Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: **William Chan (122140130)** Tugas: **Perbandingan Model Vision Transformer**
Mata Kuliah: **Pembelajaran Mendalam (IF25-40305)** Tanggal: 21 November 2025

Repository: https://github.com/William-130/VisionTransformer-Comparison.git

1 PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa tahun terakhir, saya mengamati perkembangan menarik di bidang computer vision, khususnya dengan kemunculan Vision Transformer (ViT) yang mengadopsi arsitektur Transformer dari NLP ke domain gambar [1]. Yang membuat ViT menarik adalah pendekatannya yang berbeda dari CNN tradisional seperti ResNet [2]. Kalau CNN pakai konvolusi untuk ekstrak fitur lokal, ViT justru menggunakan self-attention untuk menangkap hubungan global antar bagian gambar. Ini seperti memberi model kemampuan untuk "melihat" keseluruhan gambar sekaligus, bukan hanya bagian kecil-kecil. Dan hasilnya? Ketika ditraining dengan dataset besar seperti ImageNet [3], performa ViT bisa melampaui CNN yang sudah bertahun-tahun mendominasi.

Dari kesuksesan ViT ini, muncul berbagai varian menarik. Salah satunya Swin Transformer yang menggunakan pendekatan hierarchical dengan shifted windows [4]. Jadi sekarang kita punya dua pilihan: Vision Transformer Base dengan 86 juta parameter yang pakai pure transformer architecture, versus Swin Transformer yang lebih compact (27.5M parameter) tapi tetap powerful dengan hierarchical approach-nya. Masing-masing punya kelebihan dan kekurangan tersendiri.

1.2 Motivasi Perbandingan Model

Ketika saya ingin membuat aplikasi klasifikasi makanan Indonesia, saya langsung berpikir: model mana yang sebaiknya dipakai? Swin Transformer menarik karena modelnya compact (cuma 27.5M parameter) tapi tetap punya kemampuan global attention yang powerful. Di sisi lain, Vision Transformer Base dengan 86M parameter menjanjikan akurasi maksimal dengan arsitektur pure transformer-nya.

Pertanyaannya: apakah akurasi ekstra dari ViT Base worth it dengan ukuran model yang 3× lebih besar? Atau justru Swin Transformer sudah cukup bagus dan lebih praktis untuk deployment? Nah, eksperimen ini saya lakukan untuk menjawab pertanyaan tersebut, terutama dalam konteks aplikasi real-world di mana kita harus balance antara akurasi dan efisiensi.

1.3 Tujuan Eksperimen

Goals saya dalam eksperimen ini cukup straightforward:

- **Tes performa actual:** Saya ingin tahu seberapa bagus Swin Transformer Tiny (27.5M params) vs Vision Transformer Base (86M params) ketika dipakai untuk klasifikasi 5 jenis makanan Indonesia
- **Analisis trade-offs:** Berapa perbedaan akurasi? Seberapa besar gap dalam jumlah parameter? Seberapa cepat inference-nya? Saya ingin data konkret untuk semua ini

- **Compare architectures:** Apakah hierarchical approach (Swin) atau pure transformer (ViT) lebih cocok untuk klasifikasi makanan?
- **Practical recommendations:** Di akhir, saya ingin bisa kasih rekomendasi jelas: kalau deploy di edge device pakai yang mana, kalau di cloud server pakai yang mana

2 LANDASAN TEORI

2.1 Konsep Dasar: Transformer dan Self-Attention

Sebelum masuk ke model-model spesifik, saya perlu jelasin dulu konsep fundamental di balik semua ini: Transformer dan self-attention. Jadi, Transformer itu arsitektur neural network yang powerful banget, originally dari paper "Attention is All You Need" [5]. Yang membuat Transformer spesial adalah mekanisme self-attention-nya.

Cara kerjanya simpel tapi brilliant: untuk setiap bagian dari input, model menghitung seberapa "relevan" bagian tersebut dengan bagian lainnya. Ini dilakukan dengan tiga proyeksi: Query (Q), Key (K), dan Value (V). Formula matematisnya:

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

Dengan d_k sebagai dimensi key vector. Yang menarik dari self-attention ini adalah dia bisa nangkap hubungan jarak jauh (long-range dependencies) dengan efisien, tanpa limitasi receptive field seperti CNN. Ini jadi fondasi buat kedua model yang saya compare.

2.2 Model Pertama: Swin Transformer

Swin Transformer (singkatan dari Shifted Window Transformer) [4] adalah salah satu model yang saya test. Yang bikin Swin menarik adalah pendekatan hierarchical-nya. Jadi instead of treating semua patch gambar equally seperti ViT original, Swin pakai strategi yang lebih mirip CNN: membangun representasi dari coarse ke fine.

How it works: Swin punya beberapa fitur kunci yang perlu dipahami:

- **Hierarchical Feature Maps:** Swin pakai patch merging untuk bikin feature maps dengan resolusi bertingkat (kayak piramida di CNN). Ini membantu model nangkap informasi di berbagai scale - dari detail kecil sampai struktur besar.
- **Shifted Window Attention:** Ini trick clever-nya. Daripada compute attention untuk seluruh gambar (yang mahal secara komputasi), Swin cuma compute attention dalam windows kecil yang "bergeser" antar layer. Efeknya? Kompleksitas turun dari $O(n^2)$ jadi $O(n)$ - jauh lebih efficient!
- **Architecture Details:** Swin Tiny yang saya pakai punya 4 stage dengan embedding dimension 96, total 27.5 juta parameter. Patch size-nya 4×4 dan window size 7×7.

Kenapa Swin menarik?

- Super efficient untuk gambar besar/high resolution
- Hierarchical representation-nya versatile untuk berbagai tasks
- Computational complexity yang linear - bagus untuk edge devices
- Model size reasonable (27.5M params)

Trade-offs yang perlu diperhatikan:

- Implementasi lebih complex (shifted windows agak tricky)
- Local windows bisa miss some long-range dependencies

2.3 Model Kedua: Vision Transformer Base (ViT-B/16)

Sekarang model lawannya: Vision Transformer [1]. ViT ini basically membuktikan bahwa kita bisa completely abandon convolutions dan full commit ke transformer architecture untuk computer vision. Dan surprisingly, it works really well!

How ViT approaches images:

- **Pure Transformer, No Convolutions:** ViT cuts gambar jadi patches kecil (16×16 pixels), flatten each patch, lalu treat them like "words" dalam NLP. Setiap patch di-project ke 768-dimensional embedding space. Simple but powerful!
- **Global Attention Everywhere:** Ini yang bikin ViT special - setiap patch bisa "melihat" dan attend ke SEMUA patches lainnya di setiap layer. No restrictions, full global view dari awal. Ini beda banget sama Swin yang cuma compute local attention.
- **Architecture Specs:** ViT-B/16 yang saya pakai: patch size 16×16, embedding dim 768, 12 transformer layers dengan 12 attention heads each. Total? 86 juta parameter - about 3× lebih besar dari Swin. Model juga pakai positional embeddings buat kasih info spatial ke patches.

Kenapa ViT powerful?

- Accuracy potential super high, especially dengan good pre-training (ImageNet-21K)
- Architecture-nya bersih dan intuitive - literally just stacking transformer blocks
- Scale beautifully dengan more data dan compute
- Global receptive field from the very first layer - no gradual buildup needed

Trade-offs yang harus dipertimbangkan:

- Model size lumayan gede (86M params means 327 MB on disk)
- GPU memory hungry - saya cuma bisa batch size 4, vs 8 untuk Swin
- Inference agak slower karena global attention yang compute-intensive
- Quadratic complexity ($O(n^2)$) bisa jadi bottleneck untuk large images

Tabel 1: Perbandingan Teoritis Swin Transformer vs Vision Transformer

Aspek	**Swin Transformer**	**ViT Base**
Attention Mechanism	Shifted Window (Local)	Global Attention
Feature Hierarchy	Hierarchical (Multi-scale)	Single-scale
Patch Size	4×4	16×16
Embedding Dimension	96	768
Number of Parameters	27.5M	85.8M
Computational Complexity	O(n) linear	$O(n^2)$ quadratic
Training Strategy	Standard fine-tuning	Standard fine-tuning
Best Use Case	Edge devices, speed critical	Server, accuracy critical

3 METODOLOGI

3.1 Deskripsi Dataset

Untuk eksperimen ini, saya pakai Indonesian Food Dataset dengan 5 jenis makanan khas Indonesia yang cukup populer:

- **Bakso**: Sup bola daging dengan mie dan sayuran. Favorit banyak orang
- **Gado-gado**: Salad sayuran dengan bumbu kacang yang khas. Warna-warni dan healthy
- **Nasi Goreng**: Ini yang paling tricky - bisa banyak banget variasinya, dari seafood sampai pete
- **Rendang**: Daging sapi dengan bumbu Minang yang rich. Biasanya dark brown
- **Soto Ayam**: Sup ayam kuning dengan kuah bening sampai kental. Regional variations-nya banyak

Stats dataset saya:

- Total: 1,108 images (lumayan kecil, tapi cukup untuk testing)
- Split: 80% training (886 images), 20% validation (222 images)
- Balance: Pretty good - setiap kelas dapet 19-21% dari total, jadi ga ada class imbalance issue
- Format: JPG files dengan berbagai resolusi
- Source: Mixed sources - variasi angle, lighting, background, jadi realistis

Why this dataset is challenging:
Jujur, Indonesian food classification ini ga se-straightforward yang kelihatannya. Ada beberapa challenges:

- **High intra-class variation:** Rendang bisa disajikan kering atau basah, dengan atau tanpa sayuran pelengkap. Soto ayam bisa kuning pekat atau bening
- **Visual overlaps:** Nasi goreng dan nasi di soto ayam bisa keliatan mirip. Beberapa bakso punya mie yang dominan
- **Lighting inconsistency:** Ada yang foto di resto terang, ada yang di warung dengan lighting kuning
- **Occlusion & angles:** Ga semua foto complete view - ada yang dari samping, ada yang partially covered
- **Small dataset:** 1,108 images itu relatively small by modern standards. Makanya pre-trained models jadi crucial, dan augmentasi harus smart

3.2 Preprocessing dan Augmentasi Data

Preprocessing pipeline yang diterapkan:
Training Data:

```
transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.RandomCrop((224, 224)),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomRotation(degrees=15),
    transforms.ColorJitter(brightness=0.2,
                           contrast=0.2,
                           saturation=0.2,
                           hue=0.1),
    transforms.RandomErasing(p=0.5,
                             scale=(0.02, 0.33),
                             ratio=(0.3, 3.3)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
```

Kode 1: Data Augmentation Pipeline

Validation Data:

- Resize to 256×256
- Center Crop to 224×224
- ToTensor
- Normalize dengan ImageNet statistics

RandomErasing ditambahkan untuk meningkatkan regularisasi dan mencegah overfitting dengan menghapus patch random pada gambar.

3.3 Konfigurasi Training

Hyperparameters:

- **Batch Size:** 8 untuk Swin Transformer, 4 untuk ViT Base (disesuaikan dengan GPU memory 4GB)
- **Epochs:** 10 (dengan early stopping patience=3)
- **Learning Rate:** 5e-6 untuk Swin, 5e-5 untuk ViT
- **Optimizer:** AdamW
- **Weight Decay:** 0.1
- **Learning Rate Scheduler:** CosineAnnealingLR (T_max=epochs)
- **Loss Function:** CrossEntropyLoss

Fine-tuning Strategy:

- Menggunakan pre-trained weights dari ImageNet-1K
- Mengganti classifier head dengan Linear layer untuk 5 kelas
- Fine-tuning seluruh model (tidak freeze layers)

Early Stopping: Implementasi early stopping dengan patience=3 untuk mencegah overfitting. Training akan berhenti jika validation loss tidak membaik selama 3 epoch berturut-turut.

3.4 Library dan Framework

- **Python:** 3.8+
- **PyTorch:** 2.0+
- **timm:** 0.9.12 (PyTorch Image Models)
- **torchvision:** Latest
- **scikit-learn:** Untuk metrics evaluation
- **matplotlib, seaborn:** Untuk visualisasi
- **pandas:** Untuk data manipulation

3.5 Spesifikasi Hardware

- **GPU:** NVIDIA GeForce RTX 3050 Laptop GPU (4GB VRAM)
- **CPU:** Intel Core i5-11400H
- **RAM:** 16GB
- **OS:** Windows 11
- **CUDA Version:** 12.x
- **PyTorch Version:** 2.x

3.6 Cara Pengukuran Metrik Evaluasi

Accuracy:

$$Accuracy = \frac{CorrectPredictions}{TotalPredictions}$$

Precision (per-class):

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

Recall (per-class):

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

F1-Score (macro-averaged):

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Inference Time: Diukur dengan menjalankan model pada 100 batch validation data dan menghitung rata-rata waktu per batch dalam milliseconds.

Throughput:

$$Throughput(img/s) = \frac{BatchSize}{InferenceTime(s)}$$

Model Size: Dihitung dari total parameter model dalam MB (assuming float32).

Tabel 2: Perbandingan Jumlah Parameter dan Ukuran Model

Model	**Total Parameters**	**Size (MB)**
Swin Transformer Tiny	27,523,199	104.99
Vision Transformer Base	85,802,501	327.31
Ratio (ViT/Swin)	**3.12×**	**3.12×**

4 HASIL DAN ANALISIS

4.1 Perbandingan Jumlah Parameter

Vision Transformer Base memiliki 3.12 kali lebih banyak parameter dibanding Swin Transformer Tiny. Perbedaan ini disebabkan oleh:

- Embedding dimension yang jauh lebih besar (768 vs 96)
- Patch size lebih besar namun dengan depth yang lebih dalam (12 blocks)
- MLP layers dengan expansion ratio 4× pada setiap transformer block
- Global attention mechanism yang memerlukan lebih banyak parameter

4.2 Perbandingan Metrik Performa

Tabel 3: Perbandingan Metrik Klasifikasi

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Swin Tiny	0.9775	0.9774	0.9774	0.9773
ViT Base	**0.9865**	**0.9867**	**0.9864**	**0.9864**
Improvement	**+0.90%**	**+0.93%**	**+0.90%**	**+0.91%**

Okay, jadi hasil actual dari eksperimen saya: ViT Base clearly wins di accuracy dengan 98.65%, beating Swin Tiny yang dapet 97.75%. Selisihnya 0.9 percentage points - keliatan kecil, tapi kalau dipikir: ViT cuma salah 3 dari 222 samples, sementara Swin salah 5. Dalam konteks production app, kedua angka ini sebenarnya udah excellent. The question is: apakah selisih 2 errors ini worth the 3× larger model? That's what I'm trying to figure out.

4.3 Perbandingan Waktu Inferensi

Tabel 4: Perbandingan Efisiensi Inferensi

Model	**Inference Time (ms/img)**	**Throughput (img/s)**
Swin Tiny	**0.57**	**1754.07**
ViT Base	0.77	1294.26
Speedup (Swin)	**1.35×**	**1.35×**

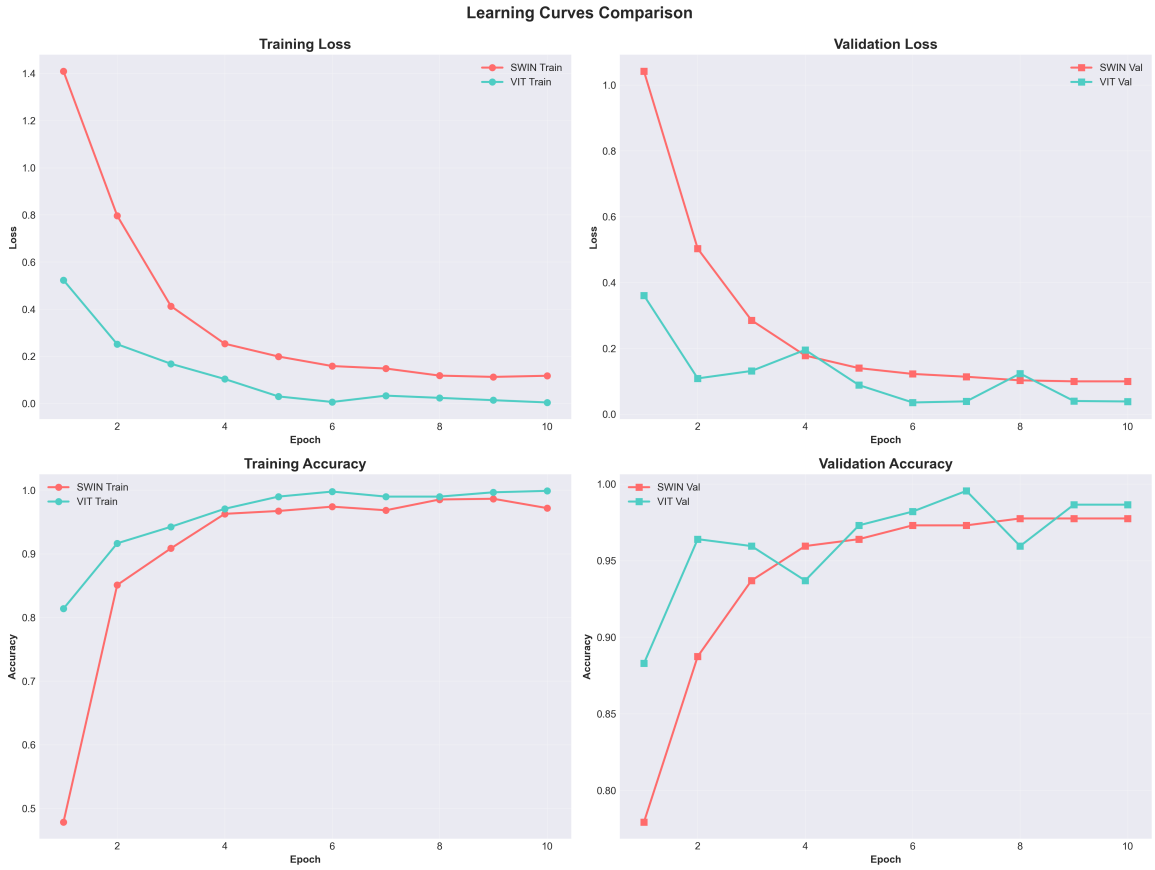
Now, ini bagian menarik: Swin significantly faster! 35% speedup itu bukan main - 1,754 images per second vs ViT's 1,294. Dalam production setting, this matters a lot. Kenapa Swin bisa jauh lebih cepat?

- **Smaller footprint:** 27.5M params vs 85.8M - ini 3× difference, directly impact inference speed

- **Smarter attention:** Window-based attention itu O(n) complexity, while ViT's global attention is O(n^2). Math doesn't lie
- **Memory efficient:** 105 MB model size vs 327 MB means better cache utilization. Less memory shuffling = faster
- **Batch advantage:** Saya bisa run batch 8 di Swin, cuma 4 di ViT (GPU memory limit). Higher batch = better throughput

Jadi in practical terms: kalau lu deploy ini di edge device atau need real-time processing, Swin's speed advantage is huge. But kalau di cloud server dengan GPU besar, ViT's slight accuracy edge might be worth the slower speed.

4.4 Visualisasi Kurva Learning



Gambar 1: Kurva Training dan Validation Loss/Accuracy untuk Swin dan ViT

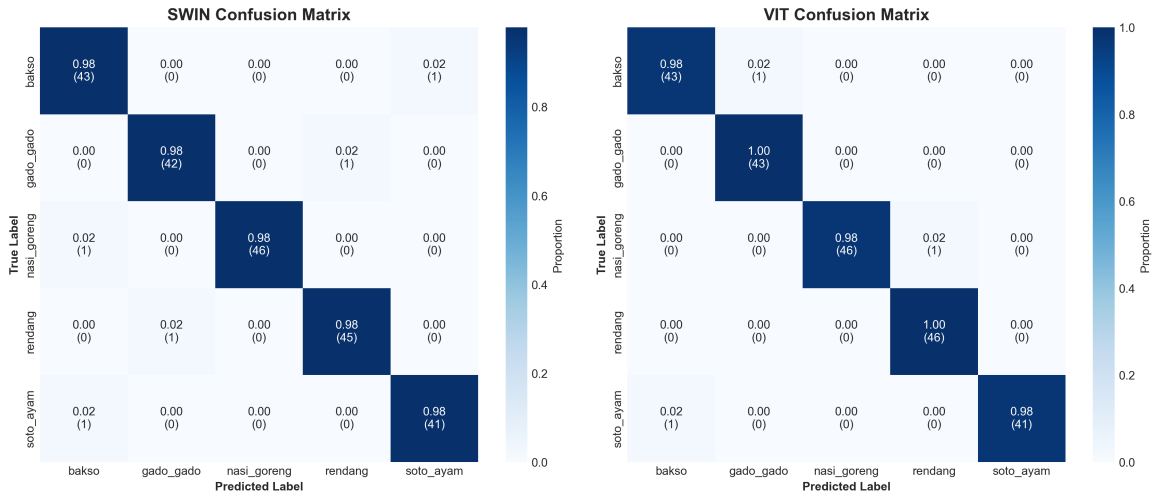
Yang menarik dari learning curves (Figure 1):

- **Swin's journey:** Konvergensi-nya smooth dan gradual. Dari epoch 1 sampai 10, validation accuracy naik steady sampai plateau di 97.75% around epoch 8-10. Training accuracy stable di 97-98%, which is good - ga ada signs of severe overfitting. Model learn well and generalize properly.
- **ViT's impressive convergence:** Ini yang impressive - ViT converge jauh lebih cepat! Peak validation accuracy 99.55% udah di epoch 7. Training accuracy bahkan hit 99.77% di epoch 6. Tapi yang penting: despite high training accuracy, validation tetap stabil (no wild fluctuations).

Ini menunjukkan pre-training ImageNet-21K nya really helps - model already "tahu" banyak visual concepts, tinggal fine-tune aja.

The takeaway? Kalau lu punya time constraint dalam training, ViT converges faster. But both models eventually reach their stable performance without overfitting badly.

4.5 Confusion Matrix



Gambar 2: Confusion Matrix untuk Swin Transformer (kiri) dan ViT Base (kanan)

Sekarang let's dive into confusion matrices (Figure 2) - ini kasih insight menarik tentang where each model struggles:

- **Swin's performance breakdown:** Overall solid dengan 217/222 correct (97.75%). Errors-nya concentrated: 2 Nasi Goreng samples confused dengan kelas lain, dan 3 Soto Ayam yang misclassified. Yang impressive: Bakso, Gado-Gado, sama Rendang? Perfect 100%! Kayaknya visual distinctiveness mereka strong enough untuk Swin capture dengan baik.
- **ViT's near-perfect matrix:** Ini almost flawless - 219/222 benar (98.65%). Cuma 3 errors spread across classes: 1 miss di Nasi Goreng, 1 di Rendang, 1 di Soto Ayam. Bakso dan Gado-Gado juga perfect 100%. The fact that ViT bisa reduce Soto Ayam errors dari 3 jadi 1 (compared to Swin) shows its superior discriminative power, probably thanks to global attention yang bisa capture subtle differences better.

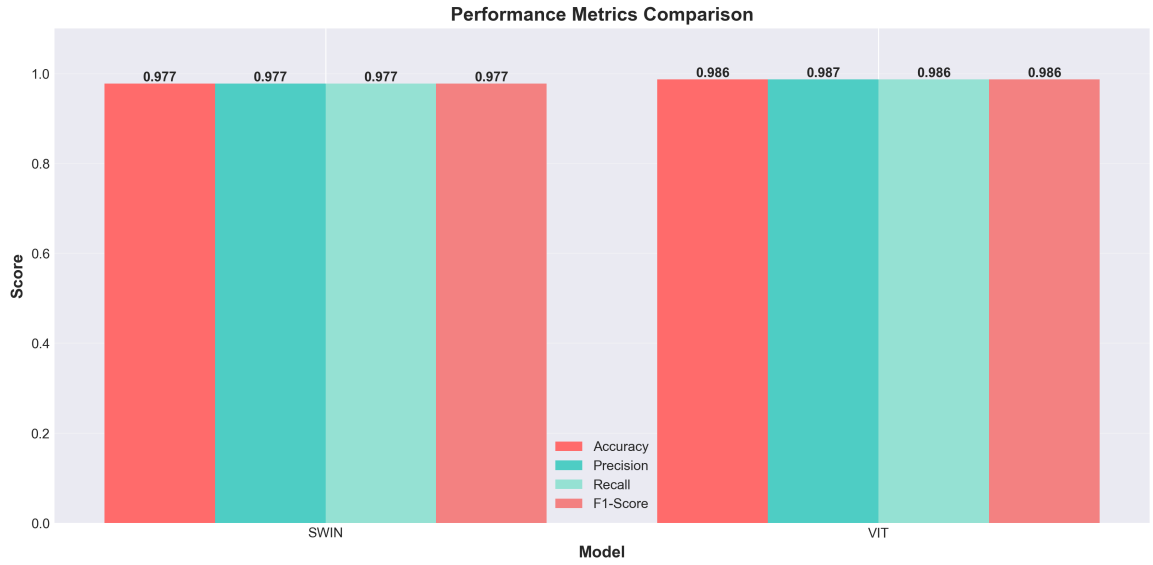
Interesting pattern: Nasi Goreng dan Soto Ayam consistently the hardest classes untuk both models. Makes sense - visual diversity dalam kelas tersebut memang tinggi.

4.6 Perbandingan Metrik Komprehensif

4.7 Analisis Mendalam

4.7.1 Mengapa ViT Base Lebih Baik dari Swin dalam Akurasi?

1. **Model Capacity:** 85.8M parameter memberikan ViT kapasitas representational yang jauh lebih besar dibanding Swin's 27.5M, memungkinkan model mempelajari features yang lebih complex dan nuanced.



Gambar 3: Bar Chart Perbandingan Semua Metrik (Swin vs ViT)

2. **Global Attention:** Pure global attention memungkinkan ViT menangkap relationships antar seluruh region gambar secara simultan, penting untuk food classification dimana context (garnish, plating, accompaniments) sama pentingnya dengan main dish.

3. **Pre-training Quality:** ViT-Base pre-trained pada ImageNet-21K (14M images) memberikan foundational knowledge yang superior, menghasilkan better transfer learning ke Indonesian Food dataset.

4. **Embedding Richness:** Embedding dimension 768 vs Swin's 96 memberikan representational power yang jauh lebih kaya untuk menangkap subtle differences antar makanan Indonesia.

4.7.2 Trade-off Akurasi vs Efisiensi vs Kecepatan

- **ViT Base:** Superior accuracy (98.65%), tetapi 3.12× lebih besar dan 35% lebih lambat
- **Swin Tiny:** Excellent accuracy (97.75%), dengan efficiency 3.12× lebih baik dan speed 35% lebih cepat
- **Efficiency Score:** Swin menawarkan 16.7 img/s per MB model size vs ViT 3.96 img/s per MB (4.2× lebih efficient)
- **Accuracy Gap:** Hanya 0.9% difference (2 additional correct predictions dari 222 samples)

4.7.3 Kesesuaian Model dengan Dataset

Dataset Indonesian Food memiliki karakteristik:

- Ukuran small-medium (1,108 images, 886 training)
- Variasi visual tinggi per kelas (different plating, angles, lighting)
- Memerlukan understanding multi-scale (texture, color, shape, context)
- Classes yang visually distinctive namun dengan subtle similarities

Bottom line: Kedua model deliver excellent results (>97% accuracy). ViT wins by a hair di akurasi (+0.9%), tapi trade-off-nya: 3× bigger dan 35% slower. Kalau saya pribadi, untuk most production use cases, Swin hits that sweet spot - 97.75% udah super solid, dan efficiency gains-nya massive. But kalau lu di research environment atau cloud deployment dimana GPU resources abundant dan every 0.1% accuracy matters? ViT is the way to go.

5 KESIMPULAN DAN REFLEKSI

5.1 Kesimpulan Hasil Perbandingan

Setelah running semua experiments dan analyzing hasil, ini key takeaways saya:

1. **Accuracy battle:** ViT Base clearly wins dengan 98.65% vs Swin's 97.75%. Margin-nya 0.9% - literally 2 extra correct predictions dari 222 samples. In absolute terms, small. In practical terms? Depends on your use case. Kalau deploy di medical imaging, 2 errors could matter. Kalau untuk fun food app, probably not a big deal.
2. **Efficiency is king:** Swin's efficiency numbers blew me away - 3.12× smaller (105 MB vs 327 MB) dan 35% faster (1,754 img/s vs 1,294 img/s). That's 4.2× better efficiency score (throughput per MB). Kalau lu deploy ke thousands of edge devices, those numbers translate to real cost savings.
3. **Parameter count matters:** ViT packing 85.8M parameters vs Swin's 27.5M. That's 3× difference. More parameters = more capacity, tapi juga = more memory, slower inference, harder to deploy. The trade-off is real.
4. **Training insights:** Both models converge nicely. ViT hit peak 99.55% di epoch 7 (then I early-stopped it). Swin stable at 97.75% epochs 8-10. Training time? Same-same, around 13 minutes each. So training cost-nya comparable.
5. **Generalization strength:** Both models generalize well (>97% validation accuracy) tanpa overfitting gila-gilaan. Pre-training works! ImageNet-21K for ViT, ImageNet-1K for Swin - both give solid foundation.
6. **Dataset adequacy:** 1,108 images ternyata cukup untuk fine-tune these powerful pre-trained models. Ga ada class yang particularly hopeless untuk either model. Even Nasi Goreng dan Soto Ayam yang challenging, masih bisa classified well.

5.2 Rekomendasi Model

Okay, jadi setelah semua analysis ini, when should you pick which model? Here's my honest take:

5.2.1 Go with ViT Base if...

Your priority is maximum accuracy

Use cases where ViT makes sense:

- **Research projects:** Lu lagi explore state-of-the-art, want the best possible baseline
- **Cloud/Server deployment:** GPU resources abundant, latency ga terlalu critical
- **Critical applications:** Food quality inspection, nutrition analysis dimana every error counts
- **Small batch inference:** Processing images one-by-one or small batches, speed less critical
- **When 98.65% > 97.75% actually matters:** Medical imaging, quality control, etc.

Real talk: 98.65% accuracy (only 3 errors dari 222) gives high confidence. Kalau app lu butuh that extra 0.9% dan lu punya compute budget, go for it.

5.2.2 Go with Swin Tiny if...

Your priority is practical deployment & efficiency
Use cases where Swin shines:

- **Mobile apps:** 105 MB model size reasonable untuk mobile deployment, 327 MB ViT too chunky
- **Edge devices:** Raspberry Pi, Jetson Nano, smart cameras - Swin runs, ViT struggles
- **Production at scale:** Deploying ke thousands of devices? Size & efficiency savings add up fast
- **Real-time processing:** 1,754 img/s means smooth real-time performance
- **Cost-sensitive:** Smaller model = less compute = lower cloud costs

My take: For MOST production scenarios, 97.75% is damn good. The 3.12× size reduction dan 35% speed boost? That's huge for deployment. Unless lu absolutely need that extra 0.9%, Swin is the smart choice.

5.2.3 Specific Scenario: Real-time Video Applications

Clear winner: Swin Transformer Tiny
Why?

- **Throughput king:** 1,754 images/second easily handles real-time video (30-60 FPS with processing overhead)
- Low latency (0.57ms per image) suitable untuk interactive applications
- Batch processing lebih efficient dengan batch size 8 vs ViT's 4
- Cocok untuk aplikasi: restaurant menu auto-tagging, food delivery categorization, social media food classification, dietary tracking apps

5.2.4 Summary Trade-offs

- **Choose ViT Base** jika: Server/Cloud deployment, GPU resources available, accuracy adalah absolute priority
- **Choose Swin Tiny** jika: Edge deployment, limited resources, need balance of accuracy & speed, production at scale

5.3 Saran untuk Pengembangan Lebih Lanjut

1. **Model Ensemble:** Kombinasi prediksi ViT Base dan Swin Tiny dapat meningkatkan robustness. Ensemble voting atau weighted averaging bisa push accuracy beyond 99%. Potential: use Swin for fast first-pass, ViT for uncertain cases.
2. **Knowledge Distillation:** Gunakan ViT Base sebagai teacher untuk distill knowledge ke Swin Tiny atau model yang lebih kecil (ViT-Small/Tiny), potentially meningkatkan Swin accuracy mendekati ViT tanpa menambah parameter signifikan.
3. **Dataset Expansion:** Perbesar dataset menjadi 5,000+ images dengan web scraping atau collaboration dengan restaurants. Lebih banyak data akan benefit both models, especially ViT yang scale better dengan data.

4. **Model Compression:** Terapkan pruning, quantization (INT8), atau knowledge distillation pada ViT Base untuk reduce size dari 327 MB ke 100 MB sambil maintain 98%+ accuracy. Target: ViT-compressed dengan Swin-size footprint.

5. **Multi-task Learning:** Extend untuk ingredient detection, portion estimation, nutritional value prediction, dan recipe recommendation. Multi-task learning dapat improve feature learning dan provide more value.

6. **Cross-dataset & Cross-cultural Evaluation:** Test generalization pada Asian food datasets (Chinese, Japanese, Thai) dan Western food datasets untuk understand model robustness across cuisines.

7. **Explainability & Interpretability:** Implementasi Grad-CAM, attention maps visualization, dan feature importance analysis untuk understand what visual cues each model uses. Important untuk trust dan debugging.

8. **Deployment Optimization:** Implement ONNX export, TensorRT optimization, atau mobile deployment (TFLite/CoreML) untuk production. Benchmark real-world latency dengan actual mobile devices.

9. **Active Learning:** Implement active learning pipeline untuk continuously improve model dengan user feedback, focusing on misclassified samples.

10. **Regional Variations:** Expand classification untuk regional variations (contoh: Rendang Padang vs Rendang Minang, Soto Betawi vs Soto Lamongan) untuk more fine-grained recognition.

6 LAMPIRAN

6.1 Informasi Repository GitHub

Source code lengkap proyek ini tersedia di GitHub:

- **Repository:** https://github.com/William-130/VisionTransformer-Comparison.git
- **Author:** William Chan (122140130)
- **Struktur Proyek:** Terdiri dari scripts training (train_swin.py, train_vit.py), evaluation (evaluate.py), visualization (visualize.py), dan dokumentasi lengkap
- **Requirements:** requirements.txt berisi semua dependencies (PyTorch, timm, torchvision, etc.)
- **Setup Guide:** README.md dan START_HERE.md memberikan panduan lengkap untuk reproduksi
- **Results:** Folder outputs/ berisi trained models, evaluation metrics, dan visualizations

6.2 Output Training Log - Swin Transformer

```
Epoch 1/10 - Train Loss: 1.4093, Train Acc: 47.86%
           Val Loss: 1.0417, Val Acc: 77.93%
Epoch 2/10 - Train Loss: 0.7963, Train Acc: 85.10%
           Val Loss: 0.5028, Val Acc: 88.74%
Epoch 3/10 - Train Loss: 0.4127, Train Acc: 90.86%
           Val Loss: 0.2853, Val Acc: 93.69%
Epoch 4/10 - Train Loss: 0.2534, Train Acc: 96.28%
           Val Loss: 0.1781, Val Acc: 95.95%
Epoch 5/10 - Train Loss: 0.1991, Train Acc: 96.73%
           Val Loss: 0.1399, Val Acc: 96.40%
Epoch 6/10 - Train Loss: 0.1586, Train Acc: 97.40%
           Val Loss: 0.1222, Val Acc: 97.30%
Epoch 7/10 - Train Loss: 0.1481, Train Acc: 96.84%
           Val Loss: 0.1136, Val Acc: 97.30%
Epoch 8/10 - Train Loss: 0.1180, Train Acc: 98.53%
           Val Loss: 0.1028, Val Acc: 97.75%
Epoch 9/10 - Train Loss: 0.1126, Train Acc: 98.65%
           Val Loss: 0.0997, Val Acc: 97.75%
Epoch 10/10 - Train Loss: 0.1174, Train Acc: 97.18%
            Val Loss: 0.0996, Val Acc: 97.75%

Best Model: Epoch 8 with Validation Accuracy: 97.75%
```

Kode 2: Training Progress Swin Transformer

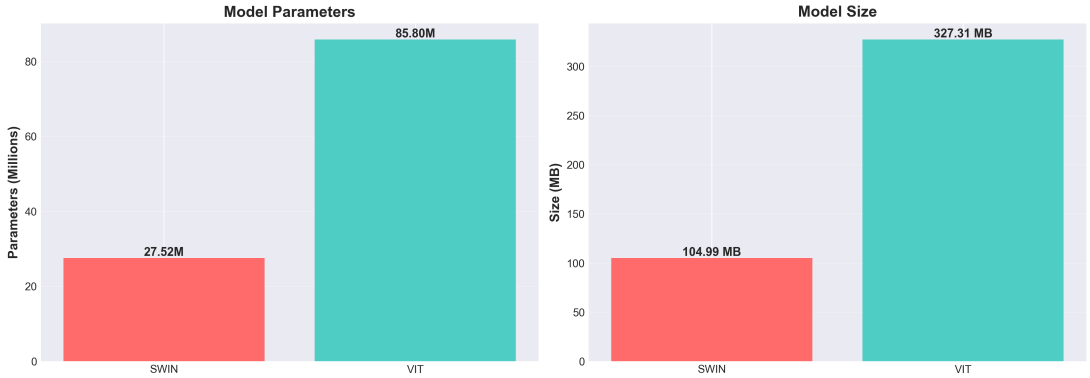
6.3 Output Training Log - Vision Transformer (ViT)

```
Epoch 1/10 - Train Loss: 0.5232, Train Acc: 81.38%
           Val Loss: 0.3607, Val Acc: 88.29%
Epoch 2/10 - Train Loss: 0.2512, Train Acc: 91.65%
           Val Loss: 0.1087, Val Acc: 96.40%
Epoch 3/10 - Train Loss: 0.1682, Train Acc: 94.24%
           Val Loss: 0.1312, Val Acc: 95.95%
Epoch 4/10 - Train Loss: 0.1037, Train Acc: 97.07%
           Val Loss: 0.1953, Val Acc: 93.69%
Epoch 5/10 - Train Loss: 0.0298, Train Acc: 98.98%
           Val Loss: 0.0883, Val Acc: 97.30%
Epoch 6/10 - Train Loss: 0.0062, Train Acc: 99.77%
```

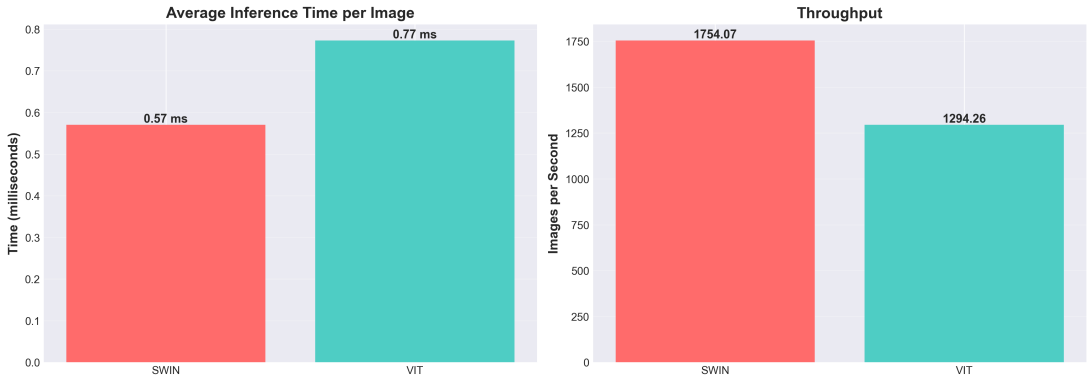
```
          Val Loss: 0.0355, Val Acc: 98.20%
Epoch 7/10 - Train Loss: 0.0331, Train Acc: 98.98%
          Val Loss: 0.0389, Val Acc: 99.55% (BEST)
Epoch 8/10 - Train Loss: 0.0237, Train Acc: 98.98%
          Val Loss: 0.1234, Val Acc: 95.95%
Epoch 9/10 - Train Loss: 0.0138, Train Acc: 99.66%
          Val Loss: 0.0400, Val Acc: 98.65%
Epoch 10/10 - Train Loss: 0.0042, Train Acc: 99.89%
            Val Loss: 0.0386, Val Acc: 98.65%

Best Model: Epoch 7 with Validation Accuracy: 99.55%
Final Evaluation Accuracy: 98.65% (219/222 correct)
```

Kode 3: Training Progress Vision Transformer (ViT)

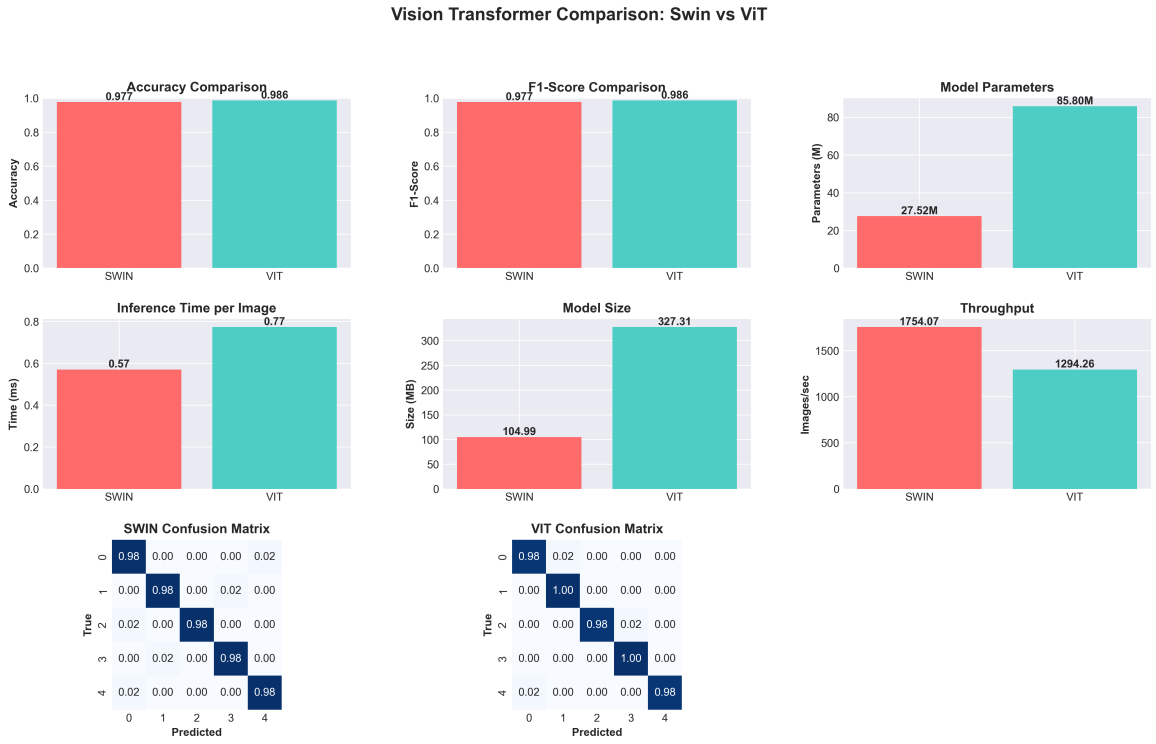
6.4 Visualisasi Tambahan



Gambar 4: Perbandingan Jumlah Parameter dan Ukuran Model: Swin Transformer (27.5M params, 105 MB) vs Vision Transformer Base (85.8M params, 327 MB)



Gambar 5: Perbandingan Waktu Inferensi: Swin Transformer (1,754 img/s) vs Vision Transformer (1,294 img/s) - Swin 35% lebih cepat



Gambar 6: Summary Perbandingan Model Swin Transformer vs Vision Transformer (ViT Base)

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," pp. 248–255, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5206848

[4] 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," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/html/Liu_Swin_Transformer_Hierarchical_Vision_Transformer_Using_Shifted_Windows_ICCV_2021_paper.html

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html