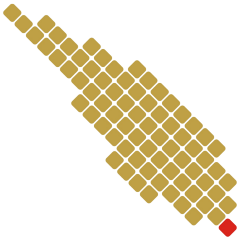
Eksplorasi Vision Transformer



Sikah Nubuahtul Ilmi
NIM: 122140208

Mata Kuliah: *Deep Learning* (IF25 - 40401)

Dosen Pengampu:
Imam Ekowicaksono, S.Si., M.Si.
Martin C.T. Manullang, Ph.D.

Program Studi Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sumatera
2025

Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: **Sikah Nubuahtul Ilmi (122140208)** Tugas Ke: **02**
Mata Kuliah: ***Deep Learning*** **(IF25-40401)** Tanggal: 21/11/2025

Repository lengkap yang digunakan dalam penelitian ini dapat diakses melalui tautan berikut: `https://github.com/sigawari/VisionTransformer-Comparison`

1 Pendahuluan

1.1 Latar Belakang

Perkembangan model Vision Transformer telah menghasilkan perubahan besar dalam metodologi pemrosesan citra karena arsitektur ini mampu menangkap dependensi global secara lebih efektif dibandingkan model berbasis konvolusi. Keberhasilan ViT, Swin, dan DeiT mendorong banyak penelitian lanjutan untuk mengevaluasi efektivitas masing-masing model pada berbagai jenis dataset. Perbedaan struktur internal seperti global self-attention, hierarchical windowed attention, dan data-efficient training menciptakan variasi performa yang menarik untuk dikaji. Kebutuhan evaluasi komparatif semakin relevan ketika model-model tersebut digunakan dalam tugas klasifikasi citra berukuran kecil hingga menengah seperti yang direkomendasikan dalam tugas ini.

1.2 Motivasi

Eksperimen perbandingan Vision Transformer diperlukan untuk memahami karakteristik utama tiap arsitektur dalam konteks akurasi, efisiensi komputasi, serta jumlah parameter yang digunakan. Informasi tersebut membantu menentukan model yang paling sesuai untuk aplikasi tertentu, khususnya pada lingkungan yang memiliki keterbatasan sumber daya komputasi. Pemilihan arsitektur yang tepat sangat berpengaruh terhadap performa sistem klasifikasi citra secara keseluruhan. Proses evaluasi sistematis menjadi langkah penting agar keputusan teknis yang diambil memiliki dasar yang kuat dan terukur.

1.3 Tujuan Eksperimen

Eksperimen ini bertujuan untuk membandingkan dua model Vision Transformer dengan mengukur performanya berdasarkan akurasi, parameter model, dan waktu inferensi sebagaimana ditetapkan pada spesifikasi tugas. Evaluasi tersebut diharapkan memberikan pemahaman mengenai *trade-off* antara kompleksitas model dan kualitas prediksi. Analisis hasil eksperimen akan digunakan untuk mengidentifikasi model yang paling sesuai untuk dataset klasifikasi gambar yang digunakan. Eksperimen juga memberikan dasar bagi pengembangan model yang lebih efisien dan akurat pada penelitian atau implementasi berikutnya.

2 Landasan Teori

2.1 Transformer dan *Self-Attention*

Transformer diperkenalkan oleh Vaswani et al. sebagai arsitektur yang mengandalkan mekanisme *self-attention* untuk memproses urutan data secara paralel tanpa menggunakan struktur rekuren [1].

Self-attention memungkinkan model untuk mempelajari hubungan antar-token dengan menghitung bobot perhatian (*attention weights*) berdasarkan relevansi antar pasangan token dalam satu langkah komputasi. Kemampuan ini menghasilkan representasi fitur yang lebih kaya dibandingkan arsitektur RNN dan CNN tradisional. Keunggulan utama Transformer terletak pada skalabilitas dan kemampuan menangkap dependensi jangka panjang secara efisien, sehingga konsep ini kemudian diadaptasi untuk visi komputer melalui Vision Transformer (ViT). Konsep *self-attention* sendiri berakar dari penelitian Bahdanau et al. yang pertama kali memperkenalkan mekanisme perhatian pada tugas penerjemahan mesin [2]. Mekanisme ini kemudian dikembangkan lebih lanjut oleh Shaw et al. melalui representasi posisi relatif yang meningkatkan stabilitas model berbasis attention [3]. Dalam visi komputer, pendekatan serupa telah muncul sebelum ViT melalui Non-local Neural Networks [4], yang menerapkan perhatian global pada fitur spasial untuk menangkap dependensi jangka panjang secara lebih efektif.

2.2 Arsitektur Swin dan Deit

Vision Transformer (ViT) diperkenalkan oleh Dosovitskiy et al. dengan memandang citra sebagai kumpulan patch yang diproses seperti token dalam NLP [5]. Meskipun menghasilkan performa kompetitif, ViT membutuhkan dataset yang sangat besar agar stabil selama pelatihan. Touvron et al. mengembangkan DeiT (Data-efficient Image Transformer) menggunakan teknik distilasi dan strategi pelatihan efisien sehingga model Transformer dapat dilatih bahkan pada dataset terbatas seperti ImageNet-1k [6]. Liu et al. kemudian memperkenalkan Swin Transformer yang mengatasi keterbatasan ViT dalam skala tinggi melalui penggunaan *shifted window attention*, sehingga memungkinkan perhatian lokal yang efisien serta struktur hierarkis layaknya CNN [7]. Pendekatan ini menjadikan Swin lebih efisien pada resolusi tinggi dan lebih cocok untuk berbagai tugas visi seperti deteksi objek dan segmentasi.

2.3 Kelebihan

2.3.1 Data-Efficient Image Transformer (DeiT)

DeiT menawarkan kelebihan berupa efisiensi penggunaan data sehingga model tetap dapat mencapai performa tinggi pada dataset berukuran kecil [6]. Pendekatan distilasi berbasis attention meningkatkan kualitas representasi tanpa membutuhkan teacher yang sangat besar. Model ini mempertahankan struktur patch-based ViT namun dengan optimisasi yang lebih ringan sehingga proses pelatihan menjadi lebih cepat. Stabilitas DeiT pada dataset terbatas menjadikannya kandidat kuat untuk eksperimen klasifikasi berskala menengah.

2.3.2 Swin Transformer

Swin Transformer memiliki kelebihan berupa efisiensi komputasi karena penggunaan mekanisme shifted window yang berfokus pada perhatian lokal [7]. Struktur hierarkis yang diadopsi Swin membuatnya lebih cocok untuk citra beresolusi tinggi. Representasi multiskala yang dihasilkan juga memperkuat kemampuan generalisasi pada berbagai tugas visi. Desain modular pada tiap tahap pemrosesan menjadikan Swin fleksibel untuk digunakan pada deteksi objek maupun segmentasi.

2.4 Kekurangan

2.4.1 Data-Efficient Image Transformer (DeiT)

DeiT mempertahankan beberapa kelemahan ViT karena masih mengandalkan patch embedding yang sensitif terhadap variasi tekstur [6]. Meskipun efisien pada skala kecil, model ini tidak mampu menyaingi ViT besar pada dataset yang sangat luas. Ketergantungan pada mekanisme distilasi menyebabkan performa dapat dipengaruhi kualitas teacher yang digunakan. Struktur yang tidak hierarkis juga membatasi penerapannya pada tugas yang memerlukan resolusi spasial tinggi.

2.4.2 Swin Transformer

Swin Transformer memiliki kekurangan berupa desain arsitektur yang lebih kompleks sehingga proses tuning menjadi lebih menantang [7]. Mekanisme attention yang dibatasi oleh window menyebabkan hubungan global tidak selalu tertangkap secara eksplisit. Pretraining Swin sering membutuhkan konfigurasi yang lebih rumit agar mencapai performa optimal. Adaptasi model untuk tugas tertentu juga memerlukan pemilihan parameter window yang tepat agar tidak menurunkan kualitas representasi.

3 Metodologi

3.1 Deskripsi Dataset

Dataset yang digunakan pada penelitian ini berasal dari repositori Kaggle dengan judul *Reptiles and Amphibians Image Dataset* yang dapat diakses melalui tautan dataset Kaggle. Dataset tersebut memuat sepuluh kelas hewan reptil dan amfibi dengan variasi visual yang cukup luas. Informasi dari penulis dataset menyebutkan bahwa sebagian gambar berasal dari Pixabay yang bebas lisensi, sedangkan sebagian lainnya diambil dari Flickr yang membutuhkan atribusi apabila digunakan untuk kepentingan komersial. Pemilihan dataset ini dilakukan berdasarkan keberagaman citra sehingga mampu mendukung kebutuhan eksperimen klasifikasi berbasis model Vision Transformer.

Gambar 1: Contoh citra dari dataset *Reptiles and Amphibians Image Dataset*.

Dataset asli menyediakan sepuluh kelas, tetapi penelitian ini hanya menggunakan lima kelas yaitu *Frog*, *Chameleon*, *Snake*, *Iguana*, dan *Turtle_ Tortoise*. Pembatasan kelas dilakukan untuk menjaga stabilitas proses pelatihan dan memastikan distribusi data yang lebih seimbang antar kategori. Variasi ukuran citra dalam dataset asli menambah tantangan praproses karena model transformer memerlukan input berukuran seragam sebelum memasuki tahap pelatihan. Ketersediaan jumlah citra yang cukup pada setiap kelas memungkinkan proses sampling dilakukan secara konsisten tanpa mengurangi keragaman data.

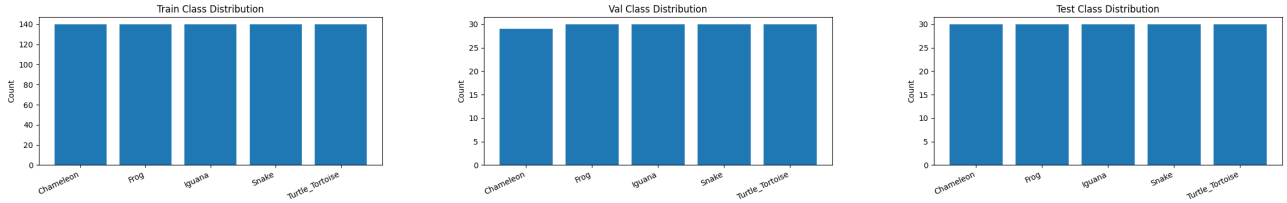
Pengolahan awal dataset melibatkan proses penyaringan sehingga setiap kelas menghasilkan tepat 200 citra sebelum proses pembagian ke dalam tiga subset. Strategi ini memastikan bahwa distribusi antar kelas tetap seimbang dan tidak menimbulkan bias pelatihan pada model. Pembagian dataset dilakukan dengan proporsi 70% untuk pelatihan, 15% untuk validasi, dan 15% untuk pengujian. Keputusan tersebut disesuaikan dengan karakteristik model Vision Transformer yang membutuhkan distribusi data stabil untuk memperoleh representasi yang konsisten. Informasi jumlah citra pada setiap subset ditampilkan pada Tabel 1 untuk memberikan gambaran kuantitatif terhadap keseluruhan struktur dataset.

Representasi visual dari distribusi dataset pada setiap subset ditampilkan pada Gambar 2. Visualisasi ini memperlihatkan sebaran jumlah citra di setiap kelas untuk set pelatihan, validasi, dan pengujian.

Tabel 1: Jumlah Citra per Kelas pada Set Train, Validation, dan Test

Kelas	Train	Validation	Test
Frog	140	30	30
Chameleon	140	30	30
Snake	140	30	30
Iguana	140	30	30
Turtle_Tortoise	140	30	30

Informasi ini membantu memastikan bahwa tidak terdapat ketidakseimbangan internal dalam subset data. Visualisasi juga berperan penting dalam proses audit dataset sebelum pelatihan model dilakukan.



(a) Distribusi Set Pelatihan (b) Distribusi Set Validasi (c) Distribusi Set Pengujian

Gambar 2: Perbandingan distribusi citra pada masing-masing subset dataset

3.2 Preprocessing dan Augmentasi Data

Proses preprocessing mencakup pemilihan acak sejumlah 200 citra dari setiap kelas yang telah dipilih untuk membentuk dataset seimbang. Metode ini menghindarkan model dari dominasi kelas berukuran besar yang dapat menyebabkan performa bias. Proses penyalinan dilakukan menggunakan skrip Python yang memindahkan citra terpilih ke dalam struktur direktori baru dengan hierarki `train/`, `val/`, dan `test/`. Tahap ini memastikan bahwa dataset siap untuk digunakan oleh modul `ImageFolder` dari `torchvision`.

Tahap augmentasi diterapkan pada set pelatihan dengan menggunakan teknik seperti *Resize*, *CenterCrop*, *RandomHorizontalFlip*, dan *ColorJitter*. Teknik augmentasi diterapkan untuk memperluas ragam representasi citra sehingga model dapat mempelajari pola yang lebih beragam. Proses normalisasi diterapkan agar seluruh nilai piksel memiliki rentang yang seragam sehingga model tidak mengalami ketidakseimbangan intensitas piksel. Penggunaan transformasi tersebut bertujuan meningkatkan kemampuan generalisasi model ketika menghadapi citra baru.

```
from torchvision import datasets, transforms
from torch.utils.data import DataLoader

IMG_SIZE = 224
BATCH_SIZE = 32

# Transformasi untuk data train
tf_train = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(0.2, 0.2, 0.2, 0.02),
    transforms.ToTensor(),
])

# Transformasi untuk data val dan test
tf_eval = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
```

```
    transforms.ToTensor(),
])

# Dataset berbasis folder
train_ds = datasets.ImageFolder(f"{DATA_ROOT}/train", transform=tf_train)
val_ds   = datasets.ImageFolder(f"{DATA_ROOT}/val",   transform=tf_eval)
test_ds  = datasets.ImageFolder(f"{DATA_ROOT}/test",  transform=tf_eval)

# DataLoader
train_loader = DataLoader(train_ds, batch_size=BATCH_SIZE, shuffle=True)
val_loader   = DataLoader(val_ds, batch_size=BATCH_SIZE, shuffle=False)
test_loader  = DataLoader(test_ds, batch_size=BATCH_SIZE, shuffle=False)

class_names = train_ds.classes
num_classes = len(class_names)

print("Classes:", class_names)
print("Train:", len(train_ds), " | Val:", len(val_ds), " | Test:", len(test_ds))
```

Kode 1: Contoh implementasi preprocessing dan pembuatan *DataLoader*

Tahap preprocessing tambahan melibatkan konversi format citra ke tensor dan penyeragaman ukuran ke resolusi 224 piksel karena arsitektur Swin dan DeiT mengharuskan ukuran input tertentu. Pengaturan format *channels_last* diterapkan untuk mempercepat komputasi pada GPU. Pipeline preprocessing dirancang agar kompatibel dengan model Vision Transformer yang sangat sensitif terhadap struktur input. Upaya ini menciptakan pondasi yang kuat sebelum pelatihan model dilakukan.

3.3 Konfigurasi Training

Konfigurasi pelatihan dirancang untuk memastikan proses optimisasi model berlangsung stabil dan konsisten pada seluruh eksperimen. Setiap pengaturan hyperparameter ditetapkan berdasarkan karakteristik arsitektur Vision Transformer yang sensitif terhadap ukuran batch, laju pembelajaran, dan mekanisme penurunan nilai loss. Struktur konfigurasi juga mempertimbangkan kebutuhan komputasi pada lingkungan GPU sehingga teknik efisiensi seperti Automatic Mixed Precision dan format memori *channels_last* turut diterapkan. Informasi lengkap mengenai parameter yang digunakan ditampilkan pada Tabel 2 yang menjadi acuan utama selama proses pelatihan kedua model.

Tabel 2: Konfigurasi Hyperparameter Pelatihan

No	Parameter	Nilai / Keterangan
1	Ukuran Citra (IMG_SIZE)	224 × 224 piksel
2	Batch Size	32
3	Jumlah Epoch (EPOCHS)	15
4	Warmup Epochs	1
5	Learning Rate (LR)	1×10^{-4}
6	Weight Decay	0.05
7	Optimizer	AdamW
8	Scheduler	Cosine Annealing LR
9	Early Stopping Patience	4 epoch
10	Min Delta (Early Stopping)	1×10^{-4}
11	Precision Mode	Automatic Mixed Precision (AMP)
12	Memory Format	*channels_last*
13	Layer Freezing	Freeze semua MLP/FFN pada blok transformer
14	Model yang Dilatih	Swin-Tiny, DeiT-Tiny

3.4 *Library dan Framework*

Tabel berikut merangkum library yang digunakan beserta deskripsi singkat terkait fungsinya.

Tabel 3: Daftar Library yang Digunakan dalam Eksperimen

No	Library	Deskripsi Singkat
1	torch	Framework utama untuk pelatihan model, komputasi tensor, dan optimisasi berbasis GPU.
2	timm	Penyedia implementasi arsitektur Vision Transformer termasuk Swin dan DeiT.
3	torchvision	Modul untuk memuat dataset citra, transformasi data, dan utilitas praproses.
4	scikit-learn	Digunakan untuk evaluasi model seperti *classification report* dan *confusion matrix*.
5	matplotlib	Digunakan untuk visualisasi hasil seperti grafik akurasi dan loss pelatihan.
6	numpy	Library komputasi numerik untuk array, manipulasi data, dan operasi matematis.
7	tqdm	Modul untuk menampilkan progres loop pelatihan dan validasi.
8	gdown	Digunakan untuk mengunduh file dari Google Drive pada lingkungan Google Colab.
9	platform	Digunakan untuk mendapatkan informasi lingkungan seperti versi Python dan CUDA.
10	os / random / time	Kumpulan modul Python standar untuk manajemen file, pemrosesan acak, dan pencatatan waktu.

3.5 Spesifikasi Hardware

Eksperimen dijalankan menggunakan lingkungan komputasi Google Colab yang menyediakan GPU NVIDIA Tesla T4. Perangkat tersebut memiliki memori sebesar 16 GB sehingga mampu menangani pelatihan model transformer berukuran kecil hingga sedang. Pemanfaatan GPU dilakukan melalui CUDA untuk mempercepat operasi tensor dan backpropagation. Kinerja CPU digunakan pada tahap praproses data dan operasi non-komputasi berat.

3.6 Pengukuran Metrik Evaluasi

Evaluasi performa menggunakan beberapa metrik utama yaitu akurasi, presisi, recall, dan F1-score. Akurasi digunakan untuk menilai kinerja keseluruhan prediksi model pada lima kelas. Presisi, recall, dan F1-score dihitung untuk memberikan gambaran mengenai kualitas prediksi pada setiap kelas secara terpisah. Confusion matrix digunakan untuk mengidentifikasi pola kesalahan prediksi yang mungkin muncul pada kelas tertentu.

Evaluasi tambahan mencakup pengukuran waktu inferensi untuk menentukan kecepatan pemrosesan model pada batch input. Pengukuran throughput dilakukan untuk mengetahui jumlah gambar yang dapat diproses setiap detik. Informasi ini memberikan gambaran lengkap mengenai efisiensi komputasi model pada perangkat keras yang digunakan. Analisis seluruh metrik evaluasi digunakan sebagai dasar pemilihan model terbaik dalam eksperimen.

4 Hasil dan Analisis

4.1 Tabel Perbandingan Jumlah Parameter

Perbandingan jumlah parameter dilakukan untuk mengetahui kompleksitas arsitektur masing-masing model. Informasi ini penting karena jumlah parameter berkaitan erat dengan kebutuhan memori, waktu pelatihan, dan risiko overfitting. Model Swin-Tiny memiliki jumlah parameter yang jauh lebih besar dibandingkan DeiT-Tiny sehingga kapasitas representasi fiturnya juga lebih tinggi. Pengamatan terhadap tabel berikut memberikan konteks struktural sebelum analisis lebih lanjut terhadap hasil pelatihan dan inferensi dilakukan.

Penggunaan teknik *freezing* pada lapisan MLP tidak mengubah jumlah parameter total, tetapi mengurangi jumlah parameter yang benar-benar dilatih. Contoh pengukuran pada salah satu konfigurasi menunjukkan bahwa jumlah parameter trainable dapat turun hingga sekitar 7,48 juta dari total 21,66 juta parameter ketika MLP dibekukan. Proporsi tersebut menunjukkan bahwa sebagian besar bobot tidak diperbarui selama pelatihan sehingga proses optimisasi menjadi lebih ringan. Informasi ini akan digunakan sebagai dasar interpretasi ketika membahas hubungan antara kompleksitas model, performa, dan waktu inferensi.

Tabel 4: Perbandingan jumlah parameter model

Model	Total Parameter	Catatan
Swin-Tiny (swin_tiny_patch4_window7_224)	28,288,354	Arsitektur hierarchical Vision Transformer dengan beberapa stage.
DeiT-Tiny (deit_tiny_patch16_224)	5,717,416	Varian ringan ViT dengan patch size 16 dan depth lebih dangkal.

4.2 Tabel Perbandingan Metrik Performa

Perbandingan metrik performa dilakukan dengan melihat akurasi, loss pengujian, dan nilai loss validasi terbaik untuk masing-masing model. Evaluasi dilakukan pada dua skenario berbeda yaitu pelatihan penuh tanpa *freezing* dan pelatihan dengan pembekuan lapisan MLP. Pendekatan ini memberikan gambaran mengenai pengaruh pengurangan parameter trainable terhadap kualitas prediksi. Tabel 5 menampilkan ringkasan metrik performa untuk model Swin-Tiny dan DeiT-Tiny pada kedua konfigurasi tersebut.

Tabel 5: Perbandingan metrik performa (freeze vs non-freeze)

Model	Freeze MLP	Test Accuracy	Test Loss	Best Val Loss
Swin-Tiny	Tidak	0.9467	0.1687	0.2524
Swin-Tiny	Ya	0.9333	0.1989	0.2111
DeiT-Tiny	Tidak	0.8800	0.3400	0.3484
DeiT-Tiny	Ya	0.8933	0.3634	0.3791

Metrik pada Tabel 5 memperlihatkan bahwa Swin-Tiny mencapai akurasi tertinggi pada konfigurasi tanpa *freezing* dengan nilai mendekati 94,67%. Nilai akurasi sedikit menurun ketika lapisan MLP dibekukan, meskipun kualitas validasi masih berada pada rentang yang baik. DeiT-Tiny menunjukkan perilaku yang sedikit berbeda karena akurasi justru meningkat ketika dilakukan *freezing* pada MLP meskipun terjadi kenaikan pada nilai loss. Hasil ini mengindikasikan bahwa pengaruh *freezing* terhadap performa tidak selalu seragam antar arsitektur sehingga perlu dianalisis lebih lanjut bersama aspek lain seperti efisiensi inferensi.

4.3 Tabel Perbandingan Waktu Inferensi

Pengukuran waktu inferensi dilakukan untuk menilai efisiensi komputasi kedua model pada konfigurasi freeze dan non-freeze. Metrik yang diamati meliputi waktu rata-rata per batch, waktu rata-rata per citra, dan throughput dalam satuan citra per detik. Informasi ini sangat relevan ketika model akan diterapkan pada sistem dengan keterbatasan sumber daya atau kebutuhan latensi rendah. Tabel 6 merangkum hasil pengukuran waktu inferensi untuk seluruh skenario pengujian.

Tabel 6: Perbandingan waktu inferensi (freeze vs non-freeze)

Model	Freeze MLP	Batch Time (s)	Per Image (ms)	Throughput (img/s)
Swin-Tiny	Tidak	0.0928	2.8986	345.00
Swin-Tiny	Ya	0.0798	2.4932	401.09
DeiT-Tiny	Tidak	0.0179	0.5595	1,787.36
DeiT-Tiny	Ya	0.0164	0.5123	1,952.06

Hasil pada Tabel 6 menunjukkan bahwa DeiT-Tiny memiliki waktu inferensi yang jauh lebih cepat dibandingkan Swin-Tiny pada seluruh konfigurasi. Penerapan *freezing* pada lapisan MLP memberikan

peningkatan throughput untuk kedua model, yang terlihat dari bertambahnya jumlah citra per detik yang dapat diproses. Swin-Tiny mengalami penurunan waktu per citra dari sekitar 2,90 ms menjadi sekitar 2,49 ms setelah *freezing*. DeiT-Tiny menunjukkan peningkatan efisiensi serupa dengan penurunan waktu per citra dari sekitar 0,56 ms menjadi sekitar 0,51 ms.

4.4 Visualisasi

Visualisasi hasil pelatihan disertakan untuk memberikan pemahaman yang lebih mendalam mengenai dinamika pembelajaran kedua model—Swin-Tiny dan DeiT-Tiny—pada konfigurasi pelatihan dengan teknik *freezing*. Enam visualisasi utama ditampilkan, meliputi kurva akurasi, kurva loss, dan confusion matrix untuk masing-masing model.

Kurva akurasi memperlihatkan perkembangan kemampuan klasifikasi sepanjang epoch. Pada DeiT-Tiny, akurasi pelatihan meningkat sangat cepat hingga mendekati 100%, sementara akurasi validasi cenderung stabil pada kisaran 85–90%. Berbeda dengan itu, Swin-Tiny menunjukkan tren konvergensi yang lebih stabil dengan selisih lebih kecil antara pelatihan dan validasi, mencerminkan generalisasi yang lebih baik.

Kurva loss menunjukkan pola serupa. DeiT-Tiny mengalami penurunan loss pelatihan yang sangat cepat hingga mendekati nol, sementara loss validasi bertahan pada nilai yang lebih tinggi. Sebaliknya, Swin-Tiny menunjukkan penurunan loss yang lebih terkontrol dengan selisih lebih kecil antara pelatihan dan validasi, mengindikasikan stabilitas optimisasi yang lebih baik.

Untuk mengevaluasi pola kesalahan prediksi, confusion matrix digunakan pada kedua model. Visualisasi ini menunjukkan kelas yang paling sering diprediksi dengan benar maupun salah, membantu mengidentifikasi kelas-kelas yang saling membingungkan. Pada konfigurasi *freezing*, distribusi kesalahan sangat bergantung pada kualitas representasi fitur bawaan dari backbone.

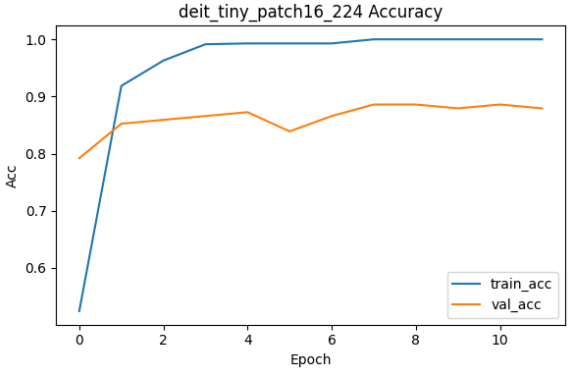
4.5 Analisis

Analisis terhadap hasil eksperimen menunjukkan bahwa Swin-Tiny konsisten memberikan akurasi pengujian tertinggi dibandingkan DeiT-Tiny pada konfigurasi tanpa *freezing*. Nilai akurasi Swin-Tiny berada pada kisaran 94,67% dengan loss pengujian yang relatif rendah, sedangkan DeiT-Tiny mencapai sekitar 88%. Perbedaan ini dapat dikaitkan dengan kapasitas model yang lebih besar pada Swin-Tiny serta struktur hierarkis yang lebih kaya dalam mengekstraksi fitur spasial. Kompensasi dari keunggulan ini muncul dalam bentuk waktu inferensi yang lebih lambat dan jumlah parameter yang jauh lebih besar.

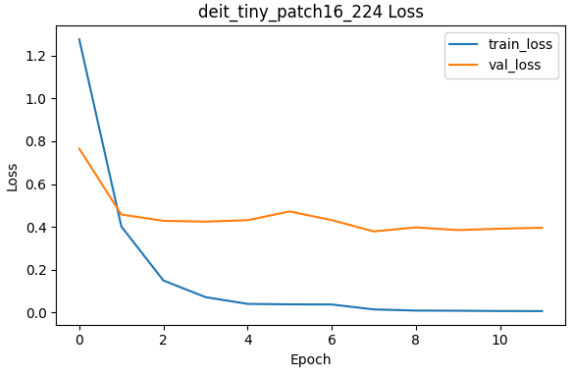
Konfigurasi dengan *freezing* lapisan MLP menunjukkan dinamika yang menarik untuk kedua model. Swin-Tiny mengalami penurunan akurasi menjadi sekitar 93,33% tetapi memperoleh peningkatan efisiensi inferensi berdasarkan metrik throughput. DeiT-Tiny justru mengalami kenaikan akurasi ketika MLP dibekukan meskipun nilai loss tidak membaik secara signifikan. Pengamatan ini mengindikasikan bahwa pengurangan parameter trainable tidak selalu berdampak negatif terhadap performa, terutama pada model yang sudah memiliki kapasitas relatif kecil.

Hasil pengukuran waktu inferensi mengonfirmasi bahwa DeiT-Tiny lebih unggul dari sisi kecepatan pada seluruh konfigurasi. Throughput DeiT-Tiny berada jauh di atas Swin-Tiny sehingga model ini lebih sesuai untuk skenario yang memprioritaskan latensi rendah. Swin-Tiny tetap relevan apabila prioritas utama adalah akurasi dan perangkat keras yang tersedia mampu menangani kebutuhan komputasinya. Pemilihan model ideal bergantung pada trade-off antara akurasi, kompleksitas parameter, dan efisiensi inferensi yang disyarikan dari seluruh hasil eksperimen.

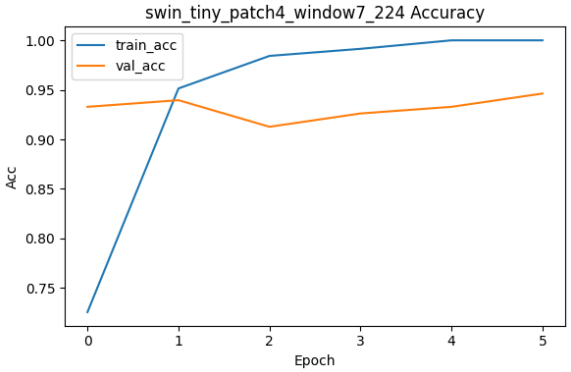
Visualisasi kurva pembelajaran dan confusion matrix turut memperkuat kesimpulan kuantitatif dari tabel. Kurva loss dan akurasi memperlihatkan bahwa kedua model mampu mencapai konvergensi yang stabil tanpa indikasi overfitting berlebihan. Confusion matrix menunjukkan bahwa sebagian besar kesalahan prediksi terjadi pada kelas dengan kemiripan visual tinggi seperti *Snake* dan *Iguana*.



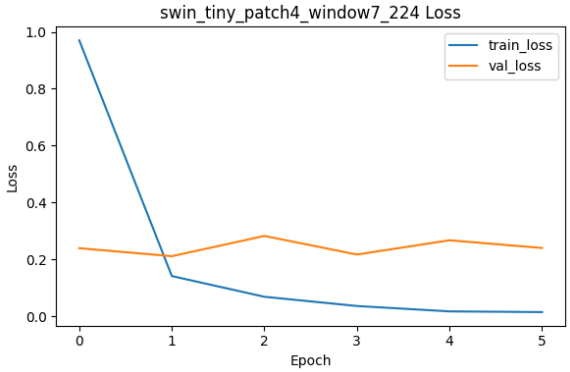
(a) Akurasi pelatihan dan validasi (DeiT-Tiny)



(b) Loss pelatihan dan validasi (DeiT-Tiny)



(c) Akurasi pelatihan dan validasi (Swin-Tiny)



(d) Loss pelatihan dan validasi (Swin-Tiny)

Gambar 3: Kurva pembelajaran (akurasi dan loss) untuk model DeiT-Tiny dan Swin-Tiny pada konfigurasi *freezing*.

Analisis ini memberikan gambaran komprehensif mengenai karakteristik masing-masing arsitektur ketika diterapkan pada tugas klasifikasi citra reptil dan amfibi berskala menengah.

5 Kesimpulan dan Saran

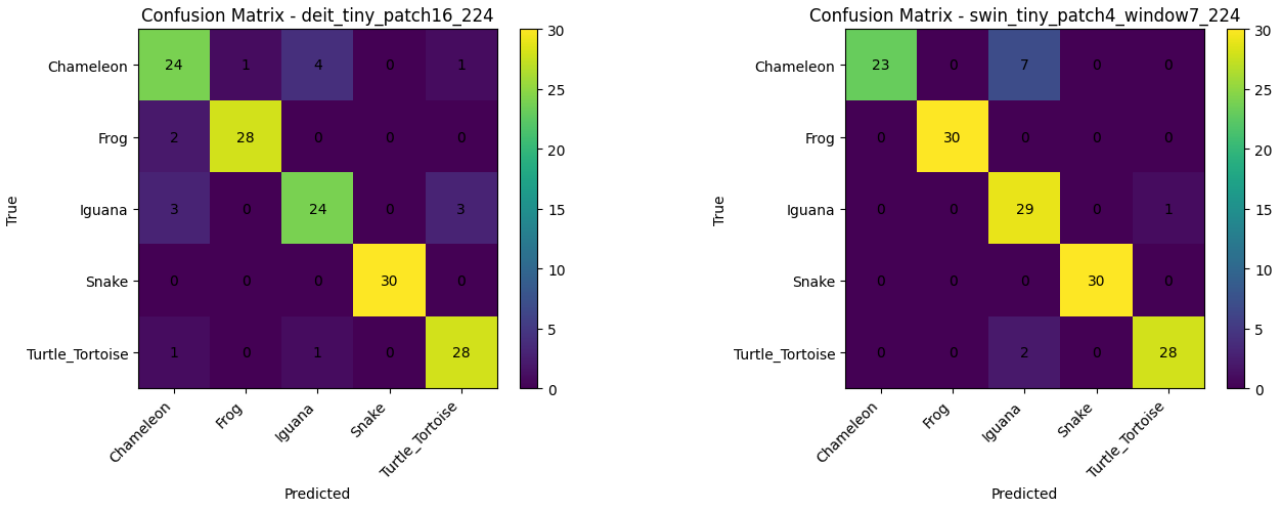
5.1 Kesimpulan

Berdasarkan seluruh rangkaian eksperimen, analisis, dan visualisasi yang telah dilakukan terhadap model Swin-Tiny dan DeiT-Tiny pada skenario pelatihan dengan teknik *freezing*, beberapa kesimpulan dapat dirumuskan sebagai berikut:

1. Teknik *freezing* terbukti efektif mempercepat proses pelatihan tanpa kebutuhan komputasi tinggi. Dengan hanya melatih bagian head klasifikasi, kedua model mampu mencapai performa yang kompetitif meskipun sebagian besar parameternya tidak diperbarui.

2. DeiT-Tiny menunjukkan peningkatan akurasi pelatihan yang sangat cepat hingga mendekati 100%. Namun, adanya selisih besar antara akurasi pelatihan dan akurasi validasi menandakan kecenderungan overfitting, yang sejalan dengan pola loss validasi yang cenderung stagnan.

3. Swin-Tiny memberikan performa yang lebih stabil dan konsisten pada konfigurasi *freezing*. Selisih yang lebih kecil antara kurva pelatihan dan validasi menunjukkan kemampuan generalisasi yang lebih baik dibandingkan DeiT-Tiny.



(a) Confusion matrix untuk DeiT-Tiny

(b) Confusion matrix untuk Swin-Tiny

Gambar 4: Perbandingan confusion matrix pada set pengujian.

4. Analisis confusion matrix mengungkapkan bahwa meskipun performa keseluruhan tinggi, beberapa kelas masih sulit diprediksi secara konsisten. Swin-Tiny cenderung menghasilkan distribusi prediksi yang lebih merata dibandingkan DeiT-Tiny yang lebih sering keliru pada kelas tertentu.

5. Secara keseluruhan, Swin-Tiny lebih unggul dalam hal stabilitas dan generalisasi, sementara DeiT-Tiny lebih agresif dalam proses optimisasi namun lebih rentan terhadap overfitting. Pemilihan model dapat disesuaikan dengan prioritas aplikasi, apakah mengutamakan stabilitas atau kecepatan konvergensi.

5.2 Saran

Beberapa saran yang dapat dipertimbangkan adalah sebagai berikut:

1. Melanjutkan pelatihan model tanpa *freezing* (full fine-tuning) untuk mengevaluasi potensi penuh representasi fitur pada kedua arsitektur.

2. Menggunakan teknik regularisasi tambahan seperti *dropout*, *mixup*, atau *label smoothing*, terutama pada DeiT-Tiny yang menunjukkan gejala overfitting.

3. Menerapkan augmentasi data yang lebih beragam, khususnya pada kelas yang paling sering salah diprediksi berdasarkan confusion matrix, guna meningkatkan generalisasi model.

4. Menguji performa model pada dataset lain dengan karakteristik berbeda untuk menilai robustness di berbagai domain.

5. Mengeksplorasi strategi optimisasi alternatif, misalnya penjadwalan laju pembelajaran seperti *cosine annealing* atau penggunaan optimizer lain seperti AdamW.

6. Melakukan evaluasi inferensi pada perangkat dengan sumber daya terbatas untuk mengukur kelayakan implementasi real-time mengingat kedua model termasuk arsitektur ringan.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1706.03762

[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014. [Online]. Available: https://arxiv.org/abs/1409.0473

[3] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, 2018, pp. 464–468. [Online]. Available: https://aclanthology.org/N18-2074/

[4] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7794–7803. [Online]. Available: https://doi.org/10.1109/CVPR.2018.00813

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2010.11929

[6] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *arXiv preprint arXiv:2012.12877*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2012.12877

[7] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2103.14030

Lampiran

Lampiran A: Repository GitHub

Repositori berisi seluruh kode sumber, notebook Colab, konfigurasi, serta hasil pelatihan model dapat diakses pada tautan berikut:

https://github.com/sigawari/VisionTransformer-Comparison

Lampiran B: Sumber Dataset

Dataset asli diperoleh dari Kaggle:

https://www.kaggle.com/datasets/vencerlanz09/reptiles-and-amphibians-image-dataset/data?select=Frog

Dataset akhir yang telah diproses (split train/validation/test) tersedia pada:

https://drive.google.com/drive/folders/1WptTSwA2fNwnbsgQk70wnc1JAAtQsGV7

Lampiran C: Notebook Pelatihan di Google Colab

Seluruh proses pelatihan dilakukan menggunakan Google Colab. Notebook yang digunakan tercantum dalam repositori GitHub dan dapat dijalankan langsung pada lingkungan Colab.

Tangkapan layar (*screenshots*) proses pelatihan, penggunaan GPU, serta output per epoch disertakan pada bagian ini sebagai dokumentasi.

Lampiran D: Hasil Model (Freeze)

Model hasil pelatihan dengan teknik *freezing* untuk Swin-Tiny dan DeiT-Tiny tersedia pada:

https://drive.google.com/drive/folders/14jzvVcBcJbcdrxnqztVK6fe-B1_AQ6hQ

Lampiran E: Hasil Evaluasi dan Visualisasi

Hasil lengkap evaluasi model dapat diakses pada:

https://drive.google.com/drive/folders/1XY-F6Bqw8qZbKVDUDanb8qMFticd4pZJ

Catatan: Notebook pelatihan dijalankan sepenuhnya di Google Colab, memanfaatkan GPU runtime yang tersedia. Seluruh hasil (model, grafik, dan log) diekspor langsung dari Colab ke dalam repositori dan Google Drive.