KERJA PRAKTIK - IF184801

Perbandingan Kinerja Model Deep Learning untuk Klasifikasi Citra Silang Serasi

Departemen Teknik Informatika ITS
Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60117
Periode: 1 Februari 2025 - 30 Juni 2025

Oleh:

Rakha Fathin Izzan Consetta 5025221156

Pembimbing Jurusan
Ilham Gurat Adillion, S.Kom., M.Eng.
Pembimbing Lapangan
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

KERJA PRAKTIK – IF184801

Perbandingan Kinerja Model Deep Learning untuk Klasifikasi Citra Silang Serasi

Departemen Teknik Informatika ITS
Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60117
Periode: 1 Februari 2025 - 30 Juni 2025

Oleh:

Rakha Fathin Izzan Consetta 5025221156

Pembimbing Jurusan
Ilham Gurat Adillion, S.Kom., M.Eng.

Pembimbing Lapangan
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

Perbandingan Kinerja Model Deep Learning untuk Klasifikasi Citra Silang Serasi

Oleh:

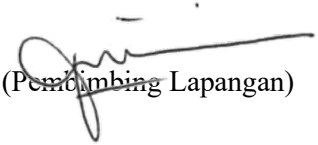
Rakha Fathin Izzan Consetta 5025221156

Disetujui oleh Pembimbing Kerja Praktik:

1. Ilham Gurat Adillion, S.Kom., M.Eng.
   NIP. 1995202411009

(Pembimbing Departemen)

2. Dini Adni Navastara, S.Kom., M.Sc.
   NIP. 198510172015042001



(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Perbandingan Kinerja Model Deep Learning untuk Klasifikasi Citra Silang Serasi

Nama Mahasiswa	: Rakha Fathin Izzan Consetta
NRP	: 5025221156
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Ilham Gurat Adillion, S.Kom., M.Eng.
Pembimbing Lapangan	: Dini Adni Navastara, S.Kom., M.Sc.

ABSTRAK

Efisiensi dan akurasi dalam pelayanan transfusi darah merupakan aspek vital, di mana uji silang serasi memegang peranan kunci. Proses manual yang bergantung pada interpretasi visual seringkali menjadi kendala karena bersifat subjektif dan lambat. Untuk mengatasi hal tersebut, otomatisasi menggunakan *deep learning* menawarkan solusi yang menjanjikan. Laporan Kerja Praktik ini berfokus pada perbandingan kinerja secara sistematis terhadap lima arsitektur model *pre-trained* modern yakni RegNetY-3.2GF, Swin Transformer V2, ConvNeXt V2 Base, EfficientNetV2 Small, dan Vision Transformer untuk tugas klasifikasi citra uji silang serasi yang telah disegmentasi.

Metodologi yang diterapkan meliputi pendekatan *transfer learning* dengan teknik pembekuan parameter dan dievaluasi melalui tiga skenario pengujian: (1) pengujian dasar dengan hyperparameter seragam, (2) validasi silang K-Fold (K-Fold Cross-Validation) untuk mengukur stabilitas, dan (3) optimasi *hyperparameter* menggunakan Optuna untuk menemukan performa puncak.

Hasil pengujian menunjukkan bahwa model berbasis *Transformer* (SwinV2 dan ViT) memiliki performa *baseline* yang sangat kuat. Namun, temuan paling signifikan adalah dampak dramatis dari

optimasi *hyperparameter*, di mana model seperti ConvNeXt V2 dan terutama EfficientNetV2 yang pada awalnya gagal (F1-Score 0.5067 dan 0.3378), berhasil mencapai performa yang sangat kompetitif (F1-Score 0.9535 dan 0.8509 secara berurutan) setelah dioptimalkan. Analisis stabilitas melalui K-Fold mengonfirmasi bahwa Swin Transformer V2 tidak hanya konsisten mencapai performa tertinggi dengan F1-Score 0.9677 setelah optimasi, tetapi juga menunjukkan standar deviasi terendah, membuktikan keandalannya yang superior.

Kerja praktik ini menyimpulkan bahwa Swin Transformer V2 merupakan model paling optimal untuk tugas ini karena menawarkan kombinasi terbaik antara akurasi, stabilitas, dan efisiensi. Selain itu, hasil studi ini menegaskan bahwa evaluasi model yang komprehensif, mencakup tuning dan analisis stabilitas, adalah krusial untuk pemilihan model yang andal di lingkungan praktis.

Kata Kunci: Pelayanan Transfusi Darah, Uji Silang Serasi, Klasifikasi Citra, Deep Learning, Transfer Learning

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS yaitu Kerja Praktik yang berjudul: Perbandingan Kinerja Model Deep Learning untuk Klasifikasi Citra Silang Serasi.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung antara lain:

1. Kedua orang tua penulis.
2. Ilham Gurat Adillion, S.Kom, M.Eng. selaku dosen pembimbing kerja praktik sekaligus koordinator kerja praktik.
3. Ibu Dini Adni Navastara, S.Kom, M.Sc selaku pembimbing lapangan selama kerja praktik berlangsung.
4. Teman-teman penulis yang senantiasa memberikan semangat ketika penulis melaksanakan KP.

Surabaya, 9 Mei 2025
Rakha Fathin Izzan Consetta

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pelayanan transfusi darah merupakan bagian penting dalam sistem layanan kesehatan yang menuntut kecepatan dan akurasi tinggi. Salah satu prosedur wajib dalam proses ini adalah uji silang serasi (crossmatch), yaitu pengujian pratransfusi untuk mendeteksi kemungkinan reaksi antigen-antibodi antara darah donor dan resipien. Proses ini biasanya dilakukan secara manual melalui interpretasi visual terhadap citra hasil uji mikroskopis, yang bersifat subjektif dan bergantung pada keahlian tenaga medis. Kondisi ini dapat menimbulkan risiko kesalahan interpretasi serta keterlambatan dalam pengambilan keputusan, terlebih ketika jumlah tenaga kesehatan tidak memadai atau dalam situasi darurat. Menurut data WHO (2023), sistem transfusi darah yang efisien dan tepat waktu dapat menyelamatkan jutaan nyawa setiap tahunnya, sehingga otomatisasi proses klasifikasi hasil uji silang serasi menjadi urgensi tersendiri.

Dalam beberapa tahun terakhir, pendekatan Deep Learning, khususnya metode transfer learning dengan arsitektur model pra-latih seperti InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, dan InceptionResNetV2, telah menunjukkan performa yang menjanjikan dalam klasifikasi citra medis. Penelitian sebelumnya juga telah berhasil membangun pipeline klasifikasi uji silang serasi yang mencakup proses preprocessing, augmentasi, segmentasi menggunakan U-Net berbasis VGG19, hingga klasifikasi. Dari beberapa model yang diuji, arsitektur EfficientNetB7 menunjukkan performa terbaik dengan nilai precision sebesar 0.9461, recall sebesar 0.9300, *f1-score* sebesar 0.9374, dan akurasi sebesar 0.9236 pada data uji. Hasil ini membuktikan bahwa pemanfaatan AI dalam domain ini berpotensi besar untuk mempercepat dan meningkatkan kualitas layanan transfusi darah.

Meskipun demikian, terdapat celah yang belum diteliti lebih lanjut, yaitu perbandingan dan evaluasi menyeluruh antar model klasifikasi terkini dari sisi akurasi, efisiensi komputasi, dan kelayakan implementasi di lapangan. Selain itu, diperlukan pendekatan sistematis untuk memilih model terbaik yang tidak hanya akurat, namun juga ringan, cepat, dan andal dalam konteks integrasi ke sistem aplikasi riil. Oleh karena itu, kerja praktik ini difokuskan pada eksplorasi dan pengembangan model klasifikasi citra hasil uji silang serasi berbasis Deep Learning, melalui studi komparatif terhadap beberapa arsitektur terkini guna mendukung percepatan otomatisasi proses transfusi darah yang lebih efektif.

1.2. Tujuan

Tujuan kerja praktik ini adalah untuk memenuhi kewajiban mata kuliah Kerja Praktik sebesar 2 SKS dan membantu tenaga kesehatan atau unit layanan transfusi darah dalam mengatasi kendala penerapan otomatisasi analisis uji silang serasi, melalui pengembangan dan pemilihan model klasifikasi uji silang serasi yang akan diintegrasikan pada sebuah aplikasi web pada server riset ITS

1.3. Manfaat

Pelaksanaan kerja praktik ini diharapkan memberikan manfaat dalam mendukung otomatisasi klasifikasi hasil uji silang serasi menggunakan model deep learning yang akurat dan efisien, sehingga mampu mempercepat proses pelayanan transfusi darah dan mengurangi risiko kesalahan klasifikasi manual. Selain itu, kegiatan ini melanjutkan dan memperkuat penelitian terdahulu yang berfokus pada segmentasi dan klasifikasi citra silang serasi, serta menambah kontribusi akademik dengan mengevaluasi dan membandingkan arsitektur model deep learning terbaru. Model klasifikasi yang dikembangkan juga diintegrasikan dalam aplikasi Atomics, sebuah sistem terintegrasi yang bertujuan mempermudah

implementasi teknologi AI di fasilitas kesehatan, sehingga mendukung efisiensi dan akurasi pelayanan laboratorium transfusi darah di Indonesia.

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana proses preprocessing dan preparasi data citra hasil uji silang serasi sebelum digunakan dalam pelatihan model klasifikasi?
2. Bagaimana proses pengembangan dan pelatihan model klasifikasi citra hasil uji silang serasi menggunakan beberapa arsitektur deep learning terkini (RegNetY, SwinV2, EfficientNetV2, ViT Base, dan ConvNeXt V2)?
3. Bagaimana performa masing-masing model klasifikasi dalam mengenali hasil uji silang serasi ditinjau dari metrik akurasi, precision, recall, dan *f1-score*?

1.5. Lokasi dan Waktu Kerja Praktik

Pengerjaan kerja praktik ini dilakukan secara *remote*. Adapun kerja praktik dimulai pada tanggal 1 Februari 2025 hingga 30 Juni 2025.

1.6. Metodologi Kerja Praktik

Metodologi dalam pembuatan buku kerja praktik meliputi :

1.6.1. Perumusan Masalah

Untuk merumuskan permasalahan yang relevan, penulis mendalami latar belakang kebutuhan otomatisasi klasifikasi hasil uji silang serasi dalam layanan transfusi darah. Rumusan masalah difokuskan pada peningkatan efisiensi dan akurasi klasifikasi citra berbasis deep learning.

1.6.2. Studi Literatur

Penulis melakukan studi literatur terhadap penelitian terdahulu yang berkaitan dengan klasifikasi citra medis menggunakan deep learning, khususnya pendekatan transfer learning dan penggunaan model pretrained seperti RegNetY, SwinV2, EfficientNetV2, ViT Base, dan ConvNeXt V2. Literatur juga mencakup teori preprocessing, augmentasi, serta metrik evaluasi performa model.

1.6.3. Pengumpulan dan Persiapan Dataset

Dataset citra hasil uji silang serasi diperoleh dari hasil segmentasi yang telah dilakukan pada penelitian sebelumnya. Dataset dibersihkan, diberi label, dilakukan augmentasi, dan dibagi menjadi trainset dan testset dengan proporsi tertentu untuk mempersiapkan data sebelum pelatihan model.

1.6.4. Pemilihan dan Implementasi Model Deep Learning

Beberapa arsitektur pretrained model dipilih dan diimplementasikan menggunakan framework Torch. Implementasi mencakup modifikasi pada layer akhir, fine-tuning, dan penggunaan konfigurasi hyperparameter yang sesuai.

1.6.5. Pelatihan dan Evaluasi Model

Model-model yang telah diimplementasikan dilatih menggunakan dataset yang telah dipersiapkan. Performa masing-masing model dievaluasi menggunakan metrik akurasi, precision, recall, dan *f1-score* untuk menentukan model paling optimal. Model juga dievaluasi kecepatan untuk melakukan klasifikasinya.

1.6.6. Kesimpulan dan Saran

Setelah evaluasi model dilakukan, ditarik kesimpulan mengenai performa model terbaik serta saran pengembangan lebih lanjut, baik dari sisi model maupun integrasi ke sistem aplikasi.

1.7. Sistematika Laporan

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II Profil Perusahaan

Bab ini berisi gambaran umum instansi tempat kerja praktik dilaksanakan, termasuk struktur organisasi dan bidang riset yang relevan.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi teori-teori dasar dan referensi penelitian terdahulu yang menjadi landasan dalam pelaksanaan kerja praktik.

1.7.4. Bab IV Pengumpulan dan Persiapan Dataset

Bab ini menjelaskan proses akuisisi, preprocessing, labeling, dan augmentasi data yang digunakan untuk pelatihan model klasifikasi.

1.7.5. Bab V Pemilihan dan Implementasi Model Deep Learning

Bab ini menjelaskan proses pemilihan arsitektur model, implementasi teknis, serta setup pelatihan termasuk konfigurasi hyperparameter.

1.7.6. Bab VI Pelatihan dan Evaluasi Model

Bab ini memuat hasil pelatihan setiap model, analisis performa berdasarkan metrik evaluasi, serta perbandingan antar model.

1.7.7. Bab VII Kesimpulan dan Saran

Bab ini berisi ringkasan hasil kerja praktik serta saran untuk pengembangan model atau sistem lebih lanjut.

[Halaman ini sengaja dikosongkan]

BAB II PROFIL PERUSAHAAN

2.1. Profil Lokasi Kerja Praktik

Melalui instruksi pemerintah pada tahun 1985 untuk membuka Program Studi S1 baru untuk bidang ilmu teknologi komputer, ITS mendirikan program bernama Program Studi Teknik Komputer. Namun sejak tahun 1993, nama Program Studi Teknik Komputer diubah menjadi Jurusan Teknik Komputer. Akhirnya, pada tahun 1996 secara resmi jurusan ini berganti nama menjadi Jurusan Teknik Informatika berdasarkan Surat Keputusan Direktur Jendral Pendidikan Tinggi Nomor 224/DIKTI/Kep/1996, tanggal 11 Juli 1996. Pada saat ini, Jurusan Teknik Informatika memperoleh nilai akreditasi A berdasarkan Surat Keputusan Badan Akreditasi Nasional Perguruan Tinggi (BAN-PT) Nomor 003/BAN-PT/Ak-X/S1/V/2006, tanggal 18 Mei 2006.

Selain program Sarjana (S1), Jurusan Teknik Informatika juga menyelenggarakan program Pasca Sarjana (S2) yang dirintis sejak tahun 1994, dengan surat keputusan Direktur Jendral Pendidikan Tinggi No. 2851/D/T/2001, perihal ijin penyelenggaraan Program-Program Studi Jenjang Program Strata-2 (S2) pada Institut Teknologi Sepuluh Nopember Surabaya. Dan pada tahun 2011, Jurusan Teknik Informatika mulai menyelenggarakan program Doktor (S3)

2.2. Lokasi Kerja Praktik

Sesuai dengan visi ITS yaitu menjadi perguruan tinggi dengan reputasi internasional dalam ilmu pengetahuan, teknologi, dan seni, terutama yang menunjang industri dan kelautan yang berwawasan lingkungan, maka visi Departemen Teknik Informatika adalah menjadi inovator bidang informatika yang unggul di tingkat nasional dengan reputasi internasional, serta

berperan aktif dalam upaya memajukan dan mensejahterakan bangsa

2.3. Misi Lokasi Kerja Praktik

Departemen Teknik Informatika memiliki misi sebagai berikut:

1. Menyelenggarakan proses pembelajaran yang berkualitas, dan memenuhi standar nasional maupun internasional,
2. Melaksanakan penelitian yang inovatif, bermutu, dan bermanfaat,
3. Meningkatkan pemanfaatan teknologi informasi dan komunikasi untuk masyarakat,
4. Menjalin kemitraan dengan berbagai lembaga, baik di dalam maupun di luar negeri.

2.4. Fasilitas Lokasi Kerja Praktik

1. Laboratorium Rekayasa Perangkat Lunak

Di Laboratorium ini ditawarkan bidang minat yang berfokus pada keahlian melakukan pengujian perangkat lunak, Kemampuan mengelola proyek perangkat lunak, Kemampuan mengurangi resiko kesalahan perangkat lunak, dan Kemampuan membuat perangkat lunak game.

2. Laboratorium Komputasi Berbasis Jaringan

Di Laboratorium ini ditawarkan bidang keahlian yang ditekankan pada Kemampuan lulusan sarjana/magister/doktor dalam membangun infrastruktur jaringan yang aman, kemampuan membangun sistem grid, Kemampuan membangun aplikasi jaringan sesuai Standard dan Kemampuan membangun aplikasi multimedia berbasis jaringan.

3. Laboratorium Komputasi Cerdas dan Visi

Di Laboratorium ini ditawarkan bidang keahlian yang ditekankan pada kemampuan lulusan dalam memanipulasi dan

menganalisis data citra pada berbagai bidang aplikasi (a.l. biomedika, industri), kemampuan menerapkan metode sistem cerdas pada berbagai bidang aplikasi dan kemampuan memodelkan dan mengoptimasikan sistem nyata.

4. **Laboratorium Teknologi Jaringan dan Keamanan Siber Cerdas**

Laboratorium di bidang minat ini menawarkan bidang keahlian yang ditekankan pada Kemampuan lulusan dalam membangun berbagai macam arsitektur jaringan sesuai standar teknologi terkini dan menerapkan keamanan jaringan.

5. **Laboratorium Grafika, Interaksi, Gim dan Analitik**

Laboratorium ini di bidang minat ini menawarkan bidang keahlian yang ditekankan pada kemampuan lulusan dalam mendesain, mengembangkan dan mendokumentasikan proses pembuatan game sesuai dengan standar. Serta membuat model 3 dimensi dan pemograman di dalam realitas virtual serta aplikasi realitas virtual 3 dimensi dengan menggunakan game engine.

6. **Laboratorium Algoritma dan Pemrograman**

Di laboratorium ini, mahasiswa dilatih untuk merancang dan menganalisa algoritma serta mengimplementasikannya dalam bahasa pemrograman yang sesuai, guna menyelesaikan berbagai permasalahan secara efisien.

7. **Laboratorium Manajemen Cerdas Informasi**

Laboratorium di bidang minat ini menawarkan bidang keahlian yang ditekankan pada kemampuan lulusan dalam menganalisis, mensintesa dan mengevaluasi proses bisnis dan sistem informasi pada sistem Enterprise, mengimplementasikan rekayasa pengetahuan ke dalam suatu aplikasi, melakukan investigasi, pengujian, evaluasi kematangan dan kepatutan terhadap prosedur standard dan tata kelola teknologi informasi, melakukan tata kelola proyek dan sumber daya manusia dan merancang dan mengimplementasikan solusi basis data terdistribusi dan teknologi Big Data.

8. **Laboratorium Pemodelan dan Komputasi Terapan**

Laboratorium ini mewadahi riset dan kerjasama industri di bidang pemodelan & simulasi, peramalan sains, optimasi, serta komputasional saintifik.

2.5. Alamat Lokasi Kerja Praktik

Jalan Teknik Kimia - Gedung Departemen Teknik Informatika, Kampus Institut Teknologi Sepuluh Nopember, Sukolilo, Surabaya, Jawa Timur

BAB III
TINJAUAN PUSTAKA

3.1. Klasifikasi Citra Digital

Gambar atau citra merupakan salah satu cara untuk menyimpan informasi visual seperti warna, kedalaman, jarak, dan kecerahan lebih efisien daripada menyimpan data dengan cara tabular. Maka dari itu, sangat diperlukan cara untuk mengambil informasi tersebut dan membuat prediksi berdasarkannya. Klasifikasi citra digital adalah proses mengambil data dari gambar untuk memberi suatu label atau kelas pada gambar tersebut. Pentingnya klasifikasi citra digital dalam visi komputer telah meningkat secara eksponensial, utamanya karena penerapannya yang sangat luas, seperti deteksi objek, kualitas kontrol, diagnosa medis, dan analisis video.

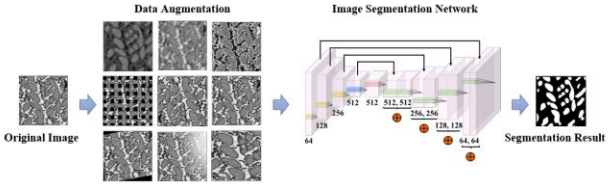
3.2. Uji Silang Serasi

Uji silang serasi merupakan pengujian yang dilakukan dengan mereaksikan silang invitro antara darah resipien dan darah pendonor. Reaksi ini dilakukan supaya mencegah reaksi hemolitik saat darah didonorkan dan supaya darah yang ditransfusikan itu memang bermanfaat bagi kesehatan resipien. Uji silang serasi dapat dilakukan dengan dua metode, yaitu metode gel dan metode tabung. Pada proyek kali ini, metode yang dilakukan adalah metode *gel*. Metode *gel test* merupakan metode untuk mendeteksi reaksi sel darah merah dengan antibodi. Metode ini lebih cepat dan lebih akurat daripada metode tabung.

3.3. *Image Augmentation* untuk Klasifikasi

Augmentasi gambar atau dikenal sebagai *Image enhancement* merupakan suatu metode untuk meningkatkan

jumlah dan keragaman data yang terbatas. Berbeda dari metode lain untuk mencegah *overfitting* seperti *pretraining*, *dropout*, *batch normalization*, dan *transfer learning*, augmentasi gambar memulai dari akar permasalahan *overfitting*, yaitu kurangnya sampel data, dan mencegah *overfitting* dengan meniru perubahan yang mungkin terjadi pada data nyata melalui perhitungan matematis.



Gambar 3.1 Contoh Proses Augmentasi Gambar

3.4. *Image Segmentation*

Segmentasi citra merupakan proses membagi sebuah citra atau gambar menjadi berbagai daerah atau segmen untuk mengidentifikasi objek atau *Region of Interest (ROI)*. Tujuan dari segmentasi citra adalah menyederhanakan penggambaran dari suatu citra menjadi bagian yang jelas dan berarti. Pengaplikasian segmentasi sangat beragam, dari pengenalan objek, manipulasi citra, dan analisis citra medis. Dalam segmentasi, proses dimulai dari pengambilan citra awal, diikuti dengan urutan pra pemrosesan seperti, mengubah ukuran, augmentasi data, *scaling*, dan *cropping*. Lalu, *feature extraction* dan metode segmentasi citra diaplikasikan. Hasil akhir dari proses ini adalah citra yang telah tersegmentasi.

3.5. Arsitektur Model Transfer Learning

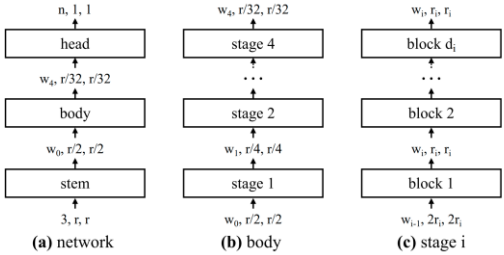
Semenjak awal mulanya era *Machine Learning (ML)*, *transfer learning* menjadi suatu hal yang suka dieksplor para ilmuwan. Sebelum naiknya model *deep learning*, *transfer learning* dikenal sebagai *domain adaptation* dan fokus kepada data homogen serta cara menghubungkan data tersebut ke satu sama

lain karena sifat algoritma ML. Metode *transfer learning* terkini bertujuan untuk mengurangi waktu dan biaya pada proses *training*, serta keperluan dataset yang banyak yang sulit didapat seperti di area medis.

3.5.1. **RegNetY-3.2GF**

Ruang desain RegNet mengadopsi arsitektur hierarkis modular yang umum dalam pengenalan visual. Jaringan ini secara fundamental terbagi menjadi tiga komponen utama: 'stem' untuk pemrosesan input awal, 'body' yang melakukan sebagian besar komputasi dan transformasi fitur, serta 'head' untuk menghasilkan prediksi akhir.

Bagian 'body' sendiri tersusun dari serangkaian 'stage' berurutan, di mana setiap stage dirancang untuk memproses fitur pada resolusi spasial yang semakin berkurang sambil menyesuaikan dimensi *channel*. Lebih lanjut, setiap 'stage' terdiri dari beberapa 'block', dengan blok pertama bertanggung jawab untuk *downsampling* resolusi dan penyesuaian channel, sementara blok-blok berikutnya menjaga dimensi fitur tetap konsisten dalam stage tersebut.

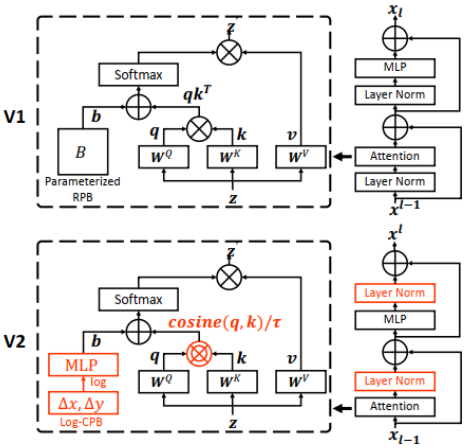


Gambar 3.2 Gambaran Umum Struktur Jaringan Ruang Desain RegNet

3.5.2. **Swin Transformer V2**

Swin Transformer V2 adalah kelanjutan dari arsitektur Swin Transformer yang menggabungkan konsep *hierarchical vision transformers* dengan efisiensi komputasi tinggi menggunakan *shifted windows*. Versi V2 memperkenalkan beberapa perbaikan signifikan, seperti skema normalisasi yang stabil untuk pelatihan dengan batch besar, dan metode inisialisasi parameter yang lebih baik.

Swin Transformer V2 Small adalah varian ringan dari arsitektur ini yang tetap mempertahankan kinerja tinggi sambil menekan kompleksitas. Berdasarkan hasil eksperimen, model ini menghasilkan top-1 accuracy sebesar 84.5% di ImageNet-1K, menjadikannya pilihan unggul untuk tugas klasifikasi citra dengan sumber daya terbatas .

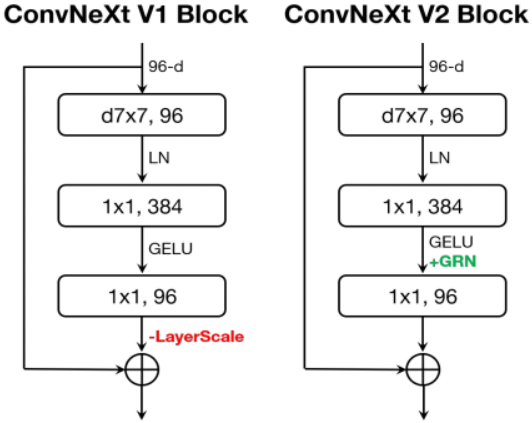


Gambar 3.3 Perbandingan Swin V1 dan V2 dengan Adaptasinya

3.5.3. **ConvNeXt V2 Base**

ConvNeXt V2 merupakan pengembangan dari arsitektur ConvNeXt yang memodernisasi CNN (Convolutional Neural

Networks) agar mampu bersaing dengan Vision Transformer (ViT). ConvNeXt V2 memperkenalkan *Fully Convolutional Masked Autoencoder* (FCMAE) dan *Global Response Normalization* (GRN) sebagai dua inovasi utama. FCMAE mengadaptasi pendekatan *masked autoencoding* dari ViT ke dalam CNN, memungkinkan representasi visual yang lebih efisien dan padat tanpa harus menggunakan arsitektur transformer. Sementara GRN membantu mencegah fenomena *feature collapse* dengan meningkatkan kompetisi antar *channel*, sehingga mendukung diversifikasi representasi fitur.

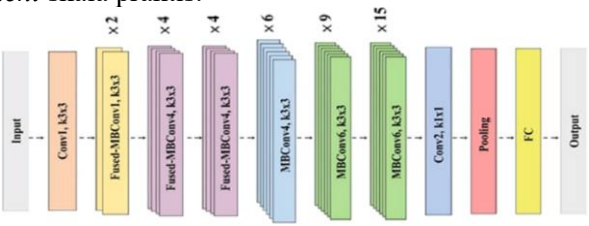


Gambar 3.3 Desain Blok ConvNeXt V1 dan V2

3.5.4. **EfficientNetV2 Small**

EfficientNetV2 Small adalah anggota keluarga EfficientNet generasi kedua yang mengutamakan efisiensi parameter dan kecepatan training. Arsitektur ini diperoleh melalui proses *training-aware neural architecture search* (NAS) dan teknik *progressive learning*. Fitur utama dari EfficientNetV2 adalah penggunaan Fused-MBConv pada lapisan awal, yang menggantikan *depthwise convolution* demi efisiensi GPU/TPU, serta adaptasi regularisasi dinamis sesuai ukuran input dalam

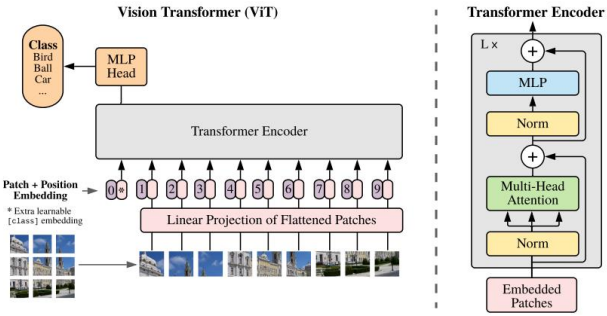
pelatihan progresif. Model ini mencapai akurasi tinggi pada ImageNet-1K dengan waktu pelatihan jauh lebih singkat dibanding model *transformer* seperti ViT, menjadikannya pilihan tepat untuk *deployment* skala praktis.



Gambar 3.5 Arsitektur EfficientNetV2

3.5.5. **Vision Transformer**

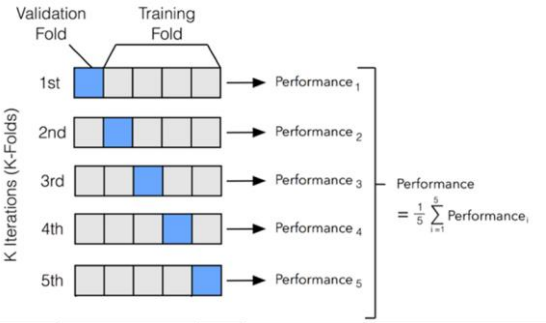
Vision Transformer (ViT) merupakan arsitektur *transformer* murni pertama yang diterapkan untuk visi komputer. Alih-alih menggunakan konvolusi, ViT membagi citra menjadi patch berukuran tetap (misalnya 16×16), lalu memproyeksikan patch tersebut ke dalam *embedding* linear yang diproses seperti token dalam NLP. Keunggulan ViT adalah kemampuannya belajar representasi global secara langsung dan mendalam. ViT Base Hybrid yang digunakan dalam proyek ini adalah hasil pretraining dari *Google Research* menggunakan ImageNet-21K, menjadikannya kuat dalam *transfer learning*. Meski akurasinya tinggi, model ini memiliki beban komputasi lebih berat dibanding arsitektur CNN efisien seperti EfficientNet atau ConvNeXt.



Gambar 3.6 Arsitektur Vision Transformer

3.6. K-Fold Cross Validation

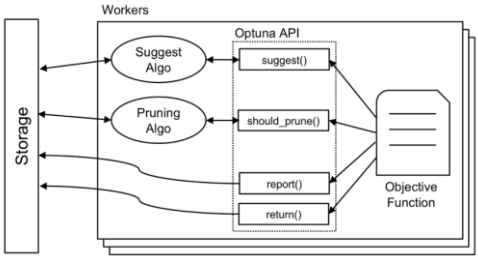
K-Fold Cross Validation adalah metode evaluasi model statistik dan *machine learning* yang bertujuan untuk mengukur performa model secara lebih andal. Dalam teknik ini, dataset dibagi menjadi k bagian atau lipatan yang ukurannya sama. Proses pelatihan dan pengujian dilakukan sebanyak k kali, di mana pada setiap iterasi, satu lipatan digunakan sebagai data uji (*testing set*) dan sisanya sebagai data latih *(training set*). Hasil evaluasi dari seluruh iterasi tersebut kemudian dirata-rata untuk mendapatkan estimasi performa model secara menyeluruh.



Gambar 3.7 Cara Kerja K-Fold Cross Validation

3.7. Optuna untuk Hyperparamter Tuning

Optuna merupakan *framework next-generation* untuk optimasi *hyperparameter* yang menerapkan pendekatan *define-by-run*, artinya ruang pencarian *hyperparameter* dapat dibentuk secara dinamis pada saat runtime. *Framework* ini menyediakan API modular seperti suggest(), report(), dan should_prune() yang digunakan untuk eksplorasi dan evaluasi *hyperparameter* dalam fungsi objektif. Arsitektur dasar dari Optuna seperti ditunjukkan pada Gambar di atas, memperlihatkan interaksi antara fungsi objektif, API Optuna, serta algoritma *suggestion* dan *pruning*, semuanya dikoordinasikan melalui *storage* bersama untuk memungkinkan pencarian dan validasi paralel



Gambar 3.8 Gambaran Umum Desain Sistem Optuna

3.8. Metode Evaluasi Model

Evaluasi model merupakan langkah yang sangat penting dalam klasifikasi. *true positive (TP)* dan *true negative (TN)* adalah jumlah data yang diklasifikasikan dengan benar oleh model, kelas positif maupun negatif. Sedangkan *false positive (FP)* dan *false*

negative (FN) adalah jumlah data yang diklasifikasikan dengan salah,kelas positif maupun negatif. Pada klasifikasi gambar, metode evaluasi model yang sering digunakan adalah *accuracy, precision, recall, f1-score,* dan *confusion matrix*.

Accuracy didapatkan dari jumlah prediksi yang benar dibagi dengan jumlah data dalam dataset. *Recall* didapatkan dari jumlah prediksi positif yang benar dibagi dengan total positif. *Precision* didapatkan dari jumlah prediksi positif yang benar dibagi dengan total prediksi positif. *f1-score* adalah pengukuran akurasi yang dihitung berdasarkan *precision* dan *recall*. Confusion matrix merupakan matrix untuk mendeskripsikan hasil prediksi dengan mencantumkan jumlah kelas terprediksi dan kelas sebenarnya

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

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

$$Precision = \frac{TP}{(TP + FP)}$$

$$F1\ Score = \frac{2 \times precision \times recall}{precision + recall}$$

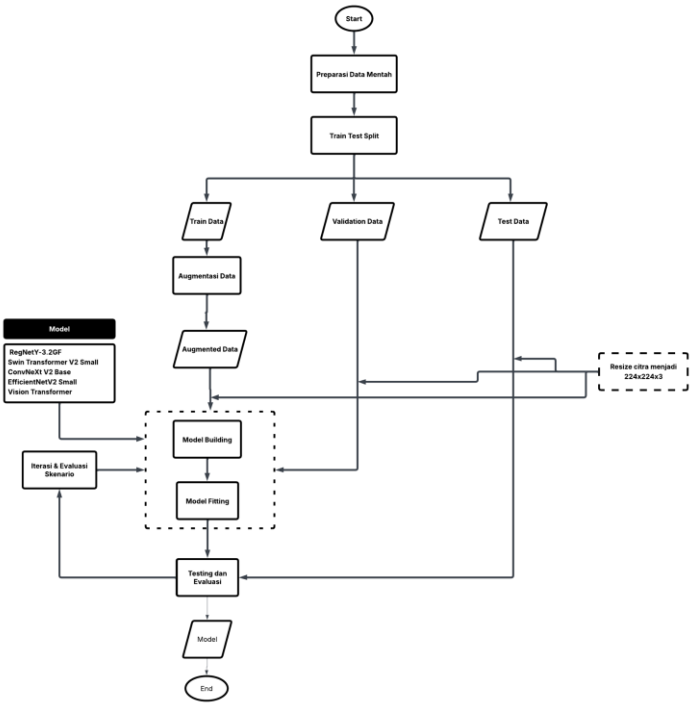
Class designation		Actual class	
		True (1)	False (0)
Predicted class	Positive (1)	TP	FP
	Negative (0)	FN	TN

Gambar 3.9 Contoh Confusion Matrix

BAB IV
ANALISIS DAN PERANCANGAN INFRASTRUKTUR SISTEM

4.1. Analisis dan Alur Sistem

Pada bab ini akan dijelaskan mengenai tahapan analisis sistem yang dilakukan. Sistem yang dikembangkan bertujuan untuk melakukan klasifikasi citra darah menggunakan metode transfer learning dengan model pretrained dari *library* timm. Alur sistem secara umum sebagai berikut:



Gambar 4.1. Diagram Alur Klasifikasi

Diagram alur sistem pada Gambar 4.1 menggambarkan tahapan proses klasifikasi hasil uji silang serasi berbasis deep learning yang dilakukan selama kerja praktik. Proses diawali dari tahap preparasi data mentah, di mana citra hasil uji silang serasi dikumpulkan dan disiapkan dalam format yang sesuai. Setelah itu dilakukan proses *train-test split* menjadi tiga bagian utama: data *training*, *validation*, dan *testing*.

Pada data pelatihan, dilakukan proses augmentasi data untuk meningkatkan keragaman citra, yaitu *flip horizontal* dan penyesuaian *brightness*/*contrast*. Seluruh data kemudian melalui proses resize ke dimensi standar model *pre-trained* yaitu 224×224×3 piksel. Tahap selanjutnya adalah pembangunan dan pelatihan model (*model building* dan *fitting*) menggunakan lima arsitektur *pre-trained* yang berbeda, yaitu RegNetY-3.2GF, Swin Transformer V2 Small, ConvNeXt V2 Base, EfficientNetV2 Small, dan Vision Transformer. Masing-masing model dilatih menggunakan data train dan divalidasi menggunakan data validasi.

Setelah pelatihan selesai, sistem memasuki tahap pengujian dan evaluasi model terhadap data uji, yang mencakup perhitungan metrik evaluasi seperti akurasi, precision, recall, dan *f1-score*. Tahap ini juga mencakup iterasi dan evaluasi ulang terhadap model jika diperlukan. Model terbaik disimpan untuk digunakan dalam sistem aplikasi Atomics.

4.2. Peralatan Pendukung

Seluruh proses pelatihan dan evaluasi model klasifikasi dilakukan menggunakan platform Kaggle Notebooks yang menyediakan dukungan komputasi GPU secara gratis. Pada eksperimen ini, digunakan konfigurasi akselerator GPU NVIDIA Tesla T4 (2x) yang memungkinkan pelatihan model berarsitektur besar secara efisien.

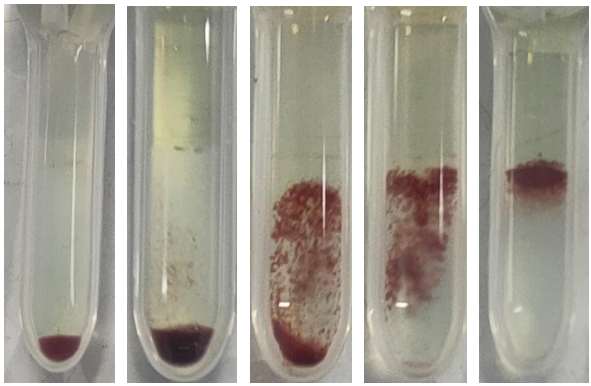
Dataset citra uji silang serasi disimpan dalam Google Drive dan diakses secara langsung melalui mounting ke Kaggle Notebook. Hal ini mempermudah proses sinkronisasi dan

pembaruan dataset tanpa perlu mengunggah ulang file ke penyimpanan lokal. Bahasa pemrograman yang digunakan yaitu Python. Framework pemrograman yang digunakan adalah PyTorch, dengan dukungan *library* tambahan *timm* untuk pemanggilan pretrained model, *torchvision* untuk transformasi citra, serta *scikit-learn* untuk evaluasi metrik performa. Kombinasi antara komputasi berbasis cloud dan framework deep learning ini memungkinkan proses kerja praktik dilakukan secara efisien tanpa keterbatasan perangkat keras lokal.

4.3. Dataset dan Preprocessing

Dataset yang digunakan merupakan dataset yang dihasilkan oleh segmentasi pada penelitian Navastara et al. (2024). Dataset pada penelitian tersebut diambil dari hasil crossmatch di bank darah RSUD Dr. Soetomo. Dataset tersebut berisi gambar reaksi darah donor terhadap sel penerima dalam tabung yang diambil menggunakan *smartphone*. Gambar harus diambil dengan pencahayaan yang cukup dan setiap tabung harus terlihat jelas. Dataset ini memiliki beberapa kelas (*multiclass*), yaitu: Negatif, Positif 1, Positif 2, Positif 3, dan Positif 4. Dataset awal terdiri dari 131 gambar, di mana setiap gambar berisi sekelompok tabung dalam satu kantong. Dari gambar-gambar tersebut, dilakukan pemotongan untuk setiap tabung guna mendapatkan *Region of Interest* (RoI), sehingga menghasilkan 719 gambar.

Gambar 4.2. Satu gambar mentah dari dataset



(a) (b) (c) (d) (e)

Gambar 4.3. Gambar yang sudah dipotong untuk setiap tabung
(a). Negatif, (b). Positif 1, (c). Positif 2, (d) Positif 3, (e). Positif 4

Beberapa contoh data yang dihasilkan oleh segmentasi pada penelitian Navastara et al. (2024) adalah sebagai berikut

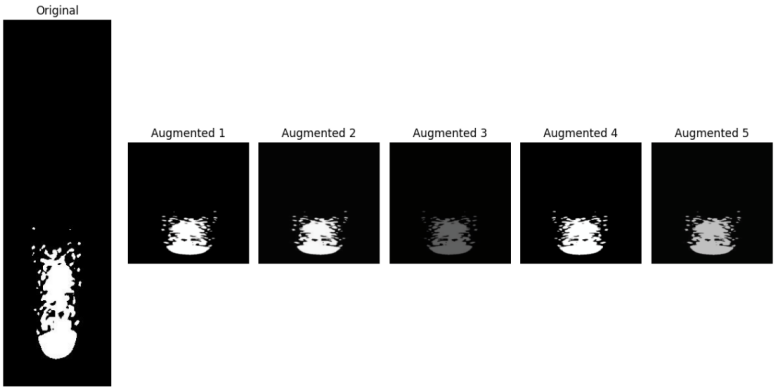
(a) (b) (c) (d) (e)

Gambar 4.4. Hasil Segmentasi dari Penelitian Terdahulu
(a). Negatif, (b). Positif 1, (c). Positif 2, (d) Positif 3, (e). Positif 4

Berdasarkan hasil segmentasi, akan dilakukan klasifikasi untuk mengklasifikasikan dan memprediksi label dari gambar tersebut. Gambar-gambar yang dapat dilihat pada Gambar 4.4 akan digunakan untuk melatih model deep learning. Kemudian data akan dibagi menjadi data pelatihan, data pengujian, dan data validasi. Data pelatihan akan digunakan untuk melatih model. Data validasi akan digunakan untuk memvalidasi model selama proses pelatihan. Data pengujian akan digunakan sebagai data yang belum pernah dilihat untuk mengevaluasi model setelah proses pelatihan selesai. Persentase pembagian data adalah 80% untuk pelatihan, 10% untuk validasi, dan 10% untuk pengujian. Persentase ini akan diterapkan secara konsisten pada semua kelas.

Sebelum melakukan pelatihan, beberapa tahap praproses akan dilakukan pada gambar. Semua gambar akan diubah ukurannya menjadi 224x224x3 piksel untuk memastikan dimensi yang konsisten. Ukuran ini dipilih karena didasarkan pada bobot

model pralatih (pretrained) yang dilatih pada gambar dengan dimensi 224x224x3. Normalisasi akan diterapkan pada gambar dengan menggunakan nilai rata-rata ImageNet (0.485, 0.456, 0.406) dan standar deviasi (0.229, 0.224, 0.225). Augmentasi yang diterapkan merupakan *random horizontal flip* dan *color jitter*, yaitu dengan memodifikasi *brightness* dan *contrast* nya. Alasan tidak dilakukan *vertical flip* adalah karena dapat mengakibatkan kebingungan dan ambigu pada gambar dengan label yang sebenarnya



Gambar 4.5. Hasil *Preprocess* dan Augmentasi Gambar

4.4. Perancangan Model

Perancangan model pada penelitian ini difokuskan untuk membangun sistem klasifikasi citra hasil uji silang serasi yang akurat dan andal. Pendekatan utama yang digunakan adalah transfer learning, yang bertujuan untuk membangun model yang robust dengan memanfaatkan pengetahuan dari model-model yang telah dilatih sebelumnya pada dataset berskala besar.

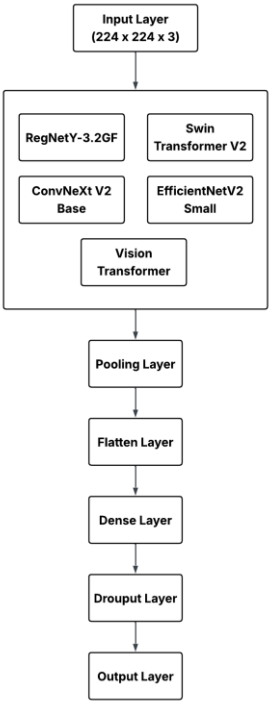
Model-model pretrained yang digunakan dalam eksperimen ini terdiri dari RegNetY-3.2GF, Swin Transformer V2 Small, ConvNeXt V2 Base, EfficientNetV2 Small, dan Vision

Model	**Tahun Rilis**	***Size* (MB)**	***Parameters***	***Activations***	***Top-1 Accuracy***	***Top-5 Accuracy***
RegNetY-3.2GF	2023	78.1	19.4M	11.3M	0.818	0.958
Swin Transformer V2	2023	199	49.7M	50.3M	0.832	0.965
ConvNeXt V2 Base	2023	355	88.7M	28.8M	0.849	0.971
EfficientNet V2 Small	2022	96.5	23.9M	21.4M	0.839	0.968
Vision Transformer	2022	392	97.9M	27.9M	0.780	0.940

Transformer (ViT) Hybrid. Pemilihan kelima model ini didasarkan pada performa tinggi mereka dalam tugas klasifikasi citra umum maupun medis, serta kemampuan generalisasi yang baik pada dataset yang terbatas.

Dalam kerja praktik ini, seluruh model diinisialisasi menggunakan bobot pelatihan awal dari ImageNet melalui *library* timm, kemudian dimodifikasi pada bagian layer akhirnya untuk menyesuaikan jumlah kelas klasifikasi uji silang serasi. Setiap model diperlakukan sebagai feature extractor utama, dan hanya bagian akhir atau head klasifikasinya saja yang dilatih ulang. Hal ini dicapai melalui teknik pembekuan parameter (*freezing layer*), di mana seluruh parameter awal dari *backbone pretrained* disetel menjadi tidak dapat dilatih. Selanjutnya, hanya layer akhir model yang diaktifkan kembali untuk proses fine-tuning. Strategi ini dilakukan untuk mempertahankan pengetahuan umum yang telah dipelajari model dari dataset skala besar, sekaligus menyesuaikannya dengan domain citra medis yang lebih spesifik. Dengan pendekatan ini, diharapkan model mampu mencapai akurasi tinggi tanpa kehilangan efisiensi dan generalisasi

Tabel 4.1. Keterangan dan struktur model yang digunakan



Gambar 4.6. Arsitektur Klasifikasi Secara Umum

Selain lebih baru dibanding model-model yang digunakan pada penelitian sebelumnya, berikut alasan dipilihnya pretrained model pada kerja praktik ini

1. **RegNetY-3.2GF:** mengadopsi arsitektur hierarkis modular yang terbukti efektif dalam pengenalan visual.
2. **Swin Transformer V2 Small:** menawarkan efisiensi tinggi melalui mekanisme shifted windows, cocok untuk skenario terbatas sumber daya.

3. **ConvNeXt V2 Base:** mewakili CNN modern dengan inovasi seperti FCMAE dan GRN, yang mendukung representasi visual yang stabil dan beragam.
4. **EfficientNetV2 Small** mengutamakan efisiensi dan kecepatan pelatihan berkat Fused-MBConv dan desain arsitektur yang dioptimalkan.
5. **Vision Transformer (ViT) Hybrid** mampu menangkap representasi global secara efektif dan unggul dalam transfer learning melalui pretraining skala besar.

Demikian mengenai keterangan secara umum pretrained model dan strukturnya. Gambar 4.6. menunjukkan arsitektur setiap model yang disederhanakan. Keterangan detail setiap model *pretrained* terdapat pada tabel 4.1.

4.5. Skenario Pengujian

Dalam kerja praktik ini, dilakukan tiga skenario pengujian yang bertujuan mengevaluasi pengaruh jenis data masukan dan strategi pelatihan terhadap performa klasifikasi model deep learning. Setiap skenario dilakukan terhadap kelima model pretrained yang sama

1. **Skenario Dasar (*Segmented Images*):** Skenario dasar menggunakan dataset citra hasil segmentasi, di mana hanya area yang mengandung aglutinasi atau sedimen darah (sebagai indikasi reaksi silang serasi) yang disorot. Citra hasil segmentasi bersifat biner (putih-hitam), seperti ditunjukkan pada Gambar 4.2. Model dilatih dengan pendekatan dasar. Tujuan dari skenario ini adalah menilai performa klasifikasi model pada citra yang telah difokuskan hanya pada bagian relevan (*foreground*). Hal ini juga bertujuan mengurangi *noise* dari latar belakang tabung.
2. ***K-Fold Cross-Validation (Stratified)***: Skenario kedua bertujuan untuk mengevaluasi konsistensi dan generalisasi model melalui teknik Stratified K-Fold Cross-Validation.

Pendekatan ini membagi dataset menjadi k subset (fold), di mana setiap subset digunakan secara bergiliran sebagai data uji, sementara sisanya sebagai data latih. Dalam penelitian ini digunakan 5-fold cross-validation, memastikan proporsi kelas tetap seimbang di setiap fold.

Setiap model pre-trained diuji pada seluruh fold, dan metrik performa seperti akurasi, presisi, recall, *f1-score*, serta waktu pelatihan dicatat untuk setiap iterasi. Strategi ini memungkinkan evaluasi yang lebih menyeluruh terhadap stabilitas performa model pada variasi data yang berbeda-beda, serta mengurangi ketergantungan pada satu subset data tertentu.

3. **Optuna *Hyperparameter Tuning*:** Skenario ini difokuskan pada optimalisasi hyperparameter secara individual untuk setiap arsitektur model, dengan tujuan menemukan konfigurasi parameter pelatihan yang paling optimal guna memaksimalkan performa klasifikasi. Metode yang digunakan adalah tuning berbasis Bayesian Optimization yang diimplementasikan melalui library Optuna. Untuk setiap model, ruang pencarian hyperparameter didefinisikan mencakup learning rate (dari 1e-5 hingga 1e-1, dalam skala logaritmik), weight decay (dari 1e-6 hingga 1e-1, dalam skala logaritmik), dan pilihan optimizer (Adam atau AdamW).

   Proses tuning dilakukan dengan menjalankan sejumlah 20 percobaan per model, di mana Optuna secara cerdas menjelajahi ruang hyperparameter dan mengevaluasi kinerja model pada set validasi. Mekanisme pruning (pemangkasan) diimplementasikan untuk secara otomatis menghentikan percobaan yang tidak menjanjikan sejak awal, sehingga menghemat waktu komputasi. Akurasi validasi tertinggi dari setiap percobaan digunakan sebagai metrik untuk memandu proses optimasi. Hyperparameter terbaik yang ditemukan dari skenario ini kemudian akan digunakan untuk melatih ulang model final

sebelum evaluasi performa akhir pada data uji. Skenario ini penting untuk memastikan setiap model dapat menampilkan potensi terbaiknya, terutama bagi arsitektur yang sensitif terhadap konfigurasi pelatihan.

4.6. Evaluasi Performa

Evaluasi performa model dilakukan untuk mengukur seberapa baik model dalam mengklasifikasikan citra hasil uji silang serasi. Beberapa metrik utama yang digunakan dalam pengujian ini antara lain **accuracy**, **precision**, **recall**, ***f1-score***, dan **confusion matrix**. Metrik-metrik ini merupakan standar dalam evaluasi klasifikasi citra, terutama dalam konteks multi-kelas seperti kerja praktik ini.

Selain itu, untuk memperkuat analisis performa, dilakukan juga visualisasi terhadap beberapa gambar yang salah diklasifikasikan oleh model. Visualisasi ini menunjukkan contoh citra hasil uji silang serasi yang diprediksi berbeda dari label sebenarnya, lengkap dengan informasi kelas prediksi dan kelas sebenarnya. Dari pengamatan, kesalahan klasifikasi umumnya terjadi antara kelas yang secara visual sangat mirip, seperti antara *Negatif* dengan *Positif 1*. Visualisasi ini membantu mengevaluasi karakteristik citra yang menjadi sumber kebingungan bagi model, serta menjadi masukan penting dalam pengembangan preprocessing lanjutan atau fine-tuning model.

[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari sistem yang kami buat. Implementasi ini akan dibagi ke dalam beberapa bagian, yaitu bagian implementasi dataset dan loader, implementasi model *pretrained*, implementasi train model, implementasi evaluasi dan visualisasi, dan implementasi *cross validation*.

5.1. Implementasi Dataset dan Loader

Implementasi pertama yang dilakukan adalah mengimport *library. Library* atau pustaka yang digunakan pada kerja praktik ini secara umum adalah sebagai berikut

```
1.  import torch
2.  import torchvision.transforms as transforms
3.  import torchvision.models as models
4.  from torch.utils.data import DataLoader, Dataset
5.  import timm
6.  import pandas as pd
7.  import numpy as np
8.  import os
9.  import time
10. from PIL import Image
11. import matplotlib.pyplot as plt
12. import seaborn as sns
13. from sklearn.metrics import accuracy_score,
    precision_score, recall_score, f1_score,
    confusion_matrix
```

Kode Sumber 5.1. *Import Library*

Pada tahap ini, dilakukan proses pemuatan dataset hasil segmentasi citra uji silang serasi serta pembentukan mekanisme data loader untuk proses pelatihan model deep learning. Dataset dimuat dari file berformat JSON yang berisi informasi nama file

gambar serta label kelasnya, kemudian dibagi ke dalam tiga subset data: pelatihan, validasi, dan pengujian. Pembagian dilakukan secara acak dengan proporsi 80% data pelatihan, 10% data validasi, dan 10% data pengujian.

```
1.  DF <- read_json("LABEL_FILE")

2.  class_map <- {}
3.  FOR index, class_name in
    enumerate(unique(DF['class'])): DO
4.      class_map[class_name] <- index
5.  ENDFOR

6.  FOR each row in full_dataset DO
7.      label <- row["class"]
8.      row["class"] <- class_map[label]
9.  ENDFOR

10. SEED <- 42
11. DF <- shuffle(DF, random_seed=SEED)

12. total_data <- length(DF)
13. train_size <- 0.8 * total_data
14. val_size <- 0.1 * total_data
15. test_size <- total_data - train_size - val_size

16. train_df <- DF[0 : train_size]
17. val_df <- DF[train_size : train_size + val_size]
18. test_df <- DF[train_size + val_size : end]
```

Kode Sumber 5.2. Pseudocode untuk split dan *class mapping dataset*

Dataset dimuat menggunakan *library* pandas dari file JSON, yang diasumsikan berisi kolom filename dan class. Label kelas yang semula dalam format teks (misalnya “Negatif”, “Positif 1”) dikonversi menjadi angka menggunakan dictionary class_mapping. Data diacak menggunakan seed acak (random_state=42) untuk menjaga reprodusibilitas. Setelah itu,

dilakukan pembagian data menjadi tiga subset: 80% untuk pelatihan, 10% untuk validasi, dan 10% untuk pengujian. Pembagian ini bertujuan menjaga proporsi kelas yang seimbang dalam setiap subset dan memastikan evaluasi model dilakukan secara objektif.

```
1.  BATCH_SIZE <- 32
2.  TARGET_SIZE <- (224, 224)
3.  EPOCHS <- 30
4.  NUM_CLASSES <- length(class_map)
5.  DEVICE <- "cuda" IF GPU available ELSE "cpu"

6.  train_transform <- [
7.      Resize to TARGET_SIZE,
8.      Random Horizontal Flip,
9.      Adjust Brightness and Contrast (ColorJitter),
10.     Convert to Tensor,
11.     Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
    0.224, 0.225])
12. ]

13. test_transform <- [
14.     Resize to TARGET_SIZE,
15.     Convert to Tensor,
16. Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
    0.224, 0.225])
17. ]
```

Kode Sumber 5.3. Pseudocode sumber untuk transform citra

Untuk data pelatihan, dilakukan augmentasi berupa horizontal flip, modifikasi *brightness* dan *contrast* untuk meningkatkan variasi data dan mencegah overfitting. Seluruh gambar diubah ukurannya menjadi 224x224 piksel agar sesuai dengan dimensi input model pretrained. Selain itu, dilakukan normalisasi berdasarkan nilai rata-rata dan standar deviasi dataset ImageNet.

5.1.1. Class Dataset dan DataLoader

Dataset dikemas ke dalam class Python berbasis torch.utils.data.Dataset. Class ini bertugas memuat citra dari direktori tertentu, membaca label dari DataFrame, menerapkan transformasi yang diperlukan, dan mengembalikan pasangan (image, label) untuk digunakan dalam pelatihan model. Dataset kemudian digunakan untuk membentuk objek DataLoader dari PyTorch, dengan parameter batch size sebesar 32. Loader ini akan mengatur proses batching, shuffling, dan prefetching data selama proses pelatihan dan evaluasi model.

```
1.  CLASS ImageDataset:
2.      INIT(dataframe, root_dir, transform):
3.          self.data <- dataframe
4.          self.dir <- root_dir
5.          self.transform <- transform

6.      LENGTH():
7.          RETURN length(self.data)

8.      GET_ITEM(index):
9.          image_path <- self.dir + "/" +
    self.data[index]["filename"]
10.         image <- load_image(image_path)
11.         image <- convert_to_RGB(image)
12.         label <- self.data[index]["class"]
13.         IF self.transform is not NULL THEN:
14.             image <- apply_transform(self.transform,
    image)
15.         RETURN (image, label)

16. train_dataset <- ImageDataset(train_df, FINAL_DIR,
    train_transform)
17. val_dataset <- ImageDataset(val_df, FINAL_DIR,
    test_transform)
18. test_dataset <- ImageDataset(test_df, FINAL_DIR,
    test_transform)
```

```
train_loader <- DataLoader(train_dataset,
    batch_size=BATCH_SIZE, shuffle=TRUE)
val_loader <- DataLoader(val_dataset,
    batch_size=BATCH_SIZE, shuffle=FALSE)
test_loader <- DataLoader(test_dataset,
    batch_size=BATCH_SIZE, shuffle=FALSE)
```

Kode Sumber 5.4. Pseudocode dataset kustom dan *loader*

5.2. Implementasi Model Deep Learning

Setelah dataset dan data loader disiapkan, tahap selanjutnya adalah implementasi model deep learning. Pada kerja praktik ini, digunakan pendekatan transfer learning dengan memanfaatkan lima arsitektur model *pre-trained* yang berbeda dari library *timm*. Proses ini mencakup inisialisasi model dengan bobot dari *ImageNet*, modifikasi lapisan klasifikasi akhir, dan penerapan teknik *freezing layer* untuk melatih hanya sebagian parameter model.

5.2.1. Inisialisasi dan Konfigurasi Model

Proses implementasi diawali dengan mendefinisikan daftar nama model yang akan digunakan. Setiap model dimuat menggunakan fungsi *timm.create_model.* Parameter *pretrained=True* digunakan untuk menginisialisasi model dengan bobot yang telah dilatih pada dataset *ImageNet*, sementara *num_classes=NUM_CLASSES* secara otomatis mengganti lapisan klasifikasi akhir agar sesuai dengan jumlah kelas pada dataset uji silang serasi (yaitu, 5 kelas).

```
MODEL_NAMES <- ["regnety_032.tv2_in1k",
                "swinv2_cr_small_224.sw_in1k",
                "convnextv2_base.fcmae_ft_in1k",
```

```
                    "efficientnetv2_rw_s.ra2_in1k",
                    "vit_base_r50_s16_224.orig_in21k"]

MODELS <- {}
NUM_CLASSES <- 5

FOR each model_name in MODEL_NAMES DO
    model <- create_model(model_name, pretrained=TRUE,
        num_classes=NUM_CLASSES)

    MODELS[model_name] <- model
ENDFOR
```

Kode Sumber 5.5. Pseudocode untuk inisialisasi model pre-trained

5.2.2. Penerapan Freezing Layer

Setelah inisialisasi model, sebagian besar parameter model "dibekukan" (*freeze*) sehingga tidak akan diperbarui selama proses pelatihan. Hanya parameter pada lapisan klasifikasi akhir yang baru saja ditambahkan yang akan dilatih. Hal ini bertujuan untuk menyesuaikan model dengan domain citra medis yang spesifik tanpa merusak fitur-fitur umum yang telah dipelajari.

Proses freezing dilakukan dengan mengiterasi seluruh parameter dari setiap model. Parameter diidentifikasi berdasarkan namanya. Jika nama sebuah parameter diawali dengan nama lapisan klasifikasi yang spesifik untuk model tersebut (misalnya, *fc, head,* atau *head.fc*), maka gradiennya diaktifkan (*requires_grad = True*). Sebaliknya, semua parameter lain gradiennya dinonaktifkan (*requires_grad = False*).

```
FOR each model_name, model in MODELS DO
    classifier_key <- default_classifier_name(model)

    FOR each param_name, param in model.parameters DO
        IF param_name starts_with classifier_key THEN
```

```
            param.requires_grad <- TRUE
        ELSE
            param.requires_grad <- FALSE
        ENDIF
    ENDFOR

ENDFOR
```

Kode Sumber 5.6. Pseudocode untuk penerapan freezing layer

5.3. Implementasi Pelatihan Model

Tahap implementasi inti adalah proses pelatihan model. Proses ini melibatkan pendefinisian *loss function*, *optimizer*, dan alur pelatihan utama yang mencakup *forward pa*ss, *backward pass*, validasi per-epoch, serta mekanisme untuk meningkatkan stabilitas dan efisiensi pelatihan, yaitu learning rate scheduler dan early stopping.

5.3.1. Fungsi Pelatihan dan Validasi

Sebuah fungsi utama, *train_model*, dirancang untuk mengelola seluruh siklus hidup pelatihan untuk satu model. Fungsi ini menerima model yang akan dilatih, data loader untuk data pelatihan dan validasi, serta parameter seperti jumlah epoch dan patience untuk *early stopping*.

Di dalam fungsi ini, *optimizer* Adam dipilih untuk memperbarui bobot model, dengan *learning rate* awal sebesar 1e-4 dan *weight decay* sebesar 1e-5 untuk regularisasi. Fungsi kerugian yang digunakan adalah *Cross-Entropy Loss*, yang merupakan standar untuk masalah klasifikasi multi-kelas. Selain itu, *ReduceLROnPlateau scheduler* diimplementasikan untuk secara dinamis mengurangi *learning rate* jika *validation loss* tidak menunjukkan perbaikan

.

```
FUNCTION train_model(model, train_loader, val_loader,
```

```
    epochs, patience):
2.      optimizer <- Adam(model.parameters, lr=1e-4,
    weight_decay=1e-5)
3.      scheduler <- ReduceLROnPlateau(optimizer,
    mode='min', patience=2)
4.      criterion <- CrossEntropyLoss()
5.      model <- move_to_device(model, DEVICE)

6.      best_val_loss <- infinity
7.      best_model_weights <- NULL
8.      epochs_without_improvement <- 0
9.      history <- {}

10.     FOR epoch FROM 1 TO epochs DO
11.         model.train_mode()
12.         total_train_loss <- 0, total_train_correct <- 0
13.         FOR each (images, labels) in train_loader DO
14.             images, labels <- move_to_device(images,
    labels, DEVICE)

15.             optimizer.zero_grad()
16.             outputs <- model(images)
17.             loss <- criterion(outputs, labels)
18.             loss.backward
19.             optimizer.step()

20.             total_train_loss += loss
21.             total_train_correct +=
    count_correct_predictions(outputs, labels)
22.         ENDFOR

23.         model.evaluation_mode()
24.         total_val_loss <- 0, total_val_correct <- 0
25.         WITH no_grad():
26.             FOR each (images, labels) in val_loader DO
27.                 images, labels <-
    move_to_device(images, labels, DEVICE)
28.                 outputs <- model(images)
29.                 loss <- criterion(outputs, labels)
```

```
                total_val_loss += loss
                total_val_correct +=
    count_correct_predictions(outputs, labels)
            ENDFOR
        ENDFOR

        avg_train_loss, avg_train_acc <-
    calculate_average(total_train_loss,
    total_train_correct, train_loader.dataset)
        avg_val_loss, avg_val_acc <-
    calculate_average(total_val_loss, total_val_correct,
    val_loader.dataset)
        add_to_history(history, avg_train_loss,
    avg_train_acc, avg_val_loss, avg_val_acc)

        scheduler.step(avg_val_loss)
        IF avg_val_loss < best_val_loss THEN
            best_val_loss <- avg_val_loss
            best_model_weights <- copy(model.weights)
            epochs_without_improvement <- 0
        ELSE
            epochs_without_improvement += 1
            IF epochs_without_improvement >= patience
    THEN
                PRINT "Early stopping triggered"
                BREAK LOOP
            ENDIF
        ENDIF
    ENDFOR

    IF best_model_weights IS NOT NULL THEN
        model.load_weights(best_model_weights)
    ENDIF

    RETURN history, total_training_time
```

Kode Sumber 5.7.Pseudocode untuk fungsi pelatihan dan validasi model

5.3.2. Eksekusi Pelatihan untuk Semua Model

Setelah fungsi *train_model* didefinisikan, sebuah perulangan utama dieksekusi untuk melatih setiap model yang telah diinisialisasi sebelumnya. Setiap model dari dictionary MODELS dilatih secara berurutan menggunakan fungsi *train_model*. Hasil pelatihan, berupa *history loss* dan akurasi serta total waktu pelatihan, disimpan dalam dictionary terpisah untuk analisis lebih lanjut.

```
1.  TRAINING_HISTORIES <- {}
2.  TRAINING_TIMES <- {}

3.  FOR each model_name, model in MODELS DO
4.      history, train_time <- train_model(model,
    train_loader, val_loader, epochs=30, patience=5)
5.      TRAINING_HISTORIES[model_name] <- history
6.      TRAINING_TIMES[model_name] <- train_time
7.  ENDFOR
```

Kode Sumber 5.8.Pseudocode untuk eksekusi loop pelatihan

5.4. Implementasi Evaluasi dan Visualisasi

Setelah semua model dilatih, tahap akhir adalah mengevaluasi performa mereka secara kuantitatif dan kualitatif. Tahap ini mencakup visualisasi kurva pembelajaran (learning curve), perhitungan metrik performa standar pada data uji, serta visualisasi confusion matrix dan identifikasi kesalahan klasifikasi untuk analisis yang lebih mendalam.

5.4.1. Visualisasi Kurva Pembelajaran

Untuk menganalisis perilaku model selama proses pelatihan, diimplementasikan fungsi plot_learning_curve. Fungsi ini mengambil data histori pelatihan (yang berisi loss dan akurasi per-epoch) dan menghasilkan dua grafik: satu untuk loss (pelatihan

vs. validasi) dan satu lagi untuk akurasi (pelatihan vs. validasi). Visualisasi ini sangat penting untuk mendeteksi masalah seperti overfitting (di mana train loss terus menurun sementara validation loss meningkat) atau underfitting (di mana kedua loss tetap tinggi).

```
1.  FUNCTION plot_learning_curve(history, model_name):
2.      epochs <- range from 1 to
    length(history['train_loss'])
3.
4.      CREATE figure with two subplots (1 row, 2 columns)

5.      SUBPLOT 1:
6.          PLOT epochs vs history['train_loss'] with label
    "Train Loss"
7.          PLOT epochs vs history['val_loss'] with label
    "Validation Loss"
8.          SET title "Loss per Epoch"
9.          SET x-label "Epoch", y-label "Loss"
10.         ADD legend and grid

11.     SUBPLOT 2:
12.         PLOT epochs vs history['train_acc'] with label
    "Train Accuracy"
13.         PLOT epochs vs history['val_acc'] with label
    "Validation Accuracy"
14.         SET title "Accuracy per Epoch"
15.         SET x-label "Epoch", y-label "Accuracy"
16.         ADD legend and grid

17.     SHOW plot with main title "Learning Curve: " +
    model_name
18. ENDFUNCTION

19. FOR each model_name, history in TRAINING_HISTORIES DO
20.     plot_learning_curve(history, model_name)
21. ENDFOR
```

Kode Sumber 5.9.Pseudocode untuk visualisasi kurva pembelajaran

5.4.2. Evaluasi pada Data Uji

Evaluasi performa final dilakukan pada test set, yaitu data yang sama sekali belum pernah dilihat oleh model selama proses pelatihan atau validasi. Untuk membantu analisis kualitatif, sebuah fungsi *find_mistakes* diimplementasikan untuk mengidentifikasi semua sampel yang salah diklasifikasikan. Fungsi ini mengiterasi hasil prediksi dan membandingkan setiap label prediksi dengan label sebenarnya. Jika ditemukan ketidaksesuaian, nama file gambar beserta label asli dan label prediksinya akan dicatat. Hasil dari fungsi ini adalah sebuah dictionary yang memetakan nama file ke kesalahan klasifikasinya, memungkinkan inspeksi visual yang mudah terhadap kasus-kasus sulit.

```
1.  FUNCTION find_mistakes(true_labels, pred_labels,
    filenames):
2.      mistakes <- {}
3.      FOR each (true, pred, fname) in (true_labels,
    pred_labels, filenames):
4.          IF true IS NOT EQUAL TO pred THEN
5.              mistakes[fname] <- (true, pred)
6.          ENDIF
7.      ENDFOR
8.      RETURN mistakes
9.  ENDFUNCTION
```

Kode Sumber 5.10. Pseudocode untuk fungsi *find_mistake*

Sebuah loop evaluasi diimplementasikan untuk setiap model. Di dalam loop ini, model diatur ke mode evaluasi (model.eval()) dan semua prediksi dibuat tanpa menghitung gradien. Label prediksi dan label sebenarnya dikumpulkan untuk seluruh data uji. Berdasarkan prediksi dan label sebenarnya, serangkaian metrik performa dihitung menggunakan library scikit-learn, yaitu *Accuracy, Precision, Recall, F1-Score*, serta waktu prediksi.

```
ALL_SCORES <- {ACCURACY:{}, PRECISION:{}, RECALL:{}, F1_SCORE:{}, TIME:{}}
ALL_MATRICES <- {}
ALL_MISTAKES <- {}

FOR each model_name, model in MODELS DO
    model.evaluation_mode()
    all_predictions <- [], all_actuals <- []

    start_time <- current_time()
    WITH no_grad():
        FOR each (images, labels) in test_loader DO
            images <- move_to_device(images, DEVICE)
            outputs <- model(images)

            predictions <- argmax(outputs)

            add predictions to all_predictions
            add labels to all_actuals
        ENDFOR
    ENDFOR
    end_time <- current_time()

    ALL_SCORES['ACCURACY'][model_name] <- calculate_accuracy(all_actuals, all_predictions)
    ALL_SCORES['PRECISION'][model_name] <- calculate_precision(all_actuals, all_predictions, average='macro')
    ALL_SCORES['RECALL'][model_name] <- calculate_recall(all_actuals, all_predictions, average='macro')
    ALL_SCORES['F1_SCORE'][model_name] <- calculate_f1_score(all_actuals, all_predictions, average='macro')
    ALL_SCORES['TIME'][model_name] <- end_time - start_time

    cm <- create_confusion_matrix(all_actuals, all_predictions)
    ALL_MATRICES[model_name] <- cm
```

```
    mistakes <- find_mistakes(all_actuals,
    all_predictions, test_filenames)
    ALL_MISTAKES[model_name] <- mistakes
ENDFOR
```

Kode Sumber 5.11. Pseudocode untuk evaluasi kuantitatif

5.4.3. Analisis dan Visualisasi Kesalahan

Selain metrik kuantitatif, analisis kualitatif juga penting untuk memahami kekuatan dan kelemahan model. Dua pendekatan visualisasi diimplementasikan. Pendekatan pertama adalah *Confusion Matrix*. Untuk setiap model, dibuat sebuah *confusion matrix* yang memvisualisasikan performa klasifikasi untuk setiap kelas. Sumbu-y merepresentasikan kelas sebenarnya, sedangkan sumbu-x merepresentasikan kelas yang diprediksi. Diagonal utama menunjukkan jumlah prediksi yang benar, sementara sel di luar diagonal menunjukkan kesalahan klasifikasi. Ini membantu mengidentifikasi kelas mana yang sering tertukar satu sama lain.

```
FUNCTION draw_confusion_matrix(cm, class_labels,
model_name):
    CREATE figure
    USE heatmap to visualize the confusion matrix 'cm'
    SET x-label "Predicted Label", y-label "True Label"
    SET title "Confusion Matrix: " + model_name
    SHOW plot
ENDFUNCTION

FOR each model_name in MODELS DO
    cm <- ALL_MATRICES[model_name]
    draw_confusion_matrix(cm, class_labels, model_name)
ENDFOR
```

Kode Sumber 5.12. Pseudocode untuk visualisasi confusion matrix

Sebuah fungsi juga diimplementasikan untuk mengidentifikasi dan menyimpan gambar yang salah

diklasifikasikan oleh model. Informasi yang disimpan mencakup nama file, label asli, dan label yang diprediksi. Analisis ini memungkinkan pengamatan langsung terhadap contoh-contoh sulit yang membingungkan model, memberikan wawasan untuk perbaikan di masa depan, seperti augmentasi data yang lebih spesifik atau penyesuaian arsitektur.

```
1.  FUNCTION show_mistakes(mistakes, root_dir, model_name,
    max_images=10):
2.      mistake_items <- get first 'max_images' items from
    mistakes dictionary
3.      IF mistake_items is empty THEN
4.          PRINT "No mistakes found for " + model_name
5.          RETURN
6.      ENDIF
7.      CREATE figure with subplots (1 row,
    length(mistake_items) columns)
8.      SET main title "Misclassified Images - " +
    model_name

9.      FOR each subplot_ax, (filename, (true_label,
    pred_label)) in (subplots, mistake_items):
10.         image_path <- combine_path(root_dir, filename)
11.         image <- load_image(image_path)

12.         subplot_ax.show_image(image)
13.         subplot_ax.set_title("True: " + true_label +
    "\nPred: " + pred_label)
14.         subplot_ax.hide_axes()
15.     ENDFOR

16.     SHOW plot
17. ENDFUNCTION

18. FOR each model_name in MODELS DO
19.     mistakes_for_model <- ALL_MISTAKES[model_name]
20.     show_mistakes(mistakes_for_model, IMAGE_DIRECTORY,
    model_name)
21. ENDFOR
```

Kode Sumber 5.13. Pseudocode untuk visualisasi kesalahan klasifikasi

5.5. Implementasi K-Fold Cross Validation

Untuk mendapatkan estimasi performa model yang lebih *robust* dan tidak bias terhadap satu pembagian data tertentu, diimplementasikan skenario pengujian menggunakan *Stratified K-Fold Cross-Validation*. Metode ini memastikan bahwa setiap sampel data memiliki kesempatan untuk menjadi bagian dari data uji, sehingga memberikan evaluasi yang lebih komprehensif terhadap kemampuan generalisasi model. Dalam kerja praktik ini, digunakan k=5 *folds*.

5.5.1. Inisialisasi dan Pembagian Data

Proses diawali dengan menginisialisasi objek StratifiedKFold dari library *scikit-learn* dengan jumlah n_splits=5. Penggunaan StratifiedKFold penting untuk memastikan bahwa distribusi kelas di setiap fold tetap proporsional dengan distribusi kelas di keseluruhan dataset, yang sangat relevan untuk masalah klasifikasi multi-kelas. Objek ini kemudian digunakan untuk menghasilkan indeks data latih dan data uji untuk setiap iterasi fold.

```
1.  X_features <- all_filenames_from_dataset
2.  Y_labels <- all_class_labels_from_dataset

3.  N_SPLITS <- 5
4.  skf <- StratifiedKFold(n_splits=N_SPLITS, shuffle=TRUE,
    random_state=SEED)

5.  cv_results <- {}
6.  FOR each model_name in MODEL_NAMES DO
7.      cv_results[model_name] <- {accuracy:[],
    precision:[], recall:[], f1_score:[], train_time:[]}
8.  ENDFOR
```

Kode Sumber 5.14. Pseudocode untuk inisialisasi K-Fold

5.5.2. Loop Pelatihan dan Evaluasi K-Fold

Implementasi utama dari *cross-validation* adalah sebuah perulangan bersarang (*nested loop*). Perulangan terluar mengiterasi setiap arsitektur model, sedangkan perulangan di dalamnya mengiterasi setiap dari 5 *folds* yang dihasilkan oleh StratifiedKFold. Berikut adalah implementasi untuk evaluasi satu *fold* .

```
1.  FUNCTION evaluate_fold(model, val_loader):
2.      model.evaluation_mode()
3.      all_predictions <- [], all_actuals <- []
4.      start_time <- current_time()
5.      WITH no_grad():
6.          FOR each (images, labels) in test_loader DO
7.              images <- move_to_device(images, DEVICE)
8.              outputs <- model(images)
9.              predictions <- argmax(outputs)
10.             add predictions to all_predictions
11.             add labels to all_actuals
12.         ENDFOR
13.     ENDFOR
14.     end_time <- current_time()

15.     accuracy ← calculate_accuracy(all_labels,
    all_predictions)
16.     precision ← calculate_precision(all_labels,
    all_predictions, average='macro'0)
17.     recall ← calculate_recall(all_labels,
    all_predictions, average='macro')
18.     f1 ← calculate_f1(all_labels, all_predictions,
    average='macro')

19.     RETURN accuracy, precision, recall, f1
20. ENDFUNCTION
```

Kode Sumber 5.15. Pseudocode untuk evaluasi satu *fold*

Di dalam setiap iterasi fold, serangkaian langkah krusial dijalankan. Proses *cross-validation* dimulai dengan inisialisasi ulang model menggunakan bobot pre-trained untuk menghindari kebocoran data dari *fold* sebelumnya. Selanjutnya, dataset dibagi menjadi subset pelatihan dan validasi menggunakan skf.split(), kemudian dibuat *data loader* baru khusus untuk fold tersebut. Model dilatih menggunakan data latih dengan validasi dilakukan pada subset validasi untuk mendukung mekanisme *early stopping*, menggunakan fungsi train_model. Setelah pelatihan, model dievaluasi pada data validasi untuk menghitung metrik performa seperti akurasi, presisi, recall, dan *f1-score*, lalu hasil evaluasi disimpan ke dalam *dictionary* cv_results untuk analisis selanjutnya.

```
1.  FOR each model_name in MODEL_NAMES DO
2.      FOR each fold, (train_indices, val_indices) in
    skf.split(X_features, Y_labels) DO

3.          model <- create_model(model_name,
    pretrained=TRUE)
4.          apply_freezing_layer_logic(model)
5.          train_df_fold <-
    get_data_by_indices(train_indices)
6.          val_df_fold <- get_data_by_indices(val_indices)
7.          train_loader_fold <-
    create_dataloader(train_df_fold,
    transform=train_transform)
8.          val_loader_fold <-
    create_dataloader(val_df_fold,
    transform=test_transform)

9.          _, train_time <- train_model(model,
    train_loader_fold, val_loader_fold)
10.         accuracy, precision, recall, f1 <-
    evaluate_fold(model, val_loader_fold)
11.         add_to_dictionary(cv_results[model_name],
    accuracy, precision, recall, f1, train_time)
```

```
    ENDFOR
ENDFOR
```

Kode Sumber 5.16. Pseudocode untuk loop utama K-Fold Cross-Validation

5.5.3. Agregasi Hasil Akhir

Setelah semua model selesai dilatih dan dievaluasi pada kelima *folds*, langkah terakhir adalah mengagregasi hasilnya. Untuk setiap model, dihitung nilai rata-rata (*mean*) dan standar deviasi (*standard deviation*) dari metrik performa yang telah dikumpulkan (akurasi, *recall*, *precision*, dan *f1-score*,). Nilai rata-rata menunjukkan performa umum model, sementara standar deviasi menunjukkan tingkat konsistensi atau stabilitas performa model di berbagai subset data. Hasil agregat ini memberikan gambaran yang lebih andal mengenai seberapa baik performa model yang diharapkan pada data baru yang belum pernah dilihat.

```
FOR each model_name in MODEL_NAMES DO
    accuracies <- cv_results[model_name]['accuracy']
    f1_scores <- cv_results[model_name]['f1_score']

    mean_accuracy <- calculate_mean(accuracies)
    std_accuracy <- calculate_std_dev(accuracies)
    mean_f1 <- calculate_mean(f1_scores)
    std_f1 <- calculate_std_dev(f1_scores)

    PRINT model_name + ": Accuracy = " + mean_accuracy
+ " ± " + std_accuracy
    PRINT model_name + ": F1-Score = " + mean_f1 + " ±
" + std_f1
ENDFOR
```

Kode Sumber 5.16. Pseudocode untuk agregasi hasil K-Fold

5.6. Implementasi Hyperparameter Tuning dengan Optuna

Untuk memastikan setiap arsitektur model mencapai potensi performa maksimalnya, diimplementasikan skenario optimasi *hyperparameter* otomatis. Proses ini bertujuan untuk menemukan kombinasi parameter pelatihan terbaik secara individual untuk setiap model, mengingat beberapa arsitektur bisa jadi sangat sensitif terhadap konfigurasi pelatihan. *Library* Optuna digunakan untuk mengelola proses pencarian ini secara efisien.

5.6.1. Definisi Fungsi Objektif dan Ruang Pencarian

Inti dari implementasi Optuna adalah sebuah fungsi objective, yaitu "train_and_validate_for_tuning". Fungsi ini dirancang untuk dieksekusi oleh Optuna pada setiap percobaan. Di dalam fungsi ini, didefinisikan ruang pencarian untuk hyperparameter yang akan dioptimalkan, yaitu *learning rate* dalam rentang logaritmik antara 1e-5 hingga 1e-1, *weight decay* dalam rentang logaritmik antara 1e-6 hingga 1e-1 untuk regularisasi, dan Pilihan *Optimizer* antara *Adam* dan *AdamW*.

Untuk setiap percobaan, Optuna akan menyarankan satu set nilai dari ruang pencarian ini. Fungsi objektif kemudian akan menginisialisasi model, melatihnya dengan *hyperparameter* yang disarankan, dan mengevaluasi performanya pada set validasi. Akurasi validasi tertinggi yang dicapai selama pelatihan dikembalikan sebagai nilai skor untuk percobaan tersebut.

```
1.  FUNCTION train_and_validate_for_tuning(trial,
    model_name, train_loader, val_loader):
2.      lr <- trial.suggest_float("lr", 1e-5, 1e-1,
    log=TRUE)
3.      weight_decay <- trial.suggest_float("weight_decay",
    1e-6, 1e-1, log=TRUE)
4.      optimizer_name <-
    trial.suggest_categorical("optimizer", ["Adam",
```

```
        "AdamW"])

5.      model <- create_model(model_name, pretrained=TRUE)
6.      apply_freezing_layer_logic(model)
7.      optimizer <- create_optimizer(optimizer_name,
    model.parameters, lr, weight_decay)

8.      best_val_accuracy <- 0.0
9.      FOR epoch FROM 1 TO EPOCHS_FOR_TUNING DO
10.         train_one_epoch(model, train_loader, optimizer)
11.
12.         current_val_accuracy <-
    validate_one_epoch(model, val_loader)
13.
14.         best_val_accuracy <- max(best_val_accuracy,
    current_val_accuracy)
15.
16.         trial.report(best_val_accuracy, epoch)
17.         IF trial.should_prune() THEN
18.             THROW PrunedTrialException
19.         ENDIF
20.
21.         check_early_stopping(...)
22.     ENDFOR
23.     RETURN best_val_accuracy
```

Kode Sumber 5.17. Pseudocode untuk Fungsi Objektif Optuna

5.6.2. Eksekusi Proses Optimasi

Proses *tuning* dijalankan dalam sebuah loop utama yang mengiterasi setiap nama model. Untuk setiap model, sebuah *study* Optuna dibuat dengan tujuan untuk memaksimalkan (direction="maximize") akurasi validasi yang dikembalikan oleh fungsi objektif.

Proses optimasi “study.optimize” kemudian dijalankan untuk sejumlah 20 percobaan per model. Setelah semua percobaan selesai, Optuna menyediakan akses mudah ke *hyperparameter* dari

percobaan terbaik “study.best_trial.params” serta nilai akurasi validasi terbaik yang dicapai “study.best_value”.

```
1.  best_hyperparams <- {}
2.  N_TRIALS <- 20

3.  FOR each model_name in MODEL_NAMES DO
4.      objective_function <- (lambda trial:
5.          train_and_validate_for_tuning(trial,
    model_name, train_loader, val_loader))
6.
7.      study <- create_optuna_study(direction="maximize",
    pruner=MedianPruner)
8.      study.optimize(objective_function,
    n_trials=N_TRIALS)
9.      best_hyperparams[model_name] <-
    study.best_trial.params

10.     PRINT "Tuning for " + model_name + " completed."
11.     PRINT "Best Validation Accuracy: " +
    study.best_value
12.     PRINT "Best Hyperparameters: " +
    study.best_trial.params
13. ENDFOR

14. PRINT best_hyperparams
```

Kode Sumber 5.18. Pseudocode untuk loop utama proses hyperparameter tuning

BAB VI
HASIL DAN EVALUASI MODEL

Bab ini menyajikan secara rinci hasil dari serangkaian pengujian yang telah dilakukan untuk mengevaluasi dan membandingkan kinerja lima arsitektur model deep learning dalam tugas klasifikasi citra uji silang serasi. Evaluasi dilakukan melalui tiga skenario pengujian yang telah dirancang pada Bab IV, yaitu pengujian dasar dengan parameter seragam, validasi silang K-Fold untuk mengukur stabilitas, dan optimasi hyperparameter menggunakan Optuna untuk menemukan performa puncak setiap model.

6.1. Tujuan Pengujian

Tujuan akhir dari bab ini adalah untuk menganalisis hasil dari setiap skenario, membandingkan performa antar model secara komprehensif, dan memilih arsitektur model yang paling optimal berdasarkan bukti kuantitatif dan kualitatif yang terkumpul.

6.2. Deskripsi Skenario

Skenario uji coba dilakukan pada seluruh *pre-trained* model yang digunakan. Tabel 6.1 menjelaskan mengenai setiap variabel yang digunakan dalam setiap skenario.

Skenario	Deskripsi
1	- Variasi Model *Pre-trined* yang digunakan 1. RegNetY-3.2GF 2. Swin Transformer V2 3. ConvNext V2 Base 4. EfficientNet V2 Small 5. Vision Transformer - *Epochs*: 30 - *Loss Function*: *Cross-Entropy Loss*

	- *Batch Size*: 32 - *Image Size*: 224 x 224 piksel - *Learning Rate*:1e-4 - *Weight Decay*: 1e-5 - *Optimizer*: Adam - *Data Split*: 80:10::10
2	- Variasi Model *Pre-trined* yang digunakan 1. RegNetY-3.2GF 2. Swin Transformer V2 3. ConvNext V2 Base 4. EfficientNet V2 Small 5. Vision Transformer - *Epochs*: 30 - *Loss Function*: *Cross-Entropy Loss* - *Batch Size*: 32 - *Image Size*: 224 x 224 piksel - *Learning Rate*: 1e-4 - *Weight Decay*: 1e-5 - *Optimizer*: Adam - *Data Split*: *Stratified 5-Fold*
3	- Variasi Model *Pre-trined* yang digunakan 1. RegNetY-3.2GF 2. Swin Transformer V2 3. ConvNext V2 Base 4. EfficientNet V2 Small 5. Vision Transformer - *Epochs*: 30 - *Loss Function*: *Cross-Entropy Loss* - *Batch Size*: 32 - *Image Size*: 224 x 224 piksel - *Learning Rate*: 1e-5 - 1e-1 - *Weight Decay*: 1e-6 - 1e-1

	- *Optimizer*: Adam, AdamW - *Data Split*: 80:10:10

Tabel 6.1. Deskripsi Skenario Pengujian

6.3. Skenario 1: Pengujian Dasar

Skenario pertama ini bertujuan untuk mendapatkan gambaran performa dasar (baseline) dari kelima model. Setiap model dilatih menggunakan serangkaian hyperparameter awal yang seragam (*learning rate* = 1e-4, *weight decay* = 1e-5, *optimizer* = Adam) pada pembagian data 80% latih, 10% validasi, dan 10% uji. Hasil dari skenario ini menjadi titik acuan untuk mengevaluasi dampak dari metode validasi dan optimasi pada skenario berikutnya.

6.3.1. Performa dari Hasil Uji Coba Skenario Dasar

Hasil evaluasi performa pada data uji untuk skenario dasar disajikan pada Tabel 6.1. Metrik yang diukur meliputi akurasi, presisi, recall, dan *f1-score*.

Model	***Accuracy***	***Precision***	***Recall***	***F1-score***
RegNetY-3.2GF	0.8493	0.7133	0.6035	0.6121
Swin Transformer V2	**0.9589**	**0.9694**	**0.9668**	**0.9677**
ConvNext V2 Base	0.8219	0.4779	0.5392	0.5067
EfficientNet V2 Small	0.4932	0.3488	0.3279	0.3378
Vision Transformer	0.9452	0.9535	0.9535	0.9535

Tabel 6.2. Hasil Evaluasi Kuantitatif Uji Coba Skenario Dasar

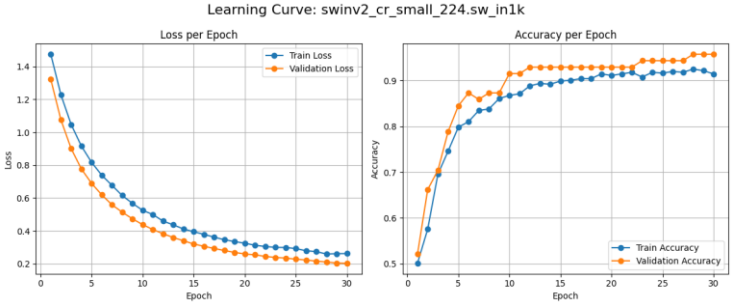
6.3.2. Analisis dan Pembahasan

Berdasarkan data performa pada Tabel 6.1, dapat dilakukan analisis mendalam mengenai karakteristik setiap model pada konfigurasi baseline. Secara umum, terdapat disparitas performa yang sangat jelas antar arsitektur.

Model berbasis *Transformer*, yaitu Swin Transformer V2 dan Vision Transformer, menunjukkan superioritas yang nyata. Swin Transformer V2 menjadi model dengan performa terbaik, mencapai *F1-Score* sebesar 0.9677. Keberhasilan ini mengindikasikan bahwa kemampuan arsitektur *Transformer* untuk menangkap hubungan kontekstual global pada citra sangat cocok untuk membedakan pola pada citra uji silang serasi.

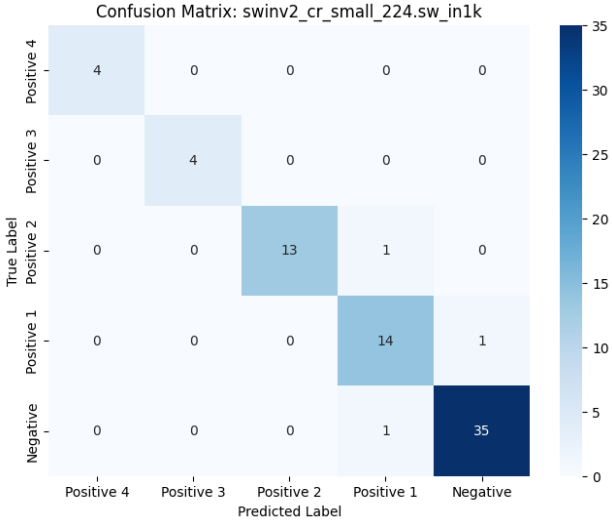
Sebaliknya, EfficientNetV2 Small menunjukkan performa yang sangat buruk dengan akurasi hanya 49.32%, yang hanya sedikit lebih baik dari tebakan acak pada 5 kelas (20%). Hasil ini mengindikasikan bahwa model mengalami kegagalan konvergensi dengan *hyperparameter* awal yang diberikan, dan sangat sensitif terhadap konfigurasi pelatihan. Sementara itu, dua model berbasis CNN lainnya, RegNetY dan ConvNeXt, memberikan hasil akurasi yang moderat. Namun, nilai *f1-score* mereka yang relatif rendah (masing-masing 0.6121 dan 0.5067) menunjukkan bahwa model-model ini kesulitan dalam menyeimbangkan presisi dan recall, yang mengarah pada diagnosis yang kurang andal.

Untuk memahami perilaku pelatihan lebih lanjut, kurva pembelajaran dari model berkinerja terbaik, Swin Transformer V2, disajikan pada Gambar 6.1. Kurva ini menunjukkan proses pembelajaran yang bagus, dimana loss pada data latih dan validasi sama-sama menurun secara konsisten, dan akurasi keduanya meningkat dan mendekati satu sama lain, menandakan tidak adanya *overfitting* yang signifikan.



Gambar 6.1. Kurva Pembelajaran Swin Transformer V2 pada Skenario Dasar

Analisis kesalahan dilakukan untuk memahami kelemahan spesifik model. *Confusion matrix* dari Swin Transformer V2. Gambar 6.2 menunjukkan bahwa kesalahan klasifikasi paling sering terjadi antara kelas Negatif dan Positif 1. Hal ini wajar mengingat kemiripan visual antara kedua kelas tersebut.



Gambar 6.2. *Confusion Matrix* Swin Transformer V2 pada Skenario Dasar

Visualisasi pada Gambar 6.3 menampilkan beberapa contoh gambar yang salah diklasifikasikan oleh model Swin Transformer V2. Terlihat bahwa kesalahan juga terjadi pada kelas-kelas yang berdekatan, seperti Positif 1 yang diprediksi sebagai Negatif, atau Positif 2 yang diprediksi sebagai Positif 1. Kesalahan ini umumnya terjadi pada sampel di mana pola aglutinasinya tidak jelas atau berada di ambang batas antar kelas.

Gambar 6.3. Contoh Kesalahan Klasifikasi oleh Model Swin Transformer V2

Secara keseluruhan, skenario dasar ini menetapkan bahwa arsitektur Transformer memiliki keunggulan inheren pada tugas ini, namun juga menyoroti potensi masalah pada model-model tertentu yang memerlukan investigasi lebih lanjut melalui skenario pengujian berikutnya.

6.4. Skenario 2: *K-Fold Cross-Validation*

Skenario kedua ini bertujuan untuk menguji stabilitas dan keandalan performa model terhadap variasi data latih. Dengan menggunakan *5-Fold Cross-Validation*, setiap model dievaluasi pada lima subset data yang berbeda, sehingga memberikan estimasi performa generalisasi yang lebih andal daripada pembagian data tunggal. Hasil yang disajikan pada Tabel 6.3 dan tabel 6.4 adalah nilai rata-rata (*mean*) dan standar deviasi (*standard deviation*) dari metrik yang diukur selama 5 iterasi.

Model	***Accuracy***	***Precision***	***Recall***	***F1-score***
RegNetY-3.2GF	0.8317	0.7392	0.5682	0.5785
Swin Transformer V2	**0.9346**	**0.9256**	**0.8975**	**0.9079**

ConvNext V2 Base	0.8512	0.8030	0.6197	0.6414
EfficientNet V2 Small	0.5870	0.4407	0.4373	0.4228
Vision Transformer	0.9277	0.9152	0.8818	0.8917

Tabel 6.3. Hasil Rata-rata Metrik Uji Coba pada Skenario *K-Fold*

Model	***Accuracy***	***Precision***	***Recall***	***F1-score***
RegNetY-3.2GF	0.0048	0.1585	0.0206	0.0470
Swin Transformer V2	**0.0033**	**0.0134**	**0.0094**	**0.0112**
ConvNext V2 Base	0.0127	0.1271	0.0537	0.0774
EfficientNet V2 Small	0.1138	0.0954	0.0806	0.0880
Vision Transformer	0.0175	0.0179	0.0333	0.0247

Tabel 6.4. Hasil Standar Deviasi Metrik Uji Coba pada Skenario *K-Fold*

Hasil dari K-Fold Cross-Validation memperkuat temuan awal bahwa arsitektur berbasis Transformer menunjukkan performa rata-rata yang superior. Namun, wawasan paling penting dari skenario ini datang dari nilai standar deviasi. Swin Transformer V2 tidak hanya mencapai *f1-score* rata-rata tertinggi, tetapi juga memiliki standar deviasi terendah. Hal ini membuktikan bahwa Swin Transformer V2 adalah model yang paling stabil dan dapat diandalkan, dengan performa yang konsisten di berbagai subset data.

Sebaliknya, EfficientNetV2 Small menunjukkan ketidakstabilan yang ekstrem, ditandai dengan standar deviasi yang cukup tinggi. Ini mengonfirmasi bahwa performa model ini sangat rentan terhadap data latih yang spesifik, menjadikannya pilihan yang kurang dapat diandalkan tanpa adanya optimasi lebih lanjut. Temuan ini menggarisbawahi pentingnya tidak hanya melihat performa rata-rata, tetapi juga konsistensinya.

6.5. Skenario 3: Optimasi Hyperparameter Optuna

Skenario pengujian terakhir ini bertujuan untuk membuka potensi performa maksimal dari setiap arsitektur. Berdasarkan hasil

sebelumnya yang menunjukkan sensitivitas beberapa model terhadap konfigurasi awal, proses optimasi hyperparameter otomatis dijalankan menggunakan Optuna.

6.6.1. Hyperparameter Optimal yang Ditemukan

Setelah dijalankan 20 percobaan pencarian untuk setiap model dengan *range* parameter sesuai tabel 6.1, Kombinasi hyperparameter yang paling optimal berhasil diidentifikasi oleh Optuna. Hasil hyperparameter terbaik dirangkum pada Tabel 6.5.

Model	***Learning Rate***	***Weight Decay***	***Optimizer***
RegNetY-3.2GF	1.16e-2	3.55e-3	Adam
Swin Transformer V2	6.63e-3	3.09e-3	Adam
ConvNext V2 Base	3.82e-2	2.55e-4	AdamW
EfficientNet V2 Small	1.07e-2	3.99e-5	Adam
Vision Transformer	6.21e-3	8.02e-5	Adam

Tabel 6.5. Hyperparameter Optimal yang Ditemukan oleh Optuna

6.6.2. Performa Model Setelah Optimalisasi

Dengan menggunakan hyperparameter terbaik dari Tabel 6.5, setiap model dilatih ulang dari awal dan dievaluasi pada data uji yang sama. Hasil performa final yang telah dioptimalkan disajikan pada Tabel 6.6.

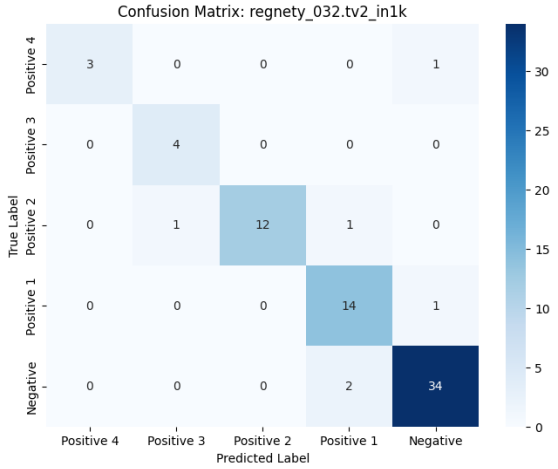
Model	***Accuracy***	***Precision***	***Recall***	***F1-score***
RegNetY-3.2GF	0.9178	0.9136	0.8970	0.8977
Swin Transformer V2	**0.9726**	**0.9811**	**0.9811**	**0.9811**
ConvNext V2 Base	0.9452	0.9592	0.9525	0.9541
EfficientNet V2 Small	0.9315	0.9133	0.9479	0.9284
Vision Transformer	0.9589	0.9667	0.9659	0.9636

Tabel 6.6. Hasil Evaluasi Final Setelah Optimaliasi Hyperparameter

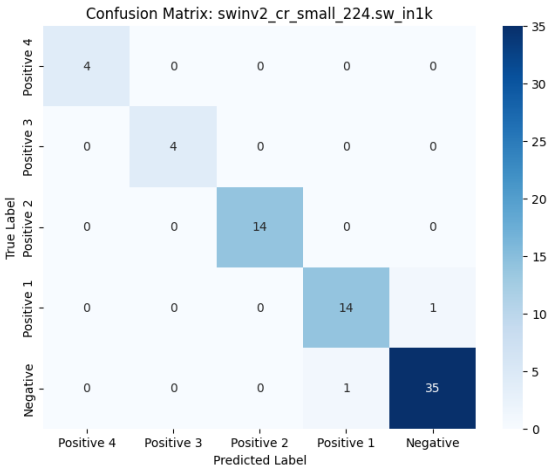
Hasil dari skenario optimasi ini memberikan wawasan yang paling transformatif. Model-model yang sebelumnya berkinerja buruk mengalami peningkatan performa yang sangat signifikan:

- EfficientNetV2: Model ini melompat dari *F1-Score* 0.3378 menjadi 0.9284, sebuah peningkatan drastis sebesar 0.5906. Ini membuktikan secara konklusif bahwa kegagalan awalnya murni disebabkan oleh hyperparameter yang tidak cocok.
- ConvNeXt: Model ini juga mengalami lonjakan performa yang luar biasa, dengan *F1-Score* meningkat dari 0.5067 menjadi 0.9541, menjadikannya salah satu model dengan performa terbaik.
- RegNetY: Model ini juga menunjukkan peningkatan yang lumayan bagus, dengan *F1-Score* naik menjadi 0.8977.

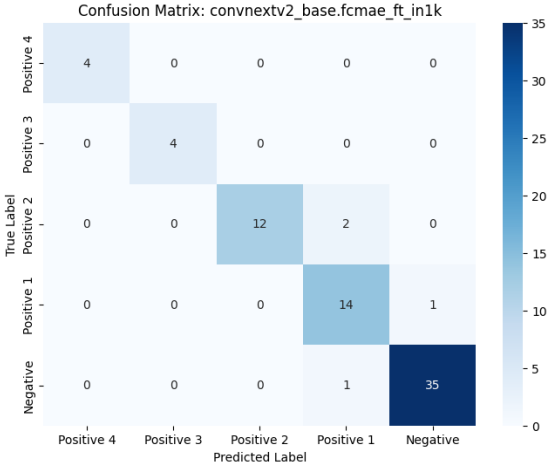
Performa Swin Transformer V2 dan Vision Transformer (ViT) lumayan meningkat, yang mengindikasikan bahwa konfigurasi awalnya sudah lumayan optimal. Untuk analisis kesalahan yang lebih detail, confusion matrix dari semua model setelah optimasi disajikan pada Gambar 6.4.



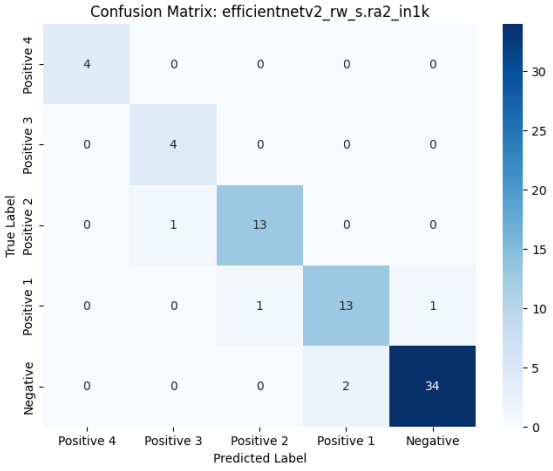
(a)


Confusion Matrix: swinv2_cr_small_224.sw_in1k
True Label
Positive 4
Positive 3
Positive 2
Positive 1
Negative
4 0 0 0 0
0 4 0 0 0
0 0 14 0 0
0 0 0 14 1
0 0 0 1 35
Positive 4
Positive 3
Positive 2
Positive 1
Negative
Predicted Label
35
30
25
20
15
10
5
0

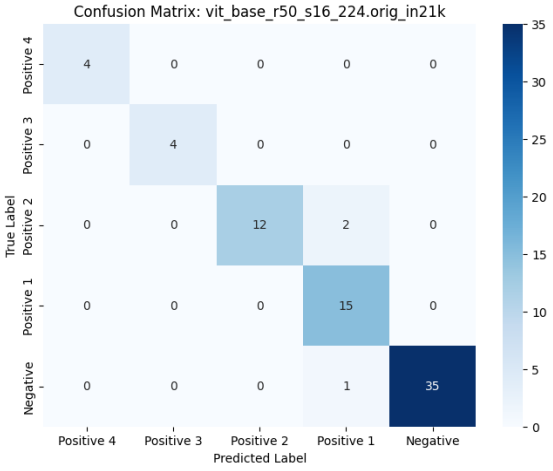
(b)


Confusion Matrix: convnextv2_base.fcmae_ft_in1k
True Label
Positive 4
Positive 3
Positive 2
Positive 1
Negative
4 0 0 0 0
0 4 0 0 0
0 0 12 2 0
0 0 0 14 1
0 0 0 1 35
Positive 4
Positive 3
Positive 2
Positive 1
Negative
Predicted Label
35
30
25
20
15
10
5
0

(c)



(d)



(e)

Gambar 6.4 *Confusion Matrix* Semua Model pada Skenario Optuna (a). RegNetY-3.2GF, (b). Swin Transformer V2, (c). ConvNext V2 Base, (d). EfficientNet V2 Small, (e). Vision Transformer

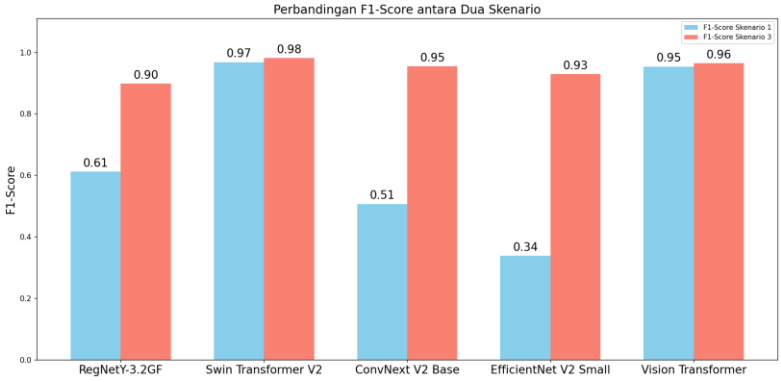
Dari visualisasi ini, terlihat bahwa setelah optimasi, semua model menjadi jauh lebih baik dalam membedakan kelas-kelas yang sulit, meskipun beberapa kebingungan minor antara kelas yang berdekatan masih terjadi. Ini menunjukkan bahwa optimasi *hyperparameter* berhasil meningkatkan kemampuan generalisasi dari hampir semua arsitektur yang diuji.

6.6. Analisis Komparatif dan Diskusi

Setelah didapatkan hasil dari ketiga skenario pengujian, akan dilakukan analisis perbandingan secara menyeluruh untuk menarik wawasan yang lebih dalam pada subbab ini. Diskusi akan difokuskan pada beberapa pertanyaan kunci yang muncul dari hasil eksperimen, seperti dampak fundamental dari optimasi hyperparameter, perbandingan stabilitas dan performa, serta evaluasi ulang terhadap persaingan antara arsitektur CNN dan Transformer.

6.6.1. Dampak Fundamental Optimalisasi Hyperparameter

Salah satu temuan paling signifikan dari kerja praktik ini adalah dampak dramatis *dari hyperparameter tuning*. Gambar 6.4 menunjukkan perbandingan *F1-Score* dari setiap model sebelum dan sesudah proses optimasi, yakni skenario 1 dan skenario 3.



Gambar 6.5 Perbandingan *F1-Score* Sebelum dan Sesudah Optimalisasi *Hyperparameter*

Grafik tersebut secara visual mengonfirmasi bahwa kesimpulan yang ditarik hanya dari performa *baseline* akan sangat tidak akurat. Model EfficientNetV2 dan ConvNeXt, yang pada awalnya dianggap tidak cocok, ternyata menjadi sangat kompetitif setelah konfigurasinya dioptimalkan. Peningkatan *F1-Score* sebesar 0.5906 untuk EfficientNetV2 dan 0.4474 untuk ConvNeXt adalah bukti nyata bahwa potensi sebuah arsitektur tidak dapat dinilai hanya dari satu set konfigurasi saja. Hal ini menegaskan pentingnya melakukan proses tuning yang cermat dalam setiap proyek deep learning untuk memastikan evaluasi yang adil dan untuk membuka potensi sebenarnya dari setiap model.

6.6.2. Stabilitas dan Performa Puncak

Meskipun EfficientNetV2 menunjukkan peningkatan yang luar biasa pada Skenario 3, hasil dari Skenario 2 memberikan perspektif yang berbeda. Dengan standar deviasi *F1-Score* sebesar 0.1138 pada metrik akurasi, EfficientNetV2 terbukti menjadi model yang paling tidak stabil. Ini memunculkan diskusi penting mengenai *trade-off* antara performa puncak dan keandalan.

Meskipun model ini mampu mencapai akurasi tinggi dengan *F1-Score* 0.9284 dengan konfigurasi yang tepat, performanya sangat rentan terhadap perubahan pada data latih. Sebaliknya, Swin Transformer V2 tidak hanya mencapai *F1-Score* tertinggi (0.9811), tetapi juga memiliki standar deviasi terendah (0.0112 pada metrik *F1-Score* di skenario 2), menjadikannya pilihan yang lebih robust dan dapat diandalkan. Untuk aplikasi kritis seperti di bidang medis, model yang stabil dan konsisten seringkali lebih diutamakan daripada model yang performanya fluktuatif meskipun berpotensi tinggi.

6.6.3. Evaluasi Ulang Arsitektur: CNN vs. Transformer

Dengan hasil akhir yang sudah dioptimalkan, terlihat bahwa arsitektur CNN modern seperti ConvNeXt V2 *(F1-Score* 0.9541) dan EfficientNetV2 (*F1-Score* 0.9284) mampu memberikan performa yang sangat kompetitif. Kesenjangan performa yang awalnya tampak besar antara CNN dan *Transformer* pada Skenario 1 hampir sepenuhnya hilang setelah optimasi. Kesimpulannya adalah bahwa pilihan arsitektur menjadi kurang krusial dibandingkan dengan penerapan proses optimasi yang cermat dan menyeluruh.

BAB VII
KESIMPULAN DAN SARAN

Bab ini menyajikan rangkuman dari seluruh rangkaian kegiatan kerja praktik yang telah dilakukan, mulai dari preparasi data hingga evaluasi model. Saran diberikan sebagai masukan untuk pengembangan penelitian atau implementasi sistem di masa mendatang.

7.1. Kesimpulan

Berdasarkan hasil analisis dan evaluasi yang telah diuraikan pada bab-bab sebelumnya, dapat ditarik beberapa kesimpulan utama:

a. Proses preparasi data meliputi pelabelan citra segmentasi ke dalam lima kelas reaksi, perubahan ukuran menjadi 224x224 piksel, normalisasi, serta augmentasi data berupa *horizontal flip* dan *color jitter*. Dataset kemudian dibagi menjadi set pelatihan (80%), validasi (10%), dan pengujian (10%) untuk evaluasi objektif.
b. Pengembangan model dilakukan dengan pendekatan *transfer learning* menggunakan *PyTorch* dan *timm* pada lima arsitektur pre-trained. Teknik *freezing layer* diterapkan, di mana model dilatih menggunakan *optimizer* Adam/AdamW dan *loss function Cross-Entropy*, serta dilengkapi mekanisme *early stopping* dan *learning rate scheduler* untuk pelatihan yang efisien.
c. Hasil evaluasi menunjukkan bahwa Swin Transformer V2 merupakan model dengan performa terbaik secara keseluruhan. Setelah optimasi *hyperparameter*, model ini mencapai *F1-Score* 0.9811 dan terbukti paling stabil dalam pengujian *K-Fold Cross-Validation*. Meskipun arsitektur CNN seperti ConvNeXtV2 dan EfficientNetV2 menunjukkan peningkatan performa yang drastis setelah

tuning, Swin Transformer V2 unggul karena menawarkan kombinasi terbaik antara akurasi, stabilitas, dan efisiensi.

7.2. Saran

Berdasarkan temuan dan pengalaman selama pelaksanaan kerja praktik, berikut adalah beberapa saran untuk pengembangan di masa depan:

a. Kerja praktik ini berfokus pada hanya melatih lapisan akhir saja. Untuk penelitian selanjutnya, disarankan untuk mengeksplorasi teknik fine-tuning di mana beberapa lapisan terakhir dari *backbone* model tidak dilakukan *freezing layer* dan dilatih dengan learning rate yang sangat kecil. Pendekatan ini berpotensi meningkatkan kemampuan model dalam menangkap fitur yang lebih spesifik dari citra uji silang serasi.
b. Performa model deep learning sangat bergantung pada kualitas dan kuantitas data. Disarankan untuk memperluas dataset dengan mengumpulkan lebih banyak sampel, terutama untuk kelas-kelas minoritas atau kasus-kasus ambigu. Pengambilan gambar dari berbagai perangkat dan kondisi pencahayaan yang berbeda juga dapat meningkatkan kemampuan generalisasi model di lingkungan dunia nyata.
c. Model terbaik yang telah dipilih, yaitu Swin Transformer V2, dapat diintegrasikan ke dalam sistem aplikasi *end-to-end*. Saran pengembangan selanjutnya adalah membangun sebuah API untuk pelayanan model ini, yang kemudian dapat dihubungkan dengan antarmuka pengguna untuk penggunaan praktis oleh tenaga medis di laboratorium.

DAFTAR PUSTAKA

[1] Dewan, J.H., Jadhav, H.A., Das, R., Narsale, N., Nambiar, S., Thepade, S.D. & Mhasawade, A., 2023. *Image Classification by Transfer Learning using Pre-Trained CNN Models*. In: IEEE RAEEUCCI 2023. DOI: 10.1109/RAEEUCCI57140.2023.10134069.

[2] Muir, A., n.d. *Crossmatching and Issuing Blood Components*. Newcastle Trust, Blood Transfusion, Blood Sciences.

[3] Ma, J., Hu, C., Zhou, P., Jin, F., Wang, X. & Huang, H., 2023. *Review of Image Augmentation Used in Deep Learning-Based Material Microscopic Image Segmentation*. *Applied Sciences*, 13(11), p.6478. DOI: https://doi.org/10.3390/app13116478.

[4] Rayed, M.E., Islam, S.M.S., Niha, S.I., Jim, J.R., Kabir, M.M. & Mridha, M.F., 2024. *Deep learning for medical image segmentation: State-of-the-art advancements and challenges*. *Informatics in Medicine Unlocked*, 47, p.101504. DOI: https://doi.org/10.1016/j.imu.2024.101504.

[5] Vujović, Ž.Đ., 2021. *Classification Model Evaluation Metrics*. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 12(6). DOI: 10.14569/IJACSA.2021.0120670.

[6] Oktari, A. & Mulyati, L., 2022. *Pengaruh Waktu dan Suhu Penyimpanan Sampel Darah terhadap Hasil Pemeriksaan Uji Silang Serasi (CROSS MATCH)*. *Journal of Indonesian Medical Laboratory and Science*, 3(2), pp.133–145.

[7] Iman, M., Arabnia, H.R. & Rasheed, K., 2023. *A Review of Deep Transfer Learning and Recent Advancements. Technologies,* 11(2), hlm. 40. DOI: https://doi.org/10.3390/technologies11020040.

[8] Title, Book & Islam, Anisul & Das, Lipon. (2024). BlossomNet: A Deep Learning Framework for Accurate Flower Identification. 10.1007/978-3-031-69146-1_10.

[9] Navastara, D.A., Banjarnahor, J.E.Z., Wibawa, I.B.K.R.P., Fatichah, C. and Tambunan, B.A. (2024) **Blood Segmentation in Crossmatch Test Images Using Deep Learning**. *2024 Beyond Technology Summit on Informatics International Conference (BTS-I2C)*, Jember, East Java, Indonesia, 2024, pp. 137-141. doi: 10.1109/BTS-I2C63534.2024.10942293.

[10] Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. arXiv preprint arXiv:1907.10902. https://doi.org/10.48550/arXiv.1907.10902

[11] Tsai, M.-J., Lin, C.-H., Lai, J.-P., & Pai, P.-F. (2024). Using deep learning, Optuna, and digital images to identify necrotizing fasciitis. Electronics, 13(22), 4421. https://doi.org/10.3390/electronics13224421

[12] Wijiyanto, W., Pradana, A. I., Sopingi, & Atina, V. (2024). Teknik K-Fold Cross Validation untuk Mengevaluasi Kinerja Mahasiswa. Jurnal Algoritma, 21(1), 239–248. https://doi.org/10.33364/algoritma/v.21-1.1618

BIODATA PENULIS I

Nama	: Rakha Fathin Izzan Consetta
Tempat, Tanggal Lahir	: Kota Madiun, 30 April 2004
Jenis Kelamin	: Pria
Telepon	: +6285733851405
Email	: rfathin18@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2022
Semester	: 6 (Enam)