pada Gambar 4.13 *website* akan menampilkan *form* untuk *input* gambar ular yang akan diklasifikasi oleh pengguna.




**Gambar 4. 13** Tampilan *Website* Halaman Utama

2. Halaman Hasil Klasifikasi

   Setelah memasukkan gambar pada halaman utama, selanjutnya akan memunculkan hasil klasifikasi spesies ular. Pada halaman hasil klasifikasi akan menampilkan nama spesies ular, status ular, skor kepercayaan, jenis racun.




**Gambar 4. 14** Tampilan *Website* Hasil Klasifikasi

Hasil dari sistem klasifikasi menghasilkan seperti pada Gambar 4.14, yang menampilkan spesies ular, status ular, skor kepercayaan, jenis racun. Hasil kepercayaan didapatkan dari perhitungan *softmax* seperti yang dihasilkan pada Tabel 3.14 dalam perhitungan *softmax* untuk kemiripan semua spesies ular.

### 4.2.6 Hasil Pengujian Sistem

Pengujian *black-box* dilakukan untuk memastikan bahwa sistem bekerja sesuai dengan fungsionalitas yang diharapkan dari sisi pengguna. Setelah model diintegrasikan dengan *website*. Website tersebut akan dilakukan proses pengujian dengan menggunakan metode black-box. Pengujian aplikasi dilakukan sesuai dengan skenario yang telah direncanakan pada Tabel 3.17. Berikut merupakan hasil dari pengujian website untuk klasifikasi spesies ular oleh .

**Tabel 4. 3** Pengujian Sistem Black-Box

| No | Pengujian | Ekspektasi Hasil | Hasil |
|---|---|---|---|
| 1 | Pengguna dapat mengambil atau mengunggah gambar ular yang telah tersimpan di perangkat mereka. | Pengguna berhasil mengunggah gambar. | Berhasil |
| 2 | Pengguna dapat mengunggah gambar melalui tombol "*Upload Image*" untuk melakukan klasifikasi ular. | Pengguna berhasil mengunggah gambar melalui tombol "*Upload Image*" untuk melakukan klasifikasi ular. | Berhasil |
| 3 | Sistem dapat menampilkan hasil prediksi spesies ular. | Sistem berhasil menampilkan hasil hasil prediksi spesies ular. | Berhasil |

Pada Tabel 4.3 hasil skenario pengujian sistem dengan 3 skenario dengan menggunakan metode *black-box* berhasil menunjukkan bahwa ketiga skenario tersebut berjalan dengan baik saat digunakan oleh pengguna.

## 4.3 Pembahasan

Penelitian ini bertujuan dalam implementasi model *Convolutional Neural Network* dengan arsitektur *EfficientNetV2S* dalam mengklasifikasikan spesies ular, baik yang berbisa

maupun tidak berbisa. Pada penelitian ini bukan membantu dalam pengenalan visual ular, agar dapat membantu proses identifikasi ular di dunia nyata lebih cepat misalnya untuk pertolongan pertama terhadap gigitan ular berbisa, edukasi masyarakat, atau penelitian keanekaragaman hayati. Dalam proses pengembangan model, digunakan dataset dari hasil gabungan dari sumber *Kaggle* dan teknik *web scraping* melalui *website Gbif*, mencakup jumlah total 4.640 gambar dari 20 spesies ular yang berbeda.

Pada tahap pelatihan model, proses dilakukan secara menyeluruh dan sistematis untuk memastikan bahwa model *EfficientNetV2S* tidak hanya mampu mengenali pola visual dari gambar ular, tetapi juga dapat menggeneralisasi ke data baru yang belum pernah dilihat sebelumnya. Salah satu pendekatan penting yang digunakan dalam tahap ini adalah penerapan teknik *augmentasi citra*, yang berfungsi untuk memperkaya variasi data latih agar model tidak hanya belajar dari pola yang sama saja. Dengan melakukan beberapa proses *augmentasi* yaitu *rotation range, width, height, shear range, zoom range, horizontal flip,* serta perubahan ukuran citra menjadi 384 x 384 piksel dan normalisasi piksel (*rescale*). terhadap gambar asli untuk menghasilkan versi baru yang berbeda pada data latih agar model lebih banyak belajar variasi gambar ular dalam menghindari *overvitting*. Teknik ini mempresentasikan dunia nyata, gambar yang diambil dari alam liar tentu tidak selalu ideal. Selain itu, untuk lebih memaksimalkan kinerja model, dilakukan juga *fine-tuning* terhadap bobot *pretrained* dari *EfficientNetV2S* yang sudah dilatih sebelumnya pada dataset besar *ImageNet*. Dengan penerapan *transfer learning* yang tidak memerlukan model belajar dari awal lagi, sehingga model hanya menerapkan pemahaman awal yang sudah dimiliki kemudian menyesuaikan terhadap karakteristik dari dataset ular yang digunakan dalam pelatihan pada penelitian. Hasil dari penerapan ini menunjukkan performa yang sangat baik. Model *EfficientNetV2S Optimizer Adam* dengan *learning rate* 0,0001 mampu mempelajari pola visual spesies ular dengan cukup baik.

Hasil model ditunjukkan mencapai akurasi keseluruhan sebesar 93%, yang mencerminkan tingkat ketepatan prediksi yang tinggi. Nilai rata-rata *precision, recall,* dan *f1-score* juga konsisten rata-rata di angka 0,93. Grafik akurasi dan *loss* selama proses pelatihan juga mendukung kekuatan performa model. Akurasi validasi tercatat stabil di atas 89%, sementara nilai *loss* terus menurun, menunjukkan bahwa model belajar dengan baik tanpa mengalami *overfitting* yang signifikan. Hasil ini dapat disimpulkan bahwa sebagian besar gambar uji berhasil diklasifikasikan dengan benar. Dari hasil *Confusion matrix* dapat dilihat, di mana spesies seperti *Bitis gabonica*, *Chrysopelea ornata*, dan *Ophiophagus hannah* hampir tidak mengalami kesalahan klasifikasi karena memiliki karakteristik visual yang kuat dan mudah dikenali oleh model. Meskipun dari beberapa spesies masih terdapat jenis spesies yang sering tertukar, seperti *Boiga irregularis* dan *Boiga kraepelini*, yang memiliki pola tubuh dan warna yang sangat mirip. Membedakan kedua spesies ini butuh pengamatan yang sangat teliti dan itu tidak selalu mudah jika hanya berdasarkan citra ular. Beberapa spesies ular tersebut memang sulit dibedakan secara langsung dengan mata manusia jika ular memiliki spesies yang sama dengan bentuk atau karakteristik yang sama.

Dari hasil penelitian ini, dapat disimpulkan bahwa penerapan arsitektur *EfficientNetV2S* telah berhasil membangun sistem identifikasi spesies ular yang dapat diandalkan secara otomatis dan akurat. Penelitian ini tidak hanya berhasil menerapkan

*Convolutional Neural Network* (CNN) dengan arsitektur *EfficientNetV2S* untuk membedakan ular berbisa dan tidak berbisa, tetapi juga membuktikan bahwa model ini mampu mencapai performa tinggi dalam klasifikasi berdasarkan dataset yang tersedia. Dengan *optimizer Adam* mendapatkan akurasi tertinggi 93,12% dan nilai loss yang terus menurun, sistem menunjukkan kemampuan belajar yang efektif tanpa indikasi *overfitting* yang signifikan. Selain itu, serangkaian pengujian *black-box* dilakukan untuk memastikan fungsionalitas sistem berjalan sesuai harapan. Pengguna dapat mengunggah gambar, hingga proses *uploud* dan memunculkan hasil klasifikasi ular. Meskipun masih terdapat beberapa tantangan terutama dalam membedakan spesies yang mirip secara visual, penelitian ini telah menunjukkan bahwa teknologi *deep learning* dapat menjadi solusi nyata dan adaptif dalam mendukung konservasi, edukasi, maupun keselamatan publik yang berkaitan dengan keberadaan ular di lingkungan sekitar.

# BAB V
# PENUTUP

## 5.1 Kesimpulan

Penelitian ini menerapkan model *Convolutional Neural Network* (CNN) dengan arsitektur *EfficientNetV2S* untuk mengklasifikasikan spesies ular, baik yang berbisa maupun tidak berbisa. Dataset yang digunakan berasal dari gabungan dua sumber, yaitu *Kaggle* dan *web scraping* dari *Gbif*, yang secara keseluruhan mencakup 4.586 gambar dari 20 spesies ular berbeda. Hasil dari penelitian dapat disimpulkan sebagai berikut:

1. Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dapat disimpulkan bahwa arsitektur *EfficientNetV2S* memiliki keunggulan dalam melakukan klasifikasi *multiclass* citra spesies ular, terutama pada aspek akurasi yang tinggi, kemampuan generalisasi yang baik, ditunjukkan bahwa model belajar dengan baik tanpa mengalami *overfitting* yang signifikan. *EfficientNetV2S* dirancang dengan prinsip *compound scaling*, yang memungkinkan model memiliki performa tinggi dengan jumlah parameter yang relatif lebih sedikit dibandingkan arsitektur lain. Hal ini membuat proses pelatihan menjadi lebih cepat dan efisien tanpa mengorbankan akurasi. Arsitektur juga memiliki kelemahan, seperti penggunaan memori GPU yang cukup tinggi ketika diterapkan pada resolusi gambar besar. Oleh karena itu, efisiensi arsitektur ini sangat bergantung pada pengelolaan data yang baik dan lingkungan komputasi yang memadai.
2. Hasil evaluasi akhir menunjukkan performa model yang sangat baik, dengan akurasi pengujian menggunakan *optimizer adam* dan *learning rate* 0,0001 mencapai 93,12%. Dengan menggunakan teknik augmentasi digunakan untuk mempebanyak variasi seperti *rotation range, width, height, shear range, zoom range, horizontal flip* untuk mendapatkan variasi dari citra ular diberbagai kondisi. Beberapa spesies seperti *Bitis gabonica*, *Chrysopelea ornata*, dan *Ophiophagus hannah* terklasifikasi dengan baik dapat dengan mudah diprediksi oleh model. Namun, terdapat tantangan dalam membedakan spesies yang memiliki kemiripan tinggi seperti spesies dengan *family* ular yang sama secara morfologis, seperti *Boiga irregularis* dan *Boiga kraepelini*. Secara keseluruhan, hasil evaluasi ini menunjukkan bahwa model memiliki potensi yang kuat dalam klasifikasi spesies ular. Evaluasi menggunakan metrik seperti *precision, recall*, dan *F1-score* memperkuat hasil bahwa model bekerja secara seimbang di seluruh kelas. Meskipun model masih bingung terutama dalam membedakan spesies dengan kemiripan morfologis tinggi, kinerja model secara umum sudah sangat baik untuk diterapkan dalam sistem identifikasi secara langsung.

## 5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa hal yang dapat menjadi ditingkatkan untuk pengembangan lebih lanjut. Pertama, memperbanyak jumlah dataset ular untuk meningkatkan kemampuan generalisasi model, sehingga model dapat lebih fokus dalam mengenali karakteristik morfologi ular seperti pola kulit, bentuk kepala, jenis gigi, dan ciri fisik lainnya yang bersifat khas dan diskriminatif. Dengan menambah

jumlah spesies dan menyertakan gambar dari berbagai kondisi lingkungan serta geografis, agar model dapat menjadi lebih andal dalam mengenali ular dari berbagai pose yang berbeda-beda. Kedua, hasil penelitian dengan menerapkan *EfficientNetV2S* telah memberikan hasil yang sangat baik, penerapan lanjutan dengan arsitektur yang lebih kompleks seperti *EfficientNetV2M* atau *EfficientNetV2L* dapa dicoba karena kapasitasnya yang lebih besar dapat membantu dalam membedakan spesies dengan kemiripan visual yang tinggi. Ketiga, pengembangan sistem tidak hanya terbatas pada klasifikasi, namun juga dapat ditambahkan untuk deteksi objek menggunakan model seperti *YOLO*. Pendekatan ini memungkinkan sistem untuk mengenali dan menandai lokasi ular secara otomatis dalam gambar atau video, yang sangat bermanfaat untuk aplikasi *real-time* seperti pada pemantauan satwa liar atau sistem peringatan dini bagi masyarakat.

# DAFTAR PUSTAKA


Ahmed, K., Gad, M. A., & Aboutabl, A. E. (2024). Snake species classification using deep learning techniques. In *Multimedia Tools and Applications* (Vol. 83, Issue 12). Springer US. https://doi.org/10.1007/s11042-023-16773-0

Aldakhil, L. A., & Almutairi, A. A. (2024). Multi-Fruit Classification and Grading Using a Same-Domain Transfer Learning Approach. *IEEE Access*, *12*(April), 44960–44971. https://doi.org/10.1109/ACCESS.2024.3379276

Anwarul, S., & Tanwar, R. (2025). Venomify: Automated Classification of Venomous and Non-Venomous Snake Species Using Deep Learning. *Procedia Computer Science*, *259*, 219–229. https://doi.org/10.1016/j.procs.2025.03.323

Bloch, L., & Friedrich, C. M. (2021). EfficientNets and vision transformers for snake species identification using image and location information FHDO biomedical computer science group (BCSG). *CEUR Workshop Proceedings*, *2936*, 1477–1498.

BRI. (2023). *Fakta menarik dibalik reputasi ular*. https://www.rri.co.id/lain-lain/829312/fakta-menarik-dibalik-reputasi-ular

BRIN. (2023). *Mengapa jumlah ular di Indonesia banyak? Ini penjelasan peneliti BRIN*. https://www.brin.go.id/news/110275/mengapa-jumlah-ular-di-indonesia-banyak-ini-penjelasan-peneliti-brin

Dafa, M. H., & Suyanto, S. (2021). Kasus Gigitan Ular di Indonesia. *Jurnal Pengabdian Masyarakat MIPA Dan Pendidikan MIPA*, *5*(1), 47–52. https://doi.org/10.21831/jpmmp.v5i1.29343

Fathoni, M. (2019). *Keanekaragaman dan Persebaran Jenis Ular (Squamata: Serpentes) di Kawasan Malang Raya*.

Garbin, C., Zhu, X., & Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, *79*(19–20), 12777–12815. https://doi.org/10.1007/s11042-019-08453-9

Gustiawan, A., Wahyudi, J., & Suryana, E. (2023). Perancangan Aplikasi Steganografi Pada Citra Digital Menggunakan Metode Pixel Value Differencing. *JUKI: Jurnal Komputer Dan Infromatika*, *5*, 151–163.

Huang, Z., Jiang, X., Huang, S., Qin, S., & Yang, S. (2023). An efficient convolutional neural network-based diagnosis system for citrus fruit diseases. *Frontiers in Genetics*, *14*(August), 1–11. https://doi.org/10.3389/fgene.2023.1253934

Iguernane, M., Ouzziki, M., Es-Saady, Y., El Hajji, M., Lansari, A., & Bouazza, A. (2025). Deep Learning-Based Snake Species Identification for Enhanced Snakebite Management. *AI (Switzerland)*, *6*(2), 1–16. https://doi.org/10.3390/ai6020021

Indrawani. (2020). Menggunakan Metode Convolutional Neural Network. *Jurnal Informatika Dan Sistem Informasi (JIFoSI)*, *1*(1), 3331–3338.

Jeremia Bu'ulölö, G., Jacobus, A., & Kambey, F. D. (2021). Identification of Cataract Eye Disease Using Convolutional Neural Network. *Jurnal Teknik Informatika*, *16*(1), 375–382.

Kalati, M. A., Shabani, H., Maghareh, M. S., Barzegar, Z., & Lashgari, R. (2025). *Classi cation and Interpretation of Histopathology Images : Leveraging Ensemble of E cientNetV1 and E cientNetV2 Models Classification and Interpretation of Histopathology Images : Leveraging Ensemble of EfficientNetV1 and*. 0–23.

Kasus, S., Penyakit, K., Rakhmat, G. A., & Taufikurohman, F. (2023). *Evaluasi Ensemble Stacking Arsitektur*. 1–13.

Khaeriyah, 2019. (2021). Implementasi Metode Convolutional Neural Network Menggunakan Tensorflow Dalam Mendeteksi Sebuah Objek. *Implementasi Metode Convolutional Neural Network Menggunakan Tensorflow Dalam Mendeteksi Sebuah Objek*, 7–26.

Khairunnas, K., Yuniarno, E. M., & Zaini, A. (2021). Pembuatan Modul Deteksi Objek Manusia Menggunakan Metode YOLO untuk Mobile Robot. *Jurnal Teknik ITS*, *10*(1). https://doi.org/10.12962/j23373539.v10i1.61622

Kurniati, K. (2021). Penerapan Metode Prototype Pada Perancangan Sistem Pengarsipan Dokumen Kantor Kecamatan Lais. *Journal of Software Engineering Ampera*, *2*(1), 16–27. https://doi.org/10.51519/journalsea.v2i1.89

Li, X., Li, X., Pan, D., & Zhu, D. (2020). On the learning property of logistic and softmax losses for deep neural networks. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, *9*, 4739–4746. https://doi.org/10.1609/aaai.v34i04.5907

Marida, W., & Radhi, M. (2019). Perilaku Satwa Liar Pada Kelas Mamalia. *Jurnal OSF Preprints*, 1–10.

Muhammadiyah Mataram, U., Hafidzah, P., Maryani, S., Yuliatin Ihsani, B., Kurniamala Niswariyana, A., & Bahasa, P. (2024). *Seminar Nasional Paedagoria Penerapan Deep Learning dalam Menganalisis Sentimen di Media Sosial*. *4*, 328–339.

Nggego, D. A., Hasbi, M., & Patawaran, N. (2025). *Data augmentation and transfer learning efficientnetv2-s on rice leaf disease classification Data augmentation and transfer learning efficientnetv2-s on rice leaf disease classification*. https://doi.org/10.1088/1755-1315/1454/1/012001

Pandangan Jogja. (2021). *Mematikannya Gigitan Ular di Indonesia di Masa Pandemi, Jangan Lakukan Hal Bodoh*. https://kumparan.com/pandangan-jogja/mematikannya-gigitan-ular-di-indonesia-di-masa-pandemi-jangan-lakukan-hal-bodoh-1wBsLPZVAQX

Pantur, V. H., Sukarno, A., & Zairina, A. (2024). Keanekaragaman Spesies Ular Di Resort Rowo Bendo Taman Nasional Alas Purwo. *Journal of Scientech Research and Development*, *6*(1), 56–64. https://doi.org/10.56670/jsrd.v6i1.293

Plateau, I., Java, E., Sidik, I., Sumitro, S. B., & Kurniawan, N. (2018). *The Linnaeus ' s Reed*

*Snake , Calamaria linnaei Boie ( Squamata : Colubridae : 5*(1), 42–53.

Qisthan, A. H. (2023). Analisis performa metode convolutional neural network dengan arsitektur convnext dalam klasifikasi spesies ular berbisa dan tidak berbisa di indonesia. *Repository.Uinjkt.Ac.Id*. https://repository.uinjkt.ac.id/dspace/handle/123456789/77034%0Ahttps://repository.uinjkt.ac.id/dspace/bitstream/123456789/77034/1/ALIFIAR HAZAZI QISTHAN-FST.pdf

Rakhmat, G. A., & Rizkiawarman, F. (2023). *Implementasi Arsitektur MobilenetV3 ( Studi Kasus Klasifikasi Jamur Beracun )*. *3*, 1–14.

Saifullah, S. (2020). *Enhancement Dalam Proses Segmenasi Citra Untuk Deteksi Fertilitas Telur*. *9*, 134–145.

Salasta, R. I., Informatika, P. S., Informatika, J., & Industri, F. T. (2024). *Penerapan Metode Convolutional Neural Network dengan Arsitektur ResNet-50 untuk Pengenalan Ekspresi Wajah*.

Salehin, I., & Kang, D. K. (2023). A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain. *Electronics (Switzerland)*, *12*(14). https://doi.org/10.3390/electronics12143106

Sarıgül, M., Ozyildirim, B. M., & Avci, M. (2019). Differential convolutional neural network. *Neural Networks*, *116*, 279–287. https://doi.org/10.1016/j.neunet.2019.04.025

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0197-0

Sidik, D. P., Utaminingrum, F., & Muflikhah, L. (2023). Penggunaan Variasi Model pada Arsitektur EfficientNetV2 untuk Prediksi Sel Kanker Serviks. *... Teknologi Informasi Dan Ilmu ...*, *7*(5), 2116–2121. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/12656

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 10691–10700.

Teng, H., Kang, S., Wang, Z., & Yin, L. (2023). Research on Structural Damage Identification Based on Multiple Model Deep ResNet. *World Earthquake Engineering*, *39*(3), 68–74. https://doi.org/10.19994/j.cnki.WEE.2023.0053

Umri, B. K., Utami, E., & Kurniawan, M. P. (2021). menggunakan Convolutional Neural Networks Systematic Literature Review of Detection Covid-19 using Convolutional Nerual Networks. *Citec Journal*, *8*(1).

Wintoko, R., & Prameswari, N. P. (2020). Manajemen Gigitan Ular Update Management of Snake Bite. *JK Unila*, *4*(1), 49.

Xu, C., Coen-Pirani, P., & Jiang, X. (2023). Empirical Study of Overfitting in Deep Learning

for Predicting Breast Cancer Metastasis. *Cancers*, *15*(7), 1–14. https://doi.org/10.3390/cancers15071969

Yenusi, Y. N., Suryasatriya Trihandaru, & Setiawan, A. (2023). Comparison of Convolutional Neural Network (CNN) Models in Face Classification of Papuan and Other Ethnicities. *JST (Jurnal Sains Dan Teknologi)*, *12*(1), 261–268. https://doi.org/10.23887/jstundiksha.v12i1.46861

Yuniari, P. N. W., Kumara, I. M. S., Agus, I. K. W. R., Bhaskara, I. M. A., Dana, G. W. P., & Darma, I. G. W. (2024). *Analisis Klasifikasi Citra Penokohan Topeng Bali*. *13*(02).