Program Studi Teknik Informatika
Institut Teknologi Sumatera

LAPORAN TUGAS PEMBELAJARAN MENDALAM

Perbandingan Model ViT-B/16 dan Swin-Tiny pada Klasifikasi Gambar Hewan Menggunakan Transfer Learning

Nama Mahasiswa	: Reynaldi Cristian Simamora
NIM	: 122140116
Program Studi	: Teknik Informatika
Mata Kuliah	: Pembelajaran Mendalam (IF25-40401)
Tugas Ke	: Minggu ke-12
Dosen Pengampu	: Rahman Indra Kesuma, S.Kom. M.Cs., Meida Cahyo Untoro, S.Kom., M.Kom
Tanggal	: 21 November 2025

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SUMATERA
2025

1 PENDAHULUAN

Perkembangan pesat dalam bidang *computer vision* selama satu dekade terakhir telah banyak didorong oleh dominasi *Convolutional Neural Networks* (CNN). CNN menjadi fondasi utama dalam berbagai aplikasi visi, berkat kemampuannya dalam mengekstraksi fitur lokal melalui operasi konvolusi. Namun, transformasi besar terjadi ketika arsitektur Transformer, yang awalnya populer pada ranah *Natural Language Processing* (NLP), mulai diterapkan untuk pemrosesan visual. Kehadiran *Vision Transformer* (ViT) menunjukkan bahwa mekanisme *self-attention* mampu menggantikan peran konvolusi dan mencapai performa kompetitif pada klasifikasi gambar berskala besar [1].

Keberhasilan ViT membuka arah baru dalam pengembangan model visi modern. Tidak seperti CNN yang berfokus pada pemodelan lokal, ViT mampu menangkap relasi global antar-patch pada keseluruhan gambar. Raghu et al. [2] menunjukkan bahwa ViT mengekstraksi representasi fitur dengan pola yang berbeda secara signifikan dari CNN, menekankan pentingnya pemahaman lebih mendalam terhadap karakteristiknya. Meski demikian, ViT juga memiliki keterbatasan, terutama dalam pemahaman struktur lokal dan kebutuhan pretraining skala besar untuk mencapai performa optimal.

Untuk mengatasi berbagai tantangan tersebut, model Transformer berbasis visi lain dikembangkan. Salah satu yang paling menonjol adalah *Swin Transformer*, yang diperkenalkan oleh Liu et al. [3]. Swin Transformer memperkenalkan mekanisme *shifted window attention* yang memungkinkan komputasi hierarkis dan efisien pada gambar beresolusi tinggi, sekaligus mempertahankan kemampuan modeling global dari arsitektur Transformer.

Perbedaan mendasar dalam arsitektur—mulai dari patch embedding, strategi attention, hingga bentuk hierarki fitur—menjadikan perbandingan performa antara ViT dan Swin Transformer sebagai langkah penting dalam memahami *trade-off* kedua model. Evaluasi diperlukan untuk menentukan bagaimana model-model ini berbeda dari segi akurasi, kompleksitas parameter, efisiensi komputasi, serta kecepatan inferensi.

Pada eksperimen ini, penulis melakukan evaluasi komparatif antara dua model Vision Transformer, yaitu *ViT Base Patch-16 (ViT-B/16)* dan *Swin Transformer Tiny*. Keduanya diuji menggunakan subset CIFAR-10 yang disederhanakan menjadi lima kelas (CIFAR-5), yaitu *bird*, *cat*, *deer*, *dog*, dan *frog*. Proses eksperimen meliputi *fine-tuning* menggunakan bobot pretrained ImageNet, pelatihan selama beberapa epoch, evaluasi metrik performa, analisis kesalahan, serta pengukuran waktu inferensi. Melalui penelitian ini, diharapkan diperoleh pemahaman yang lebih komprehensif mengenai karakteristik masing-masing model serta rekomendasi penggunaannya berdasarkan kebutuhan aplikasi.

2 LANDASAN TEORI

2.1 Arsitektur Transformer

Transformer pertama kali diperkenalkan oleh Vaswani et al. (2017) melalui paper "Attention Is All You Need"[4]. Arsitektur ini menggantikan mekanisme rekuren (RNN/LSTM) dan konvolusi (CNN) dalam pemrosesan sekuens dengan memanfaatkan mekanisme self-attention sebagai komponen inti. Self-attention memungkinkan setiap elemen input (token) untuk memperhatikan elemen lain dan memodelkan dependensi global secara paralel. Komponen Utama Transformer meliputi:

- **Multi-Head Self-Attention:** Memungkinkan model untuk fokus pada berbagai bagian input secara bersamaan, meningkatkan kapasitas representasi.
- **Feed-Forward Neural Networks:** Lapisan fully connected yang diterapkan secara independen pada setiap posisi token.
- **Positional Encoding:** Menyediakan informasi urutan token karena Transformer tidak memiliki mekanisme bawaan untuk menangani urutan.

- **Layer Normalization dan Residual Connections:** Membantu stabilisasi pelatihan dan mempercepat konvergensi.

Kelebihan Transformer memiliki kemampuan pemodelan konteks global, paralelisasi penuh tanpa ketergantungan sekuens, mudah diskalakan ke model besar.

2.2 Self-Attention pada Transformer

Self-attention merupakan mekanisme inti pada arsitektur Transformer yang memungkinkan setiap elemen dalam sebuah sekuens memperhatikan elemen lainnya secara langsung. Konsep ini diperkenalkan oleh Vaswani et al. (2017) pada makalah *Attention Is All You Need* sebagai pengganti operasi rekuren dan konvolusi dalam pemodelan sekuens. Pada domain visi komputer, self-attention berfungsi untuk memodelkan hubungan spasial antarpatch sehingga model dapat memahami konteks global secara lebih efektif dibandingkan pendekatan berbasis CNN. [4]

2.2.1 Mekanisme Dasar Self-Attention

Dalam self-attention, setiap token direpresentasikan oleh tiga vektor: Query (Q), Key (K), dan Value (V). Hubungan antar token dihitung menggunakan fungsi attention sebagai berikut:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Normalisasi menggunakan $\sqrt{d_k}$ bertujuan untuk menjaga stabilitas numerik pada operasi dot-product. Mekanisme ini memungkinkan model untuk mempelajari bobot ketergantungan antar token, sehingga setiap token dapat mengumpulkan informasi dari token lain yang relevan.

2.2.2 Multi-Head Self-Attention

Self-attention diperluas menjadi *Multi-Head Self-Attention* (MHSA), di mana beberapa head diproses secara paralel. Setiap head memiliki proyeksi Q, K, dan V yang berbeda sehingga mampu menangkap berbagai pola relasi dalam subruang representasi yang berbeda. Hasil dari semua head dikonkatenasi dan diproyeksikan ulang ke dimensi semula.

Keuntungan MHSA meliputi:

- kemampuan menangkap beragam pola ketergantungan,
- peningkatan stabilitas pemodelan,
- peningkatan kapasitas representasi tanpa peningkatan komputasi signifikan.

2.2.3 Self-Attention pada Domain Visi

Pada Vision Transformer (ViT), gambar dibagi menjadi patch yang dianggap sebagai token sehingga self-attention dapat memodelkan hubungan global antarpatch. Hal ini memberikan receptive field global sejak lapisan pertama.

Pada Swin Transformer, self-attention dihitung dalam jendela lokal (*Window-based MSA*) untuk meningkatkan efisiensi komputasi. Interaksi global kemudian diperoleh melalui mekanisme *Shifted Window MSA*, yang menggeser jendela pada lapisan berikutnya untuk memungkinkan pertukaran informasi antarjendela.

2.2.4 Keunggulan Self-Attention

Self-attention memiliki keunggulan sebagai berikut:

1. Memodelkan konteks global tanpa batasan receptive field lokal.
2. Memberikan bobot dinamis berdasarkan kesamaan konten, tidak seperti kernel konvolusi yang statis.
3. Fleksibel untuk arsitektur bertingkat maupun datar.

2.2.5 Keterbatasan Self-Attention

Mekanisme ini juga memiliki beberapa kelemahan, antara lain:

- Kompleksitas komputasi kuadratik $O(N^2)$ pada ViT.
- Membutuhkan dataset berskala besar untuk pelatihan stabil.
- Kurang optimal dalam menangkap informasi lokal tanpa mekanisme tambahan.

2.3 Vision Transformer (ViT)

Vision Transformer (ViT) diperkenalkan oleh Dosovitskiy et al. (2021) melalui paper "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale"[1]. ViT mengadaptasi arsitektur Transformer dari NLP ke ranah visi dengan mengubah gambar menjadi serangkaian patch kecil yang diproses seperti token teks. Proses utama dalam ViT meliputi:

- **Patch Embedding:** Gambar dibagi menjadi patch berukuran tetap (misalnya 16x16 piksel), yang kemudian di-flatten dan diproyeksikan ke dimensi embedding menggunakan layer linear.
- **Positional Encoding:** Ditambahkan ke patch embeddings untuk mempertahankan informasi spasial.
- **Transformer Encoder:** Patch embeddings yang telah ditambahkan positional encoding diproses melalui beberapa lapisan Transformer encoder standar.
- **Classification Head:** Token klasifikasi khusus ditambahkan ke urutan patch embeddings, dan outputnya digunakan untuk prediksi kelas.

Kelebihannya mampu menangkap hubungan global secara langsung, arsitektur sederhana namun sangat kuat, dan skalabilitas tinggi. Kelemahannya sensitif terhadap jumlah data yang dapat menyebabkan performa turun pada dataset kecil, kurang menangkap informasi lokal dibanding CNN, dan biaya komputasi meningkat seiring besarnya resolusi input ($O(N^2)$).

2.4 Swin Transformer

Swin Transformer [3] merupakan Vision Transformer generasi berikutnya yang mengatasi keterbatasan ViT dengan memperkenalkan shifted window attention dan struktur hierarchical. Model ini mampu bekerja secara efisien pada resolusi tinggi, sehingga banyak digunakan untuk deteksi objek dan segmentasi. Fitur utama Swin Transformer meliputi:

- **Hierarchical Feature Maps:** Menghasilkan representasi fitur pada berbagai skala melalui patch merging, mirip dengan CNN.
- **Shifted Window Attention:** Memecah gambar menjadi jendela-jendela kecil untuk self-attention, kemudian menggeser jendela pada lapisan berikutnya untuk memungkinkan interaksi antar jendela.

- **Efficient Computation:** Kompleksitas komputasi yang lebih rendah (O(N)) dibandingkan ViT, membuatnya lebih efisien untuk gambar beresolusi tinggi.

Untuk tahapannya mirip dengan ViT, namun dengan modifikasi pada mekanisme attention dan struktur hierarkis. Berikut adalah tahap dari Swin Transformer:

- **Patch Partitioning:** Gambar dibagi menjadi patch kecil yang kemudian diproses melalui beberapa stage.
- **Swin Transformer Blocks:** Setiap stage terdiri dari beberapa blok Swin Transformer yang menggunakan shifted window attention.
- **Patch Merging:** Setelah setiap stage, patch-patch digabungkan untuk mengurangi resolusi dan meningkatkan dimensi fitur.
- **Classification Head:** Output akhir dari stage terakhir digunakan untuk prediksi kelas melalui layer fully connected.

Kelebihannya efisien pada resolusi tinggi, menangkap informasi lokal dan global, serta fleksibel untuk berbagai tugas visi. Kelemahannya arsitektur lebih kompleks dibanding ViT, dan memerlukan tuning hyperparameter yang cermat.

2.5 Transfer Learning

Transfer learning adalah teknik pembelajaran mesin di mana pengetahuan dari model yang telah dilatih pada dataset besar digunakan kembali untuk tugas lain dengan dataset yang lebih kecil. Pada visi komputer, model biasanya pretrained menggunakan ImageNet-1k atau ImageNet-21k untuk mempercepat konvergensi dan meningkatkan performa, terutama ketika data pelatihan terbatas [5]. Pendekatan umum dalam transfer learning meliputi:

- **Feature Extraction:** Menggunakan model pretrain ed sebagai ekstraktor fitur tetap, di mana hanya lapisan klasifikasi akhir yang dilatih ulang pada dataset target.
- **Fine-Tuning:** Melatih ulang seluruh model atau sebagian besar lapisan dengan learning rate yang lebih rendah, memungkinkan model untuk menyesuaikan diri dengan karakteristik dataset target.
- **Pretrained Model:** Model yang telah dilatih pada dataset besar yang digunakan sebagai ekstraktor fitur tetap.

3 METODOLOGI

Bagian ini menjelaskan prosedur eksperimen yang dilakukan, meliputi spesifikasi perangkat, dataset, preprocessing, arsitektur model, konfigurasi pelatihan, serta metode evaluasi. Seluruh langkah disusun agar memenuhi ketentuan metodologi yang dipersyaratkan pada dokumen tugas Vision Transformer[**?**].

3.1 Spesifikasi Perangkat dan Library

Eksperimen dilakukan menggunakan lingkungan Google Colab dengan akselerasi GPU. Perangkat keras yang digunakan terdiri dari:

- GPU: NVIDIA Tesla T4 / V100 (CUDA enabled)
- VRAM: 16 GB

- RAM: 12–25 GB
- Sistem Operasi: Linux (runtime Google Colab)

Seluruh implementasi menggunakan library berikut:

- **PyTorch** untuk proses training dan inferensi model
- **torchvision** untuk dataset dan transformasi gambar
- **timm** (PyTorch Image Models) untuk model ViT-B/16 dan Swin-Tiny
- **scikit-learn** untuk perhitungan metrik performa
- **numpy**, **pandas** untuk analisis data
- **matplotlib** untuk visualisasi kurva training dan confusion matrix

Instalasi library utama dilakukan melalui:

```
!pip install timm scikit-learn pandas matplotlib --quiet
```

3.2 Dataset

Dataset yang digunakan pada penelitian ini adalah **CIFAR-10**, yaitu dataset citra berukuran 32×32 piksel dengan 10 kelas objek: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck*. Pada penelitian ini, digunakan subset berisi **5 kelas** yang dipilih berdasarkan nama kelas, yaitu:

- bird
- cat
- deer
- dog
- frog

Pemilihan kelas dilakukan dengan cara mengambil indeks kelas dari daftar lengkap CIFAR-10, kemudian memetakan ulang label menjadi rentang 0–4. Dataset kemudian dibungkus dalam kelas `CIFAR5`, yang menyeleksi gambar serta melakukan remapping label. Seluruh data training dipisahkan kembali menjadi **training set** dan **validation set** menggunakan *stratified-like random split* dengan rasio 80:20.

Pembagian Dataset

Seluruh subset dataset kemudian dimuat menggunakan `DataLoader` dengan **batch size 32**

Pendekatan ini memastikan bahwa dataset telah melalui proses seleksi kelas, remapping label, serta pembagian train–validation secara sistematis, dan siap digunakan untuk pelatihan model Vision Transformer.

Pemilihan subset kelas dilakukan melalui indeks kelas menggunakan kelas kustom `CIFAR5`, yaitu 'bird','cat','deer','dog','frog'.

Sehingga dataset akhir yang digunakan memiliki 5 kelas dengan distribusi yang seimbang.

- Jumlah kelas: 5 (bird, cat, deer, dog, frog)
- Jumlah citra per kelas: 6.000
- Total citra: 30.000

- Train set: 20.000 citra
- Test set: 5.000 citra
- Validation set: 5.000 citra

3.3 Preprocessing Data

Tahap preprocessing dilakukan mengikuti pipeline transformasi yang diimplementasikan pada kelas `CIFAR5`. Seluruh citra terlebih dahulu dikonversi menjadi objek *PIL Image* sebelum diproses oleh transformasi lainnya. Adapun tahapan preprocessing adalah sebagai berikut:

1. **ToPILImage**: Mengubah array citra CIFAR-10 menjadi format *PIL Image* agar kompatibel dengan transformasi lanjutan.
2. **Resize**: Seluruh citra diubah ukurannya menjadi 224×224 piksel menggunakan interpolasi `PIL`, menyesuaikan resolusi input yang dibutuhkan oleh ViT-B/16 dan Swin-Tiny.
3. **Random Horizontal Flip (augmentasi khusus training)**: Augmentasi ini hanya diterapkan pada mode *train* untuk meningkatkan generalisasi model. Pada mode *test*, transformasi ini digantikan oleh `Lambda` agar citra tidak mengalami perubahan.
4. **ToTensor**: Mengubah citra ke dalam format tensor PyTorch dengan nilai piksel berada pada rentang [0, 1].
5. **Normalisasi**: Tensor dinormalisasi berdasarkan mean dan standar deviasi ImageNet, yaitu:

   $$\text{mean} = [0.485, 0.456, 0.406], \qquad \text{std} = [0.229, 0.224, 0.225].$$

   Normalisasi ini dipilih karena kedua model yang digunakan adalah pretrained ImageNet, sehingga input perlu diselaraskan dengan distribusi data awal model.

Transformasi dilakukan menggunakan `torchvision.transforms`. Tidak ada augmentasi tambahan yang digunakan agar perbandingan arsitektur lebih adil.

3.4 Arsitektur Model

Dua model Vision Transformer dibandingkan pada eksperimen ini.

1. **Vision Transformer Base Patch-16 (ViT-B/16)** Model ini merupakan arsitektur transformer murni untuk visi, dengan karakteristik:
   - Patch size 16×16
   - Embedding dimension 768
   - 12 encoder layers
   - 12 attention heads
   - Pretrained ImageNet-1k
   - 85.802.501
2. **Swin Transformer Tiny** Model ini merupakan transformer hierarkis dengan mekanisme *shifted window attention*. Karakteristik utama:
   - Window size 7×7
   - Patch merging hierarkis (56→28→14→7)
   - Pretrained ImageNet-1k
   - 27.523.199

Model dimuat menggunakan library `timm` dan dirangkum menggunakan `torchsummary`.

3.5 Konfigurasi Pelatihan

Proses pelatihan dilakukan melalui fungsi `run_experiment`, yang menjalankan prosedur *fine-tuning* pada model Vision Transformer (ViT-B/16) dan Swin Transformer Tiny. Seluruh model dilatih menggunakan konfigurasi yang sama agar proses komparasi lebih adil.

3.5.1 Hyperparameter

Hyperparameter yang digunakan pada eksperimen ini adalah sebagai berikut:

- **Jumlah epoch**: 5
- **Batch size**: 32
- **Learning rate**: 1×10^{-4}
- **Optimizer**: AdamW
- **Loss function**: CrossEntropyLoss

Optimizer AdamW dipilih karena lebih stabil dalam proses fine-tuning model pretrained, terutama pada arsitektur Vision Transformer.

3.5.2 Prosedur Pelatihan

Pelatihan model dilakukan menggunakan fungsi `train_one_epoch`. Pada setiap iterasi, dilakukan langkah-langkah berikut:

1. Model diatur ke mode `train()`.
2. Batch citra dan label dipindahkan ke perangkat GPU.
3. Optimizer di-reset menggunakan `optimizer.zero_grad()`.
4. Forward pass dilakukan untuk memperoleh prediksi.
5. Nilai loss dihitung menggunakan `CrossEntropyLoss`.
6. Backpropagation dilakukan melalui `loss.backward()`.
7. Parameter model diperbarui dengan `optimizer.step()`.
8. Akurasi dan akumulasi loss dihitung.

3.5.3 Prosedur Validasi

Evaluasi pada validation set dilakukan melalui fungsi `evaluate`, yang dijalankan dalam kondisi `torch.no_grad()` untuk menghindari komputasi gradien. Proses evaluasi meliputi:

- Model diatur ke mode `eval()`.
- Forward pass pada seluruh batch validation.
- Perhitungan loss, akurasi, prediksi, dan label ground truth.

Contoh pemanggilan fungsi evaluasi:

```
val_loss, val_acc, _, _ = evaluate(
    model, val_loader, criterion)
```

3.5.4 Mekanisme Penyimpanan Model Terbaik

Selama proses training, model terbaik dipilih berdasarkan nilai akurasi pada validation set. Ketika akurasi validasi meningkat, parameter model disimpan:

3.6 Evaluasi Model

Evaluasi dilakukan menggunakan tiga aspek yang dipersyaratkan dalam tugas, yaitu: jumlah parameter, metrik performa, dan waktu inferensi.

1. **Metrik Performa** Pada tahap evaluasi digunakan metrik berikut:
   - Accuracy
   - Precision (macro)
   - Recall (macro)
   - F1-score (macro)

   Evaluasi dilakukan pada validation dan test set menggunakan fungsi `test_model`.
2. **Confusion Matrix** Confusion matrix divisualisasikan menggunakan fungsi `show_confusion`.
3. **Learning Curve** Kurva training/validation loss dan accuracy diplot menggunakan fungsi `plot_loss` dan `plot_acc`.
4. **Inference Time dan Throughput** Waktu inferensi dihitung untuk:
   - rata-rata waktu per gambar
   - total waktu untuk seluruh test set
   - throughput (gambar per detik)

   Pengujian dilakukan dalam mode evaluasi (`model.eval()`) sesuai pedoman tugas.

4 HASIL DAN ANALISIS

Bagian ini membahas hasil eksperimen dari dua arsitektur Vision Transformer, yaitu ViT-B/16 dan Swin Transformer Tiny. Analisis dilakukan berdasarkan metrik performa, visualisasi confusion matrix, kurva pelatihan, serta perbandingan kecepatan inferensi dan jumlah parameter.

4.1 Akurasi Pelatihan dan Validasi

Pelatihan dilakukan selama 5 epoch menggunakan optimizer AdamW dan loss function CrossEntropyLoss. Hasil pelatihan menunjukkan bahwa ViT-B/16 memiliki akurasi validasi yang sedikit lebih tinggi dibandingkan Swin Tiny. Performa pada validation set dapat dilihat pada Tabel 1.

Tabel 1: Akurasi Pelatihan dan Validasi

Model	Akurasi Train	Akurasi Validasi
ViT-B/16	0.9688	0.9358
Swin Tiny	0.9798	0.9506

Hasil pelatihan menunjukkan bahwa Swin Tiny mencapai akurasi training dan validasi yang lebih tinggi dibandingkan ViT-B/16, yaitu 0.9798 dan 0.9506, sementara ViT-B/16 memperoleh 0.9688 dan 0.9358. Perbedaan ini mengindikasikan bahwa arsitektur hierarkis dan window-based self-attention

pada Swin mampu menangkap pola lokal secara lebih efektif pada dataset beresolusi rendah seperti CIFAR-5, sehingga menghasilkan generalisasi yang lebih baik, ditandai dengan selisih train–val yang lebih kecil. Sebaliknya, ViT-B/16 yang mengandalkan global self-attention menunjukkan kecenderungan overfitting ringan akibat kebutuhan data yang lebih besar untuk mencapai stabilitas representasi. Secara keseluruhan, Swin Tiny terbukti lebih efisien dan lebih adaptif untuk skenario data terbatas.

4.2 Evaluasi pada Test Set

Evaluasi akhir dilakukan menggunakan test set CIFAR-5 yang berjumlah 5000 sampel. Akurasi test untuk kedua model ditunjukkan pada Tabel 2.

Tabel 2: Akurasi pada Test Set

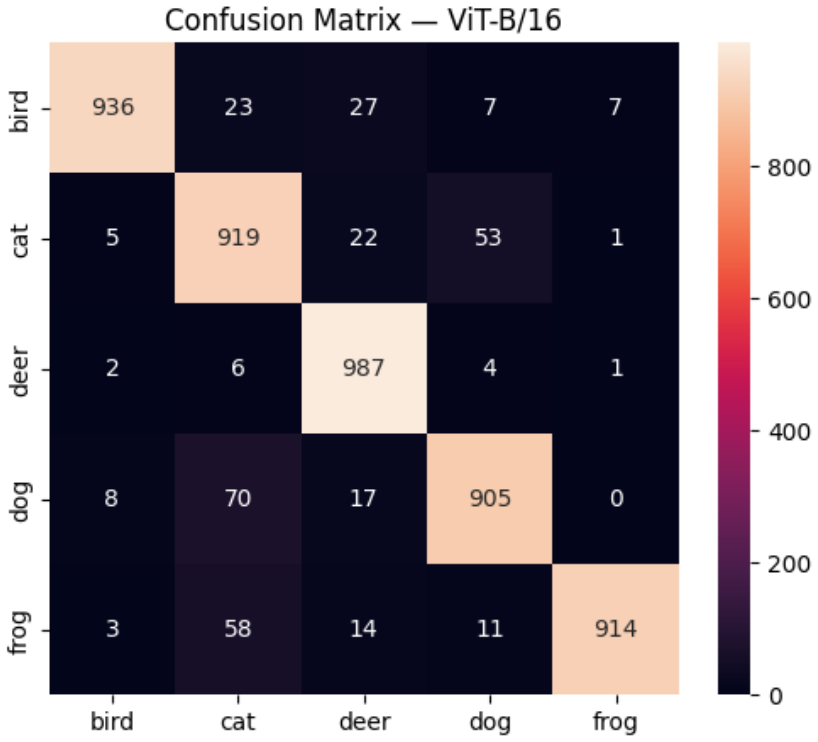
Model	**Akurasi Test**
ViT-B/16	0.9322
Swin Tiny	0.9422

Hasil pengujian pada test set menunjukkan bahwa Swin Tiny kembali unggul dengan akurasi sebesar 0.9422, melampaui ViT-B/16 yang memperoleh 0.9322. Perbedaan performa ini menguatkan temuan sebelumnya bahwa pendekatan hierarkis pada Swin, yang memproses citra melalui windowed self-attention dan patch merging bertahap, lebih efektif dalam mengekstraksi fitur pada citra berukuran kecil seperti CIFAR-5. Sebaliknya, ViT-B/16 yang menerapkan global attention secara seragam cenderung kurang optimal pada dataset kecil dan dapat mengalami sensitivitas lebih tinggi terhadap variasi minor pada data uji. Dengan demikian, hasil test set mengonfirmasi bahwa Swin Tiny memiliki generalisasi yang lebih baik dan lebih stabil dibandingkan ViT-B/16 pada skenario klasifikasi dengan data terbatas.

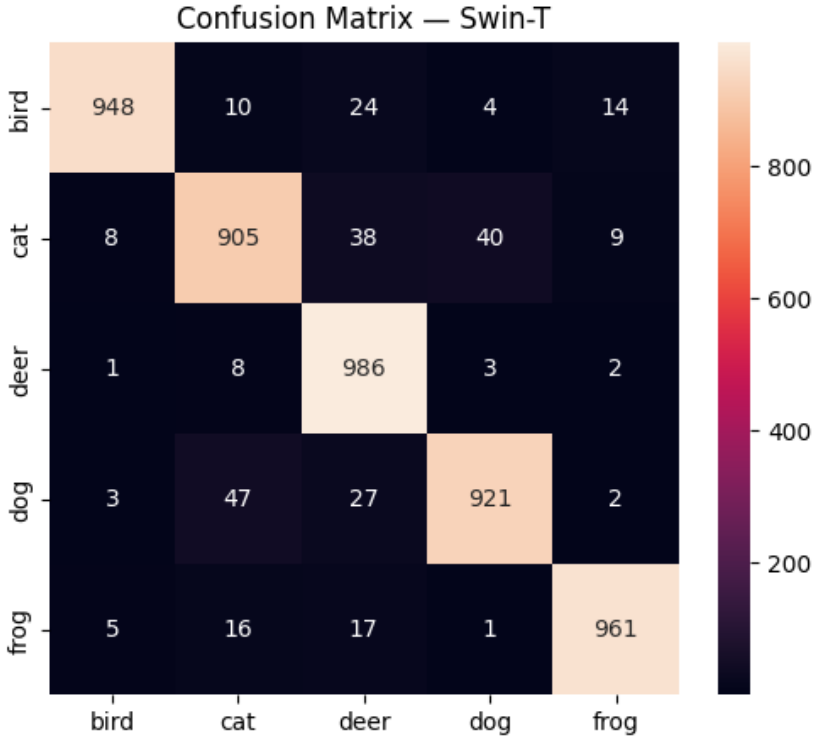
4.3 Visualisasi Confusion Matrix

Confusion matrix digunakan untuk mengevaluasi performa model secara lebih rinci terhadap setiap kelas dengan membandingkan label sebenarnya pada sumbu vertikal dengan label prediksi pada sumbu horizontal. Analisis confusion matrix pada Gambar 1 dan Gambar 2 menunjukkan bahwa kedua model cenderung melakukan kesalahan pada kelas dengan kemiripan visual. Pada model ViT-B/16, misclass terbesar terjadi pada pasangan *cat* dan *dog*, di mana banyak sampel kedua kelas tersebut saling tertukar akibat kemiripan tekstur dan bentuk objek. Sebagian kecil sampel *cat* juga salah diprediksi sebagai *frog*, yang mengindikasikan sensitivitas ViT terhadap variasi latar dan pose.

Sementara itu, Swin Tiny menunjukkan pola kesalahan yang lebih tersebar, terutama pada kelas *cat*, *deer*, dan *dog*. Hal ini terkait dengan mekanisme window-based attention yang lebih fokus pada fitur lokal, sehingga informasi global kurang kuat untuk membedakan kelas yang bentuk siluetnya serupa. Secara keseluruhan, ViT-B/16 paling kesulitan pada pasangan *cat–dog*, sedangkan Swin Tiny menunjukkan tantangan yang lebih luas pada tiga kelas berkarakteristik mirip tersebut.



Gambar 1: Confusion Matrix ViT-B/16

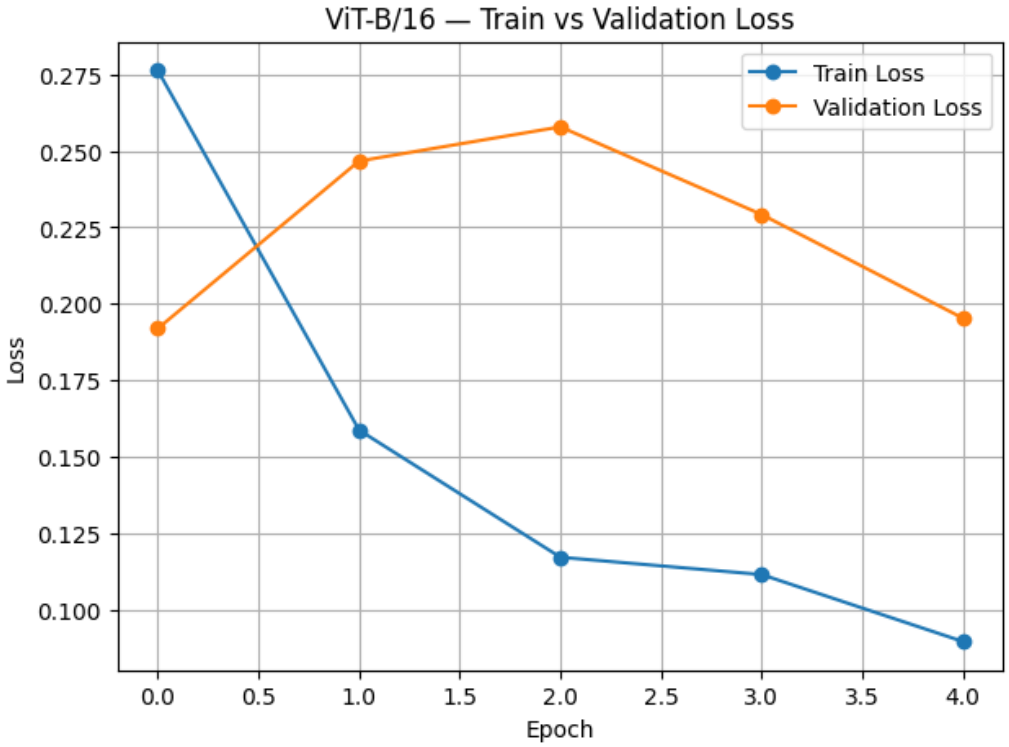


Gambar 2: Confusion Matrix Swin Tiny

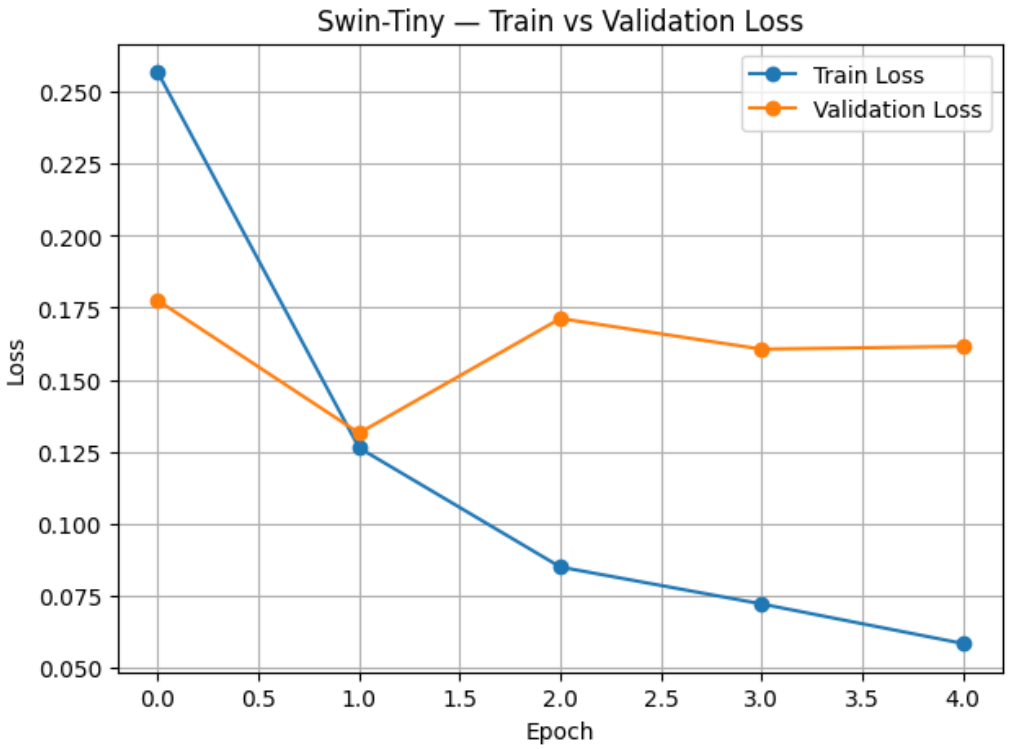
4.4 Visualisasi Kurva Pelatihan

Hasil kurva pelatihan pada Gambar 3 dan Gambar 4 menunjukkan dinamika penurunan loss selama proses pelatihan untuk kedua model. Pada model ViT-B/16, *training loss* menurun secara stabil, tetapi *validation loss* sempat meningkat pada epoch awal sebelum kembali menurun. Pola ini mengindikasikan fase adaptasi awal yang umum terjadi pada arsitektur Vision Transformer, yang memerlukan lebih banyak iterasi untuk menstabilkan representasi global, terutama pada dataset berukuran kecil seperti CIFAR-5.

Sebaliknya, Swin Tiny menunjukkan penurunan *training loss* yang lebih cepat dan agresif, selaras dengan arsitektur hierarchical window-based attention yang lebih efektif dalam menangkap fitur lokal sejak awal pelatihan. *Validation loss* pada Swin relatif stabil, meskipun terdapat sedikit kenaikan pada epoch pertengahan yang menunjukkan potensi overfitting ringan. Secara keseluruhan, kurva pelatihan memperlihatkan bahwa Swin Tiny lebih efisien dalam mempelajari data berukuran kecil, sementara ViT-B/16 mencapai stabilitas generalisasi pada epoch akhir.



Gambar 3: Kurva Loss ViT-B/16



Gambar 4: Kurva Loss Swin Tiny

4.5 Hasil Inferensi

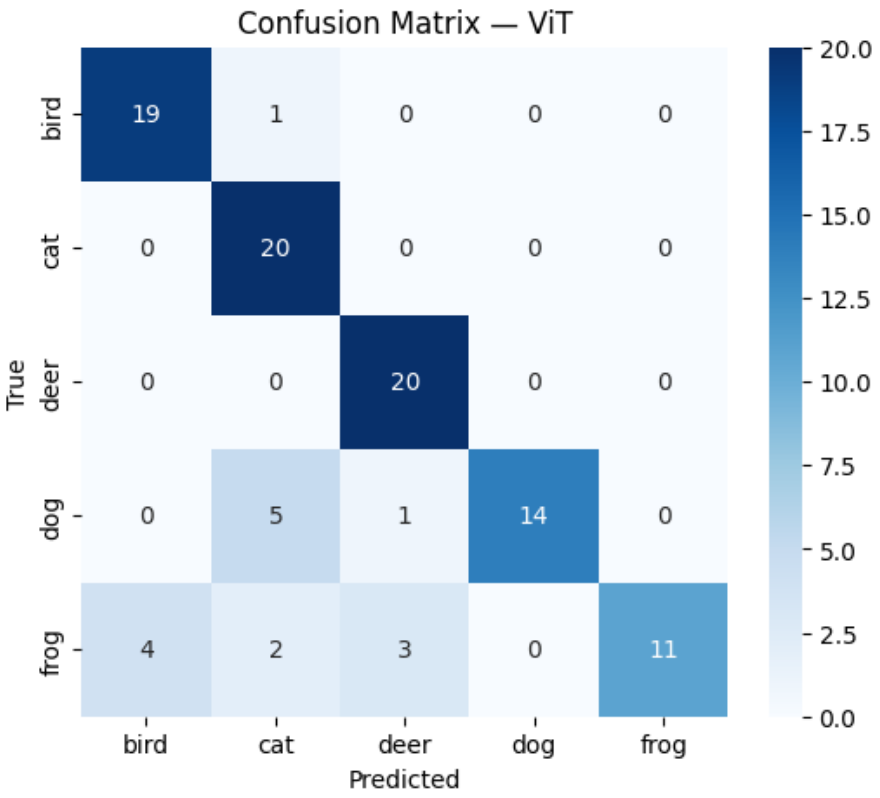
Dataset untuk proses inferensi ini diambil dari luar dataset CIFAR-10 untuk mengukur akurasi diluar lingkungan training. Dengan masing masing kelas berisi 20 gambar, sehingga total data inferensi sebanyak 100 gambar. Proses inferensi dilakukan pada himpunan data uji terpisah yang disimpan dalam struktur folder berdasarkan kelas. Model menghasilkan prediksi menggunakan mode evaluasi dengan menonaktifkan gradien, dan hasilnya kemudian digunakan untuk menyusun confusion matrix serta classification report. Selain itu, dilakukan pula *random inference* untuk mengamati prediksi model secara visual pada beberapa contoh acak.

Berdasarkan hasil inferensi, model ViT-B/16 memperoleh akurasi sebesar 0.84. Model menunjukkan performa baik pada kelas *cat* dan *deer* dengan nilai *recall* mencapai 1.00, namun mengalami penurunan performa pada kelas *frog* (recall 0.55). Kesalahan terbesar terjadi pada pasangan kelas yang memiliki kemiripan visual seperti *cat–dog* serta misclass pada kelas *frog*.

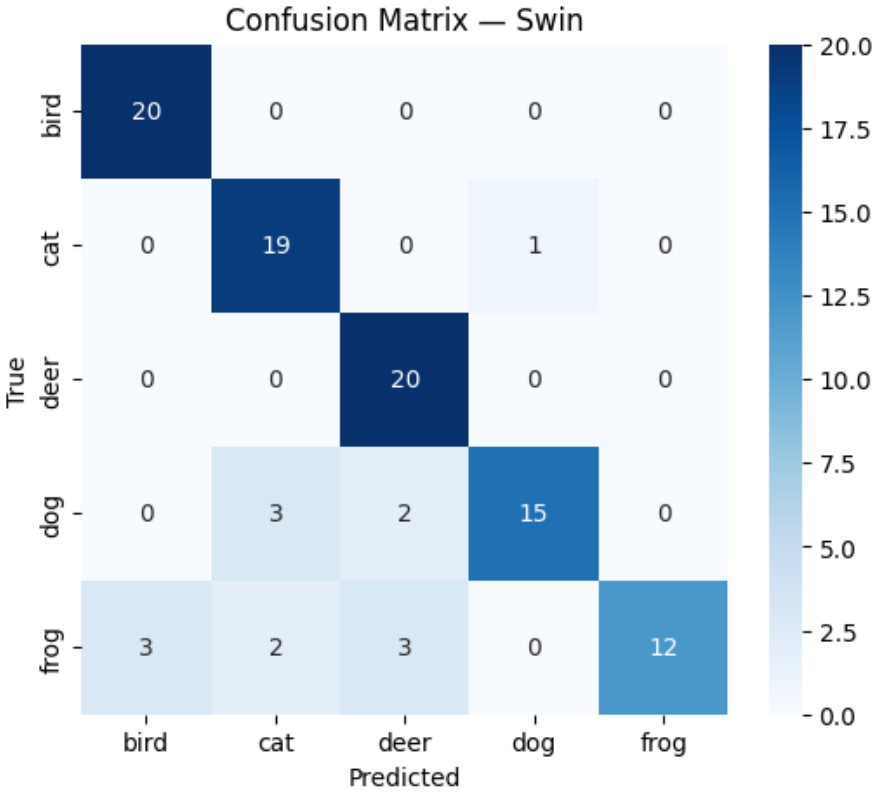
Swin Tiny menunjukkan performa yang lebih baik dengan akurasi 0.86. Model memiliki *recall* tinggi pada kelas *bird*, *cat*, dan *deer* (0.95–1.00). Namun, performa pada kelas *frog* tetap menjadi titik lemah (recall 0.60). Secara keseluruhan, arsitektur hierarkis Swin memberikan generalisasi lebih stabil pada data inferensi dibandingkan ViT-B/16.

Tabel 3: Metrik Inferensi Model ViT-B/16 dan Swin Tiny

Model	Precision	Recall	F1-Score	Accuracy
ViT-B/16	0.87	0.84	0.83	0.84
Swin Tiny	0.88	0.86	0.85	0.86



Gambar 5: Confusion Matrix Hasil Inferensi — ViT-B/16



Gambar 6: Confusion Matrix Hasil Inferensi — Swin Tiny

4.5.1 Perbandingan Waktu Inferensi

Pengukuran waktu inferensi dilakukan untuk mengevaluasi efisiensi kedua model dalam melakukan prediksi pada citra tunggal. Proses ini dilakukan dengan mengeksekusi 50 citra secara berulang menggunakan fungsi `measure_inference_time`, yang menghitung *latency* rata-rata per citra, waktu total eksekusi, serta *throughput* dalam satuan gambar per detik.

Hasil pengukuran menunjukkan bahwa ViT-B/16 memiliki *latency* rata-rata sebesar 24,955 ms per gambar dengan *throughput* 40,073 gambar/detik. Sementara itu, Swin Tiny memiliki *latency* sedikit lebih tinggi, yaitu 25,704 ms per gambar dan *throughput* 38,904 gambar/detik.

Perbedaan ini mengindikasikan bahwa meskipun Swin Tiny memiliki arsitektur lebih ringan dan jumlah parameter lebih kecil, ViT-B/16 justru memberikan waktu inferensi sedikit lebih cepat pada konfigurasi perangkat yang digunakan. Hal ini dapat disebabkan oleh optimisasi kernel dan efisiensi implementasi operator global attention pada framework yang digunakan. Namun, perbedaannya relatif kecil dan keduanya tetap berada pada kategori model dengan kecepatan inferensi yang tinggi.

Tabel 4: Perbandingan Waktu Inferensi ViT-B/16 dan Swin Tiny

Model	**Latency (ms/img)**	**Total Time (ms)**	**Throughput (img/s)**
ViT-B/16	24.955	1247.725	40.073
Swin Tiny	25.704	1285.215	38.904

5 Analisis Akhir

Berdasarkan keseluruhan hasil evaluasi yang mencakup performa pelatihan, validasi, pengujian, serta inferensi, kedua model menunjukkan karakteristik dan keunggulan yang berbeda. Swin Tiny secara konsisten menghasilkan akurasi validasi, akurasi *test set*, dan akurasi inferensi yang lebih tinggi dibandingkan ViT-B/16. Arsitektur *hierarchical window-based self-attention* pada Swin memungkinkan model untuk mengekstraksi fitur lokal secara efisien, sehingga lebih sesuai untuk dataset berukuran kecil seperti CIFAR-5 yang memiliki resolusi rendah dan pola visual yang lebih lokal. Hal ini tercermin dari stabilitas *validation loss* dan nilai metrik inferensi yang lebih baik.

Sebaliknya, ViT-B/16 yang bergantung pada mekanisme *global self-attention* menunjukkan performa kompetitif, namun kurang stabil pada dataset kecil. Model ini mengalami fluktuasi *validation loss* pada epoch awal dan memiliki akurasi inferensi yang sedikit lebih rendah. Meskipun demikian, ViT-B/16 menunjukkan keunggulan kecil pada waktu inferensi, dengan *latency* yang sedikit lebih cepat. Keunggulan ini dapat dikaitkan dengan optimisasi implementasi global attention pada perangkat komputasi, meskipun model memiliki jumlah parameter yang lebih besar.

Dari aspek *trade-off*, Swin Tiny menawarkan kombinasi paling seimbang: akurasi lebih tinggi, jumlah parameter lebih kecil, dan performa inferensi yang efisien, sehingga memberikan generalisasi yang lebih stabil. Sementara itu, ViT-B/16 tetap memiliki potensi lebih besar pada dataset berskala besar, di mana representasi global dapat dimanfaatkan secara optimal.

Secara keseluruhan, Swin Tiny merupakan model yang lebih sesuai untuk dataset kecil seperti CIFAR-5, menawarkan keseimbangan terbaik antara akurasi, efisiensi parameter, dan kecepatan inferensi. ViT-B/16 tetap merupakan arsitektur kuat, namun dalam konteks eksperimen ini, Swin menunjukkan performa yang lebih konsisten dan efisien.

6 KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil eksperimen, perbandingan performa, dan analisis mendalam terhadap model ViT-B/16 dan Swin Tiny, beberapa kesimpulan utama dapat diperoleh sebagai berikut:

- Swin Tiny menunjukkan performa yang secara konsisten lebih unggul pada *validation set*, *test set*, dan data inferensi, dengan akurasi keseluruhan lebih tinggi dibandingkan ViT-B/16.
- Arsitektur *hierarchical window-based attention* pada Swin Tiny lebih efisien dalam mempelajari pola lokal, sehingga lebih sesuai untuk dataset kecil seperti CIFAR-5.
- ViT-B/16 tetap memberikan performa yang kompetitif, namun menunjukkan instabilitas pada epoch awal pelatihan dan akurasi inferensi yang lebih rendah.
- Dari sisi kecepatan inferensi, ViT-B/16 memiliki *latency* sedikit lebih cepat, meskipun perbedaannya tidak signifikan.
- Secara umum, Swin Tiny menawarkan trade-off terbaik antara akurasi, jumlah parameter, dan efisiensi penggunaan komputasi.

6.2 Rekomendasi Model Berdasarkan Use Case

- **Untuk akurasi maksimal:** Swin Tiny merupakan pilihan terbaik, karena memberikan akurasi validasi, akurasi pengujian, dan kinerja inferensi yang lebih tinggi dibandingkan ViT-B/16.
- **Untuk efisiensi komputasi:** Swin Tiny lebih disarankan karena memiliki arsitektur yang lebih ringan dan jumlah parameter yang lebih kecil, dengan performa akurasi yang tetap tinggi.
- **Untuk aplikasi real-time:** ViT-B/16 dapat dipertimbangkan pada perangkat dengan optimisasi global attention, karena memiliki latensi inferensi sedikit lebih cepat. Namun, perbedaannya sangat kecil dan Swin Tiny tetap kompetitif untuk aplikasi real-time.

6.3 Saran

- Penelitian selanjutnya dapat menggunakan dataset dengan resolusi lebih tinggi atau ukuran lebih besar untuk mengevaluasi performa ViT secara lebih optimal.
- Eksperimen dapat diperluas dengan menambahkan teknik *data augmentation*, *regularization*, atau *fine-tuning* lanjutan untuk meningkatkan generalisasi model.
- Pengujian pada perangkat dengan kemampuan komputasi berbeda (CPU, GPU low-end, edge device) dapat memberikan gambaran lebih jelas mengenai efisiensi model di berbagai lingkungan deployment.

References

[1] 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 16×16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[2] M. Raghu, T. Unterthiner, C. Olah, and S. Kornblith, "Do vision transformers see like convolutional neural networks?" in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. [Online]. Available: https://arxiv.org/abs/2108.08810

[3] Z. Liu, Y. Lin, Y. Cao, H. Hu, Z. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *International Conference on Computer Vision (ICCV)*, 2021. [Online]. Available: https://arxiv.org/abs/2103.14030

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[5] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2661–2671. [Online]. Available: https://arxiv.org/abs/1805.08974

Lampiran

A Source Code Tugas

Source code lengkap tersedia pada repositori GitHub berikut:

- **Source code tugas:** https://github.com/anzensirc/VisionTransformer-Comparison
- **Model Vision Transformer (Official):** https://github.com/google-research/vision_transformer
- **Model Swin Transformer (Official):** https://github.com/microsoft/Swin-Transformer
- **Pretrained models (timm):** https://github.com/huggingface/pytorch-image-models

Seluruh kode yang digunakan dalam eksperimen telah diatur dan terdokumentasi dalam repositori tersebut.

B Output Training Log

Berikut adalah contoh template log hasil pelatihan. Bagian ini dapat diganti dengan log asli dari pelatihan model:

C Output Training Log

```
=== Training vit_base_patch16_224 ===

(model.safetensors downloaded successfully)

Epoch 1/5
Train Acc: 0.9046 | Val Acc: 0.9344
```

```
Epoch 2/5
Train Acc: 0.9462 | Val Acc: 0.9128

Epoch 3/5
Train Acc: 0.9595 | Val Acc: 0.9194

Epoch 4/5
Train Acc: 0.9609 | Val Acc: 0.9210

Epoch 5/5
Train Acc: 0.9688 | Val Acc: 0.9358


=== Training swin_tiny_patch4_window7_224 ===

(model.safetensors downloaded successfully)

Epoch 1/5
Train Acc: 0.9111 | Val Acc: 0.9386

Epoch 2/5
Train Acc: 0.9575 | Val Acc: 0.9582

Epoch 3/5
Train Acc: 0.9710 | Val Acc: 0.9442

Epoch 4/5
Train Acc: 0.9753 | Val Acc: 0.9488

Epoch 5/5
Train Acc: 0.9798 | Val Acc: 0.9506
```

D Screenshot Hasil Eksperimen

Contoh Hasil Inferensi (Random Prediction)

Gambar 7: Contoh hasil inferensi 5 sampel acak dengan model ViT-B/16.

Gambar 8: Contoh hasil inferensi 5 sampel acak dengan model Swin Tiny.

E Sumber Dataset Inferensi

Dataset yang digunakan untuk pengujian inferensi tambahan berasal dari sumber publik berikut:

- **Frog Dataset:** https://images.cv/dataset/frog-image-classification-dataset
- **Deer Dataset:** https://www.kaggle.com/datasets/tharunk07/deer-classification-dataset
- **Bird Dataset:** https://www.kaggle.com/datasets/rahmasleam/bird-speciees-dataset
- **Dog & Cat (Animals-10):** https://www.kaggle.com/datasets/alessiocorrado99/animals10

Dataset digunakan hanya pada proses inferensi tambahan untuk menguji ketahanan model terhadap data di luar set pelatihan.