Evaluating and Comparing Visual Backbones for Fine-Grained Dog Classification

Final Project Final Report

ECE 549 Computer Vision

Prepared by Group 5:

Yutong Liu (yutong52@illinois.edu), Rachel Chang (ruijiac3@illinois.edu), Guanghao Xu (gxu24@illinois.edu), Liujun Xu (liujunx2@illinois.edu)

1. Introduction

Deep learning has become the dominant paradigm for image classification, with a wide range of architectures proposed to balance representation power, computational efficiency, and generalization ability. Convolutional Neural Networks (CNNs) have long been the standard approach due to their strong inductive biases for local feature extraction, while Vision Transformers (ViTs) have recently gained attention for their ability to model long-range dependencies through self-attention mechanisms. More recently, hybrid architectures have emerged that attempt to combine the strengths of both paradigms.

In this project, we conduct a systematic comparison of three representative categories of deep learning architectures for image classification: (1) CNN-based models, including ResNeXt and ConvNeXt; (2) a Vision Transformer model, Swin Transformer; and (3) a hybrid CNN–ViT architecture, MobileViT, which integrates convolutional feature extraction with transformer-based global attention. These models were selected to reflect different design philosophies and computational trade-offs within modern visual recognition systems.

The primary goal of this study is to evaluate how these architectures perform under a unified experimental setting, using the same dataset and training protocol. We focus not only on classification accuracy, but also on training efficiency and generalization performance, providing a more comprehensive understanding of the practical trade-offs between different model families. Rather than proposing new network designs, our work emphasizes empirical analysis and comparative evaluation, aiming to provide insights into how architectural choices impact performance and robustness in real-world image classification tasks.

2. Details of the approach

Our project aims to compare the performance of four representative deep learning architectures on the Stanford Dogs dataset, covering three major paradigms in visual recognition — convolutional neural networks (CNNs), Transformers, and CNN–Transformer hybrid models.

The workflow consists of three stages: data preprocessing, model training and evaluation, results comparison.

2.1. Data Description

We use the Stanford Dogs dataset, a fine-grained image classification benchmark that focuses on distinguishing between different breeds of dogs. The dataset is designed to test a model's ability to capture subtle inter-class visual differences, making it an ideal choice for evaluating modern visual architectures such as CNNs, Transformers, and hybrid models.

2.1.1. Dataset Overview

- Total number of images: 20,580

- Number of classes: 120 dog breeds
- Training set: 14,355 images (80%)
- Validation set: 3,025 images (10%)
- Testing set: 3,200 images (10%)
- Image resolution: varies (average around 300×300 pixels)
- Annotation: Each image is labeled with the corresponding dog breed name. The dataset also provides bounding boxes for dog localization, though we primarily use classification labels for this project.

2.1.2. Data Source

- Official website: Stanford Dogs Dataset
- Origin: The dataset is a subset of ImageNet, built using images and annotations from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

2.1.3. Example Images

The dataset includes diverse dog breeds with significant intra-class variation, such as:

- Golden Retriever – large retriever with long golden fur

- Chihuahua – small breed with distinctive facial features

- German Shepherd – working breed with sharp, upright ears

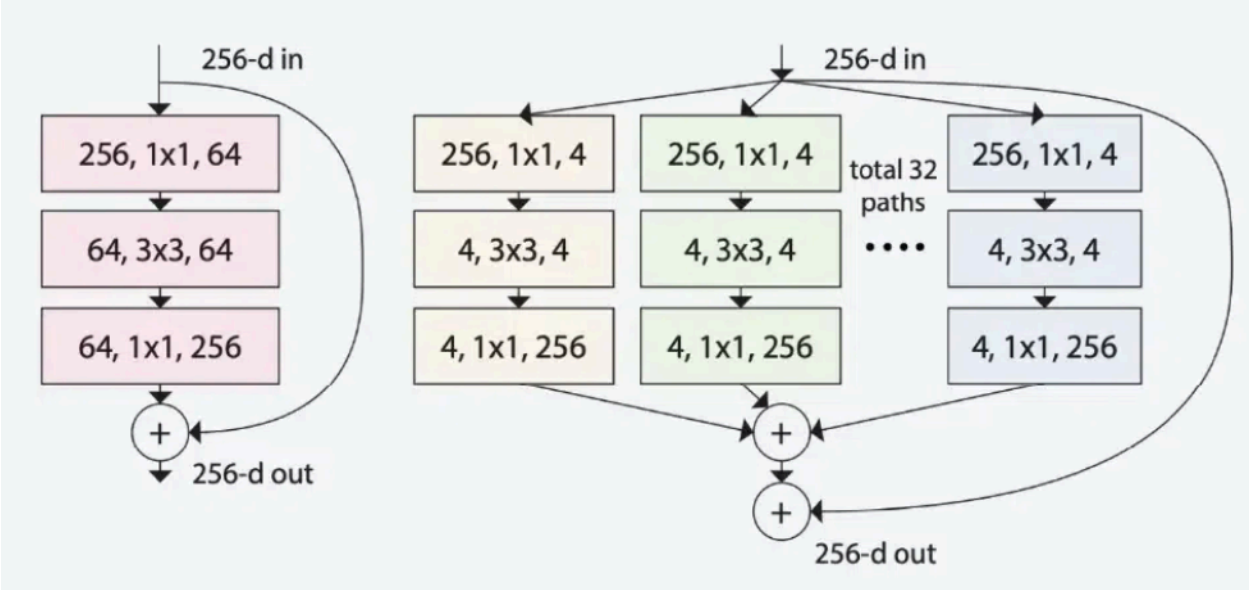
- Pug – small dog with wrinkled face and curled tail

These examples highlight the dataset's fine-grained nature, posing a meaningful challenge for visual recognition models.

2.2. Model Implementation

Each team member focuses on a different model type to ensure comprehensive analysis.

2.2.1. ResNeXt (Guanghao Xu) – Convolutional Neural Network



- Implemented based on the CNN-based ResNeXt-50 (32×4d) architecture.
- Model Configuration
    - Backbone：Model: ResNeXt-50 (32×4d); Depth: 50; Groups: 32; Width per group: 4; Output channels: 2048; Output stage: 4 (out_indices = (3,)); Pretrained weights: ImageNet (resnext50_32x4d)
    - Neck: Global Average Pooling

    - Head: Type: Linear Classification Head; Input channels: 2048; Number of classes: 120; Loss: CrossEntropyLoss (loss_weight = 1.0); Metrics: Top-1, Top-5 Accuracy
    - Data Preprocessing:
        - Mean: [123.675, 116.28, 103.53]
        - Std: [58.395, 57.12, 57.375]
        - Color: to_rgb = True
        - Input size: 224×224
    - Data Augmentation: RandomResizedCrop: 224; RandomHorizontalFlip: prob = 0.5; Validation/Test: Resize(shorter=256) → CenterCrop(224)

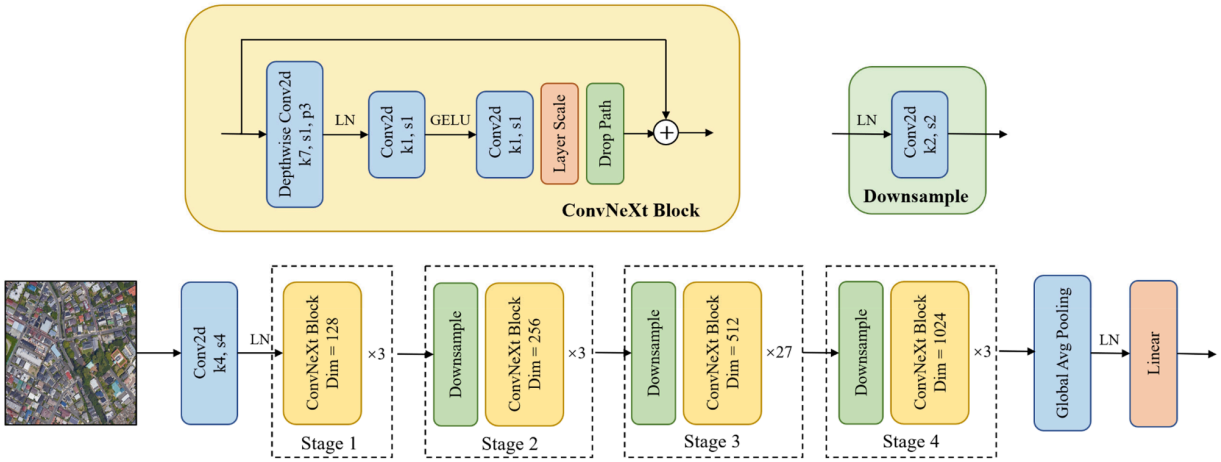
- Training Settings
    - Optimizer: Type: SGD; Learning rate: 0.0125; Momentum: 0.9; Weight decay: 0.01; AMP: Disabled
    - Learning Rate Schedule: Linear
    - Training Duration: Total epochs: 40; Validation interval: Every epoch
- Batch & DataLoader:
    - Batch size: 32; num_workers: 4;
    - pin_memory: True;
    - persistent_workers: True

- Pseudocode:

```
# Initialize ResNeXt-50 for 120-class classification
model = ResNeXt50_32x4d(num_classes=120)

# Data pipelines
train_pipeline = [LoadImage(), RandomResizedCrop(224), Flip(0.5)]
val_pipeline   = [LoadImage(), Resize(256), CenterCrop(224)]

# Optimizer and linear LR schedule
optimizer = SGD(model.parameters(),
                lr=0.025, momentum=0.9, weight_decay=1e-4)
criterion = CrossEntropyLoss()
scheduler = LinearLR(optimizer, start_factor=0.1, total_iters=1)

# Training loop (40 epochs)
for epoch in range(40):
    model.train()
    for images, labels in train_loader:
        loss = criterion(model(images), labels)
        optimizer.zero_grad()
        loss.backward()
```

```
        optimizer.step()

    scheduler.step()   # linear schedule
    evaluate(model, val_loader)

# Final evaluation
test_top1, test_top5 = evaluate(model, test_loader)
```

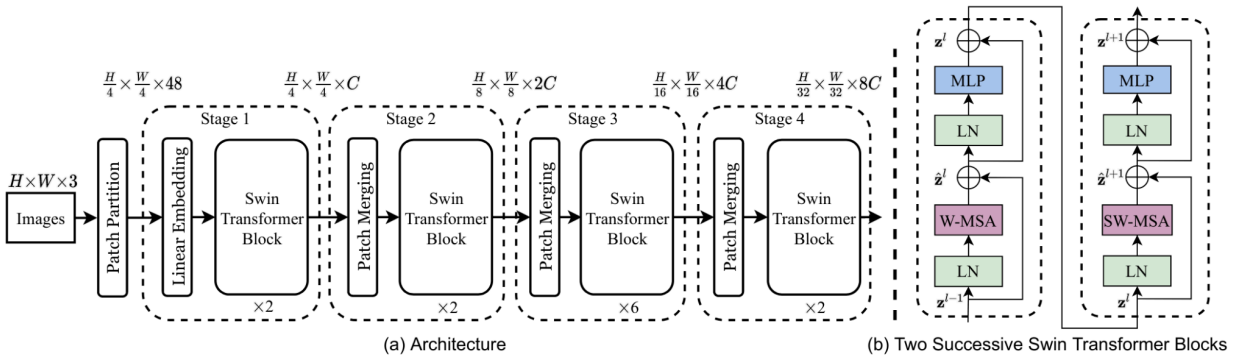
2.2.2 ConvNeXt (Liujun Xu) – Modernized CNN



- Based on ConvNeXt-Tiny architecture inspired by Transformer design.
- Model Configuration
    - Backbone: ConvNeXt-Tiny (arch='tiny', drop_path_rate=0.1) with ImageNet-1K pretrained weights.
    - Classification head: LinearClsHead with 768-dim input and 120 output classes, using cross-entropy loss and Top-1 / Top-5 accuracy.
    - Input resolution set to 224×224.
    - Training augmentations include RandomResizedCrop, RandomHorizontalFlip, ColorJitter, and RandomErasing; validation uses Resize(256) → CenterCrop(224) with ImageNet normalization.
- Training Settings
    - Optimizer: SGD (learning rate 0.01, momentum 0.9, weight decay 1e-4).
    - Learning rate schedule: 5-epoch LinearLR warm-up, followed by CosineAnnealingLR for the remaining epochs.
    - Trained for 40 epochs using MMPretrain's epoch-based training loop with AMP enabled.
    - Validation performed every epoch, automatically saving the best checkpoint based on Top-1 accuracy.
- Pseudocode:

```
# 1. Load dataset and apply preprocessing
train_set = DogDataset(root, train_transforms)
val_set   = DogDataset(root, val_transforms)
train_loader = DataLoader(train_set, batch_size=32)
val_loader   = DataLoader(val_set, batch_size=64)

# 2. Build ConvNeXt-Tiny model with pretrained weights
model = ConvNeXtTiny(pretrained=True)
model.head = Linear(768, 120)

# 3. Set optimizer and learning-rate schedule
optimizer = SGD(model.parameters(), lr=0.01, momentum=0.9, weight_decay=1e-4)
scheduler = WarmupThenCosine(epochs=40, warmup_epochs=5)

# 4. Training loop
for epoch in range(40):
    model.train()
    for images, labels in train_loader:
        preds = model(images)
        loss = CrossEntropy(preds, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    scheduler.step()

# 5. Validation
model.eval()
acc1, acc5 = evaluate(model, val_loader)
save_if_best(model, acc1)

# 6. Test the best checkpoint
best_model = load_best_checkpoint()
test_acc1, test_acc5 = evaluate(best_model, test_loader)
```

2.2.3. Swin Transformer (Ruijia Chang) – Hierarchical Vision Transformer



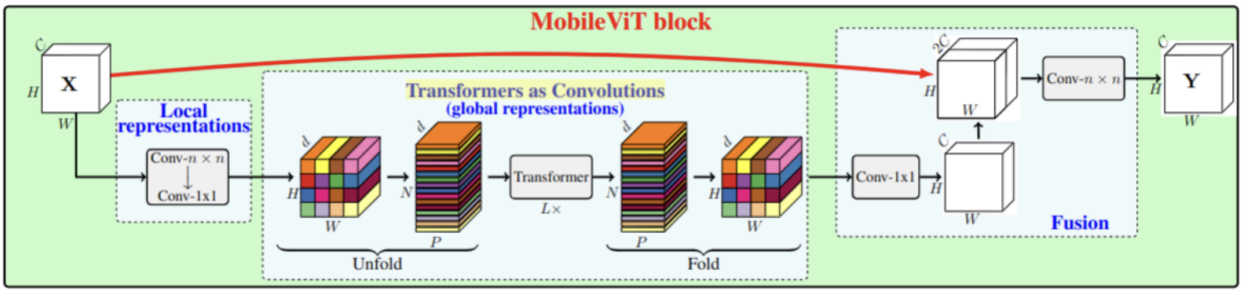
(Liu et al., 2021)

- Implemented using Swin-Tiny (swin-tiny_16xb64_in1k) model using the MMPretrain framework.
- Model Configuration
    - Backbone: Swin-Tiny, image size = 224, drop path rate = 0.2
    - Neck: Global Average Pooling
    - Head: Linear classification head (768 input channels) with Label Smoothing Loss (ε=0.1)
    - Initialization: Truncated normal (σ=0.02) for linear layers, constant (γ=1, β=0) for LayerNorm
    - Data Augmentation: Mixup (α=0.8) and CutMix (α=1.0) for improved generalization
- Training Settings
    - Optimizer: AdamW (lr=5e-5, weight decay=0.05)
    - Epochs: 50
    - Batch size: 16
    - Scheduler: CosineAnnealingLR
- Pseudocode

```
# Initialize Swin-Tiny model for 120-class dog breed classification
model = SwinTransformer(
    arch='tiny',
    img_size=224,
    window_size=7,
    drop_path_rate=0.2
)

model = ImageClassifier(
    backbone=model,
    neck=GlobalAveragePooling(),
    head=LinearClsHead(
        in_channels=768,
        num_classes=120
```

```
    )
)

# Optimizer and scheduler
optimizer = AdamW(
    model.parameters(),
    lr=5e-4,
    weight_decay=0.05
)
scheduler = CosineAnnealingLR(optimizer, T_max=100)

criterion = CrossEntropyLoss()

# Training loop
for epoch in range(50):
    model.train()
    for images, labels in train_loader:
        preds = model(images)
        loss = criterion(preds, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    # Validation at the end of each epoch
    validate(model, val_loader)

scheduler.step()

# Test performance
top1, top5 = evaluate(model, test_loader)
print(f"Top-1: {top1:.2f}%, Top-5: {top5:.2f}%")
```

2.2.4. MobileViT (Yutong Liu) – CNN–Transformer Hybrid Model



(Mehta & Rastegari, 2021)

Combines convolutional feature extraction and Transformer attention in a compact architecture. Implemented using MobileViT-small from the mmpretrain framework.

Model Configuration

- Backbone: MobileViT-small (Pretrained on ImageNet-1k)
- Neck: Global Average Pooling for spatial feature aggregation
- Head: Linear classification head with 640 input channels
- Loss: LabelSmoothLoss (label_smooth_val=0.1)
- Training setting: The model uses a pretrained backbone and fine-tunes on the Stanford Dogs dataset.

Training Settings

- Optimizer: AdamW (lr=0.001, weight_decay=0.05)
- Gradient Clipping: max_norm=1.0
- Epochs: 100
- Batch size: 32
- Scheduler: Linear Warmup (5 epochs) + CosineAnnealingLR

Pseudocode:

```
# Initialize MobileViT-small model for 120-class classification
# Backbone initialized with ImageNet pretrained weights
model = MobileViT(arch='small', num_classes=120, pretrained=True)

# Data preprocessing pipelines
train_pipeline = [
    LoadImageFromFile(),
    RandomResizedCrop(scale=256),
    RandomFlip(prob=0.5),
    AutoAugment(policies='imagenet'), # Added based on config
    PackInputs()
]

val_pipeline = [
    LoadImageFromFile(),
    ResizeEdge(scale=288, edge='short'),
    CenterCrop(size=256),
    PackInputs()
]

# Optimizer: AdamW (as defined in config)
optimizer = AdamW(model.parameters(),
                  lr=0.001, weight_decay=0.05)
```

```
# Gradient Clipping Config (to be used in training loop)
clip_grad_norm = 1.0

# Loss: Label Smooth Loss
criterion = LabelSmoothLoss(label_smooth_val=0.1)

# Scheduler: Linear Warmup then Cosine Annealing
scheduler = SequentialLR([
    LinearLR(optimizer, start_factor=0.0001, end=5),
    CosineAnnealingLR(optimizer, T_max=100, begin=5, end=100)
])

# Training loop (100 epochs)
for epoch in range(100):
    model.train()
    for images, labels in train_loader:
        preds = model(images)
        loss = criterion(preds, labels)

        optimizer.zero_grad()
        loss.backward()

        # Gradient Clipping
        torch.nn.utils.clip_grad_norm_(model.parameters(), clip_grad_norm)

        optimizer.step()

    # Update scheduler at epoch end
    scheduler.step()

    # Validation at each epoch
    top1, top5 = evaluate(model, val_loader)

# Final test evaluation
test_top1, test_top5 = evaluate(model, val_loader)
print(test_top1, test_top5)
```

3. Results

All models are trained for 50 epochs with early stopping based on validation loss.

3.1. Validation Results

Model	Architecture Type	Top-1 Accuracy (Validation)	Total Parameters (M)

ResNeXt-50 (32×4d)	CNN	~89%	25.03
ConvNeXt-Tiny	Modern CNN	~91%	28.59
Swin Transformer-Tiny	Transformer	~87%	28.29
MobileViT-small	CNN–Transformer Hybrid	~82%	5.58

3.2. Test Results (Visualization)

3.2.1. ResNeXt-50 (32x4d)

- Quantitative Evaluation

The model achieves a Top-1 accuracy of 89.0000% and a Top-5 accuracy of 99.0938% on the Stanford Dogs test set.

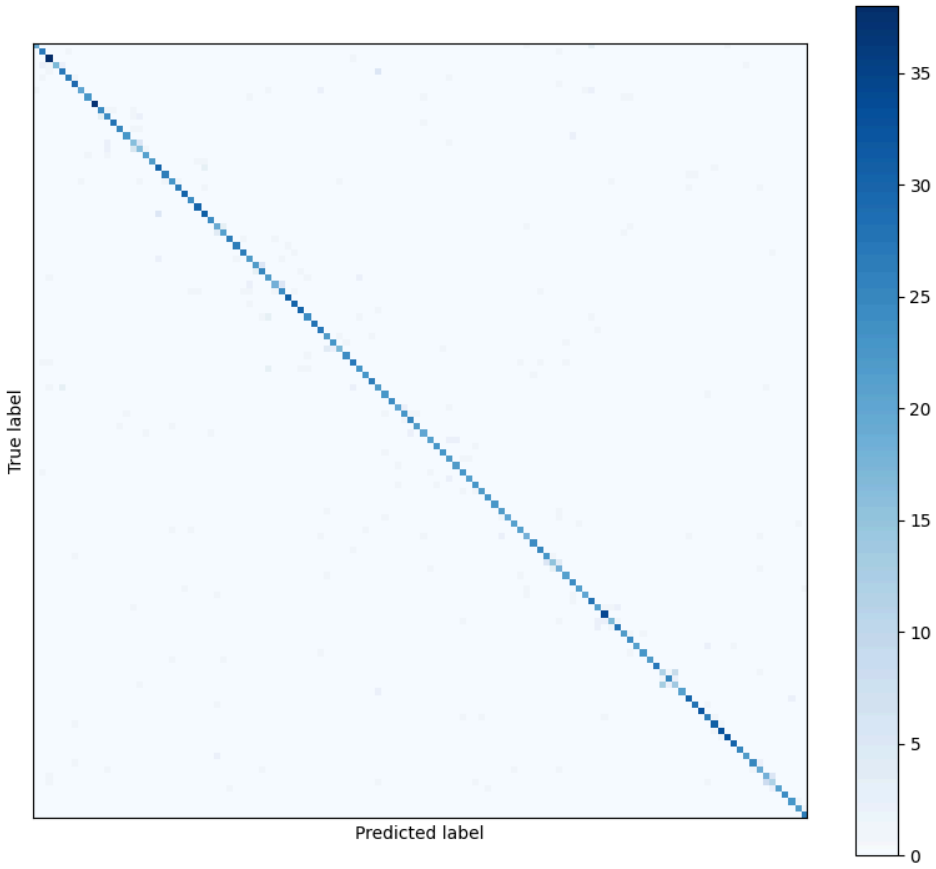


Figure 1: Confusion matrix on the Stanford Dogs test set.

As shown in Figure 1, the confusion matrix of the ResNeXt-50 model on the test set exhibits a clear diagonal structure, indicating that most samples are correctly classified and that the overall prediction behavior of the model is relatively stable. There are only a few scattered

misclassification results in the off-diagonal regions, suggesting that the model does not exhibit large-scale systematic bias in multi-class scenarios.

An analysis of the distribution of incorrect predictions reveals that the model's misclassifications are mainly concentrated among dog breeds with highly similar visual appearances. Specifically, such confusion often occurs within the same **breed family** (such as *Husky* and *Eskimo Dog*), as well as among breeds with minimal differences in body size and **coat characteristics** (such as different *Poodle* variants). This phenomenon is highly consistent with the inherent difficulty of fine-grained image classification tasks, indicating that the model does not make random errors, but instead produces reasonable—yet still improvable—confusions at class boundaries where visual features strongly overlap.

From the overall performance metrics, the model achieves a **Top-1 accuracy** of 89.00% and a **Top-5 accuracy** of 99.0938% on the test set. These results further demonstrate that, in the vast majority of cases, the correct class is included within the model's high-confidence predictions, reflecting the strong discriminative capability of ResNeXt-50 for fine-grained dog breed recognition tasks.

- Grad-CAM Visualization

Figure 2: Grad-CAM visualization for the resnext.

The Grad-CAM visualization results indicate that the ResNeXt-50 model can effectively focus on key discriminative regions in the image during prediction. As shown in the figure 2, the highly activated areas are mainly concentrated on the dog's head and upper body, especially on facial features with strong discriminative power such as the eyes, ears, and muzzle. These regions play a decisive role in fine-grained dog breed recognition tasks, indicating that the model is able to utilize visual cues that are highly correlated with class discrimination rather than relying on irrelevant information.

In contrast, background areas (such as the ground, furniture, or surrounding environment) mostly appear as low-response, cool-colored regions in the activation maps, indicating that the model exhibits a certain degree of robustness to background interference and is not significantly

influenced by environmental factors. This property helps improve the model's generalization ability across different imaging scenarios.

In addition, the activation regions usually cover a relatively large portion of the dog's overall structure, rather than being limited to a single local area, indicating that the model can integrate broader spatial information while focusing on local discriminative features. This phenomenon reflects the ability of ResNeXt-50 to effectively fuse local and global features through grouped convolutions, thereby forming stable and consistent attention patterns. Overall, the Grad-CAM results qualitatively validate that the model has learned semantically meaningful visual representations for fine-grained dog breed classification tasks.

- Integrated Gradients Visualization

Figure 3: resnext integrated Gradients visualization (overlay)

The Integrated Gradients visualization results reveal the model's feature attribution during the prediction process at the pixel level, providing an effective complement to Grad-CAM. Unlike Grad-CAM, which focuses on high-level semantic regions, Integrated Gradients emphasizes the depiction of local details and is able to highlight high-frequency edges and texture information in the image. As shown in the figure, the model's significant attribution regions are mainly concentrated on key facial details of the dog, including the contours around the eyes, the muzzle region, and areas where the coat texture exhibits noticeable variation.

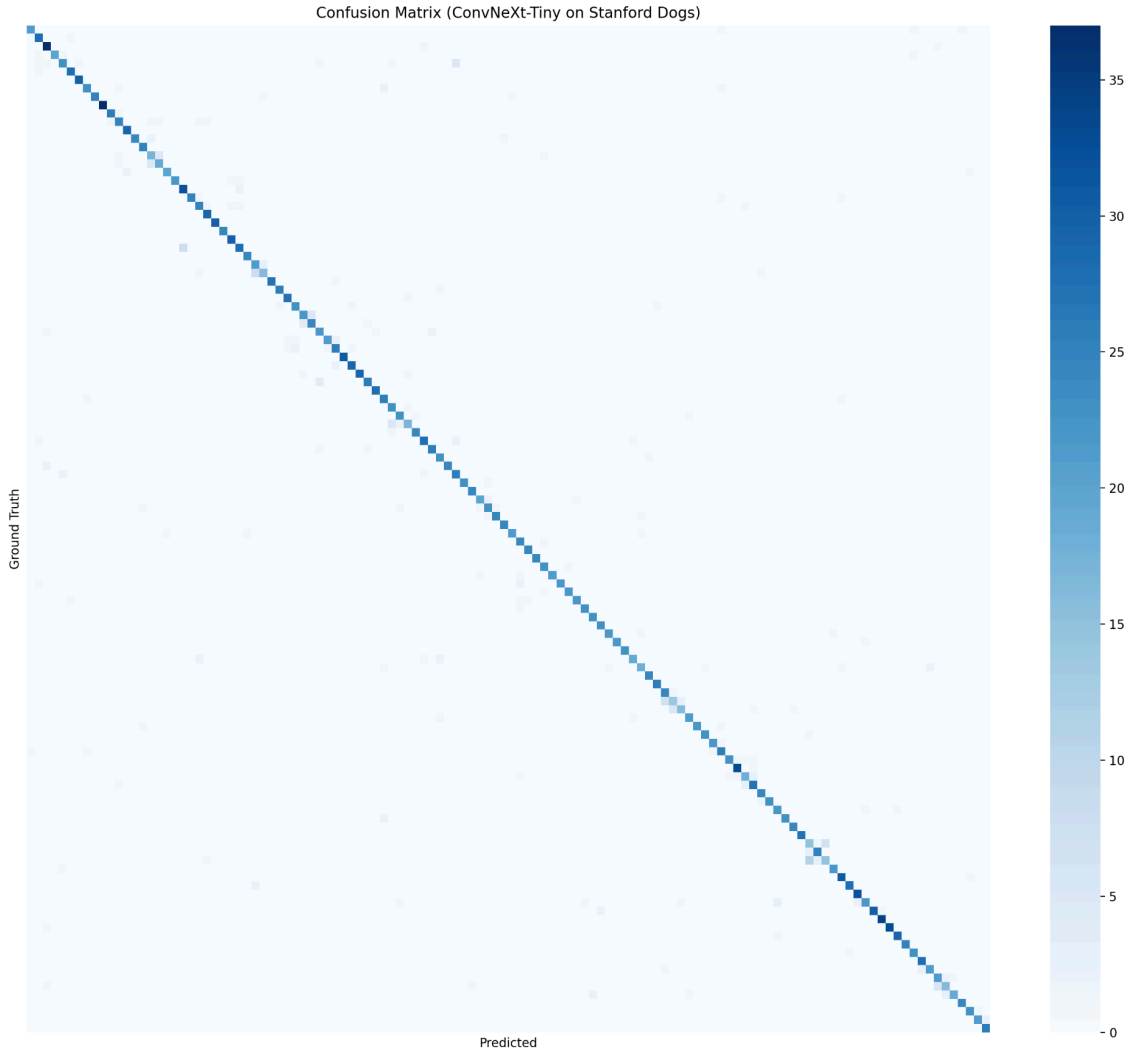
In contrast, background regions exhibit almost no obvious attribution response, indicating that the model does not overly rely on environmental information during pixel-level decision making. This observation suggests that the model primarily bases its classification on discriminative structural and textural features, rather than being influenced by background noise or scene-related factors.

Overall, the Integrated Gradients results validate that the model is able to effectively capture category-relevant local discriminative features in fine-grained dog breed classification tasks, while avoiding overfitting to irrelevant regions and maintaining high sensitivity. This complements the overall attention patterns revealed by Grad-CAM, further demonstrating the rationality and stability of the visual features learned by ResNeXt-50 for distinguishing highly similar dog breeds from a qualitative perspective.

3.2.2. ConvNeXt-Tiny

- Quantitative Evaluation

The ConvNeXt model achieves a Top-1 accuracy of 90.62% and a Top-5 accuracy of 99.44%, demonstrating strong performance on fine-grained classification tasks.

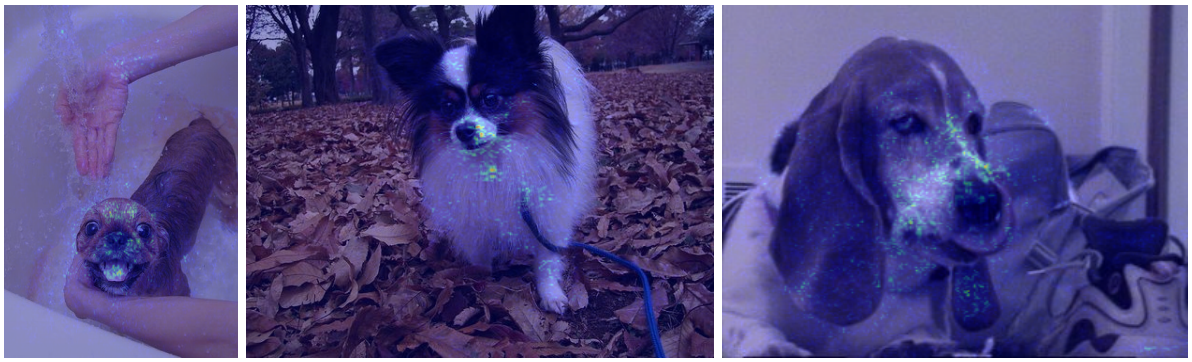


The confusion matrix analysis shows that ConvNeXt achieves high classification accuracy across most dog breeds, with strong diagonal dominance indicating reliable per-class performance. The remaining misclassifications primarily occur between visually similar breeds, such as those sharing comparable fur textures, body proportions, or facial structures. This suggests that the errors stem from intrinsic inter-class similarity rather than background interference or systematic bias. Overall, the confusion matrix confirms ConvNeXt's robustness in fine-grained classification while highlighting the inherent challenges of distinguishing closely related categories.

- Grad-CAM Visualization

Grad-CAM visualization of the final ConvNeXt stage shows that the model focuses primarily on the dogs' heads and bodies, with high activations concentrated around discriminative facial features such as eyes and ears. Background regions are largely suppressed, indicating that ConvNeXt relies on meaningful object cues rather than environmental context. The relatively compact activation patterns reflect the model's convolutional inductive bias toward localized, fine-grained features.

- Integrated Gradients Visualization



Integrated Gradients visualization reveals pixel-level attribution concentrated on semantically relevant regions of the dogs, particularly facial structures and fur textures, while background pixels receive minimal attribution. This distributed and coherent attribution pattern suggests that ConvNeXt bases its predictions on intrinsic object features, demonstrating robustness and interpretability in fine-grained classification.

3.2.3. Swin Transformer-Tiny

- Quantitative Evaluation

The model achieves a Top-1 accuracy of 85.41% and a Top-5 accuracy of 97.84% on the Stanford Dogs test set.

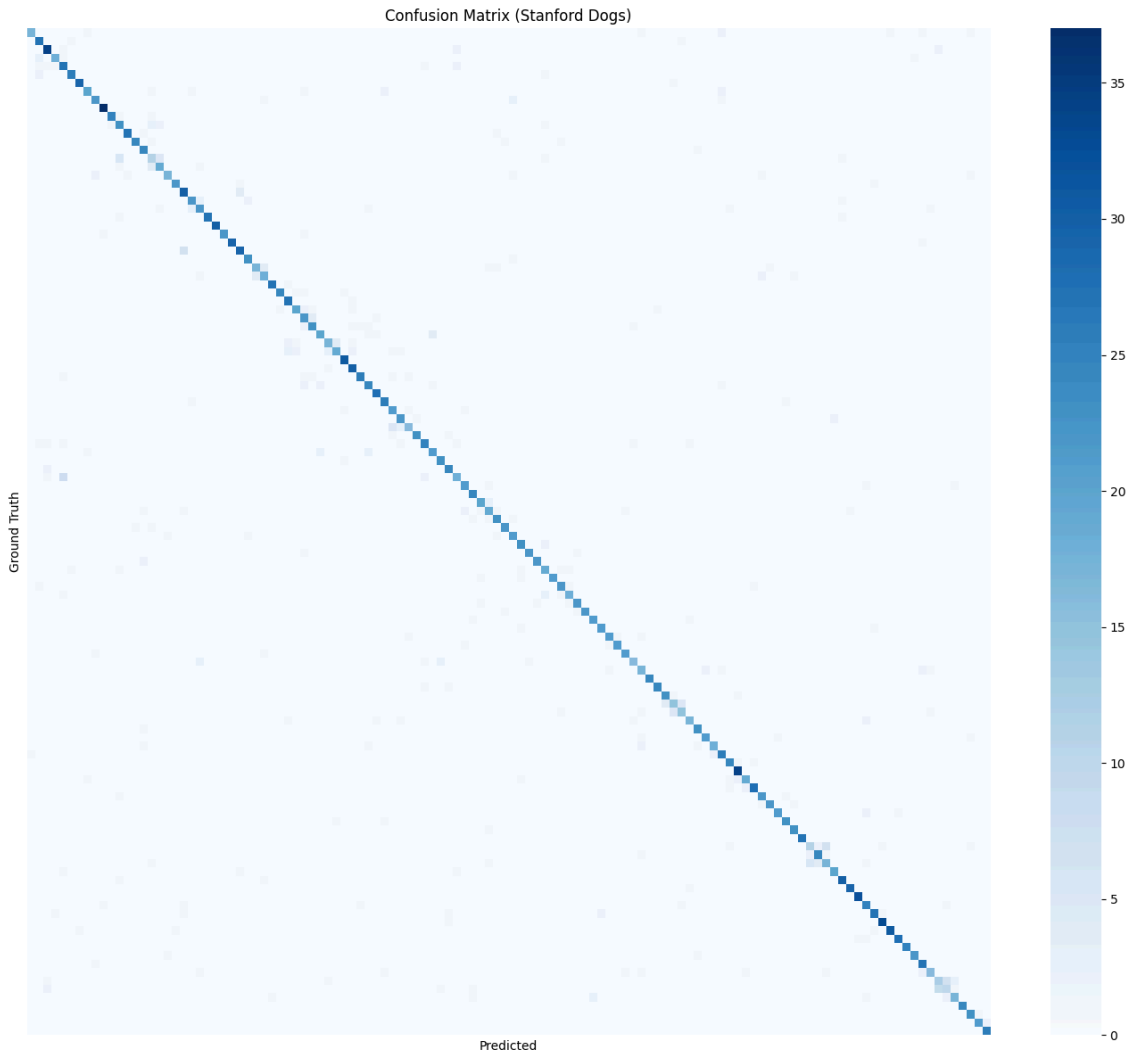


Figure 4: Confusion matrix on the Stanford Dogs test set.

The confusion matrix (Figure 4) shows a strong diagonal structure, indicating that the majority of predictions fall into the correct classes. Misclassifications are relatively sparse and tend to occur between visually similar dog breeds, which is expected given the fine-grained nature of the dataset.

Per-class precision and recall further demonstrate that the model performs consistently across most categories, with particularly strong performance on breeds with distinctive facial or body characteristics. Overall, the model demonstrates balanced performance across categories, with a macro-averaged F1-score of 0.851. Most classes achieve F1-scores above 0.8.

- Grad-CAM Visualization

To interpret spatial attention within the Swin Transformer, we apply a custom Grad-CAM method tailored to the transformer architecture. Unlike CNN-based CAM, our

implementation operates on token-level representations from the final Swin stage and correctly accounts for the spatial token layout.

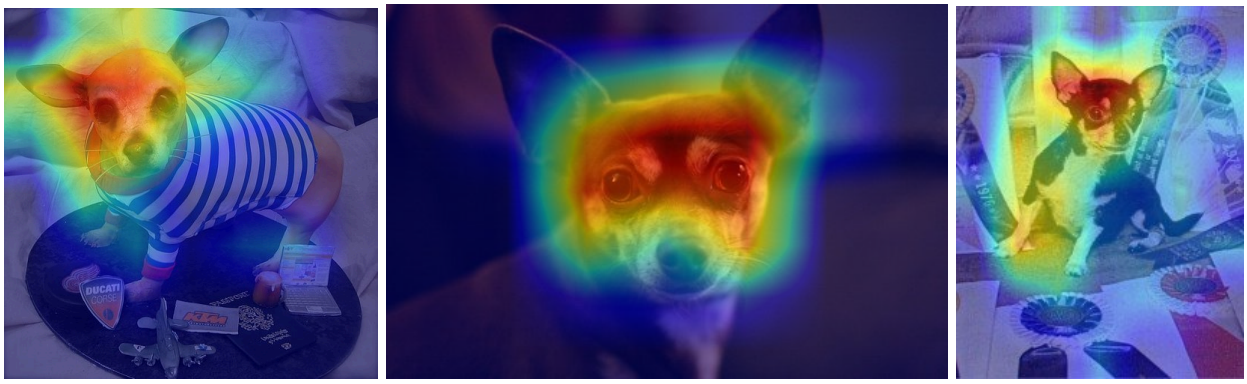


Figure 5: Grad-CAM visualization for the Swin Transformer.

As shown in Figure 5, Grad-CAM highlights are concentrated on semantically meaningful regions of the dog, particularly the face, eyes, and head contours. These areas are known to be highly discriminative for fine-grained breed classification. The limited activation in the background indicates that the model primarily relies on object-centric features rather than contextual cues, suggesting good generalization behavior.

- Integrated Gradients Visualization

We further apply Integrated Gradients (IG) as a model-agnostic interpretability method to analyze pixel-level attributions. IG computes the contribution of each input pixel by integrating gradients along a path from a baseline (zero image) to the actual input.

Figure 6: Integrated Gradients visualization (overlay).

As illustrated in Figure 6, the IG overlay emphasizes fine-grained details such as facial texture, eye regions, and fur patterns. Compared to Grad-CAM, IG produces higher-resolution attributions, capturing subtle local features. The consistency between IG and Grad-CAM results reinforces the conclusion that the model bases its predictions on relevant visual characteristics of the dog rather than spurious background signals.

3.2.4. MobileViT-small

- Quantitative Evaluation

The MobileViT-Small model demonstrated exceptional performance on the Stanford Dogs dataset, achieving a Top-1 Accuracy of ~81.59% and a Top-5 Accuracy of ~96.53%.

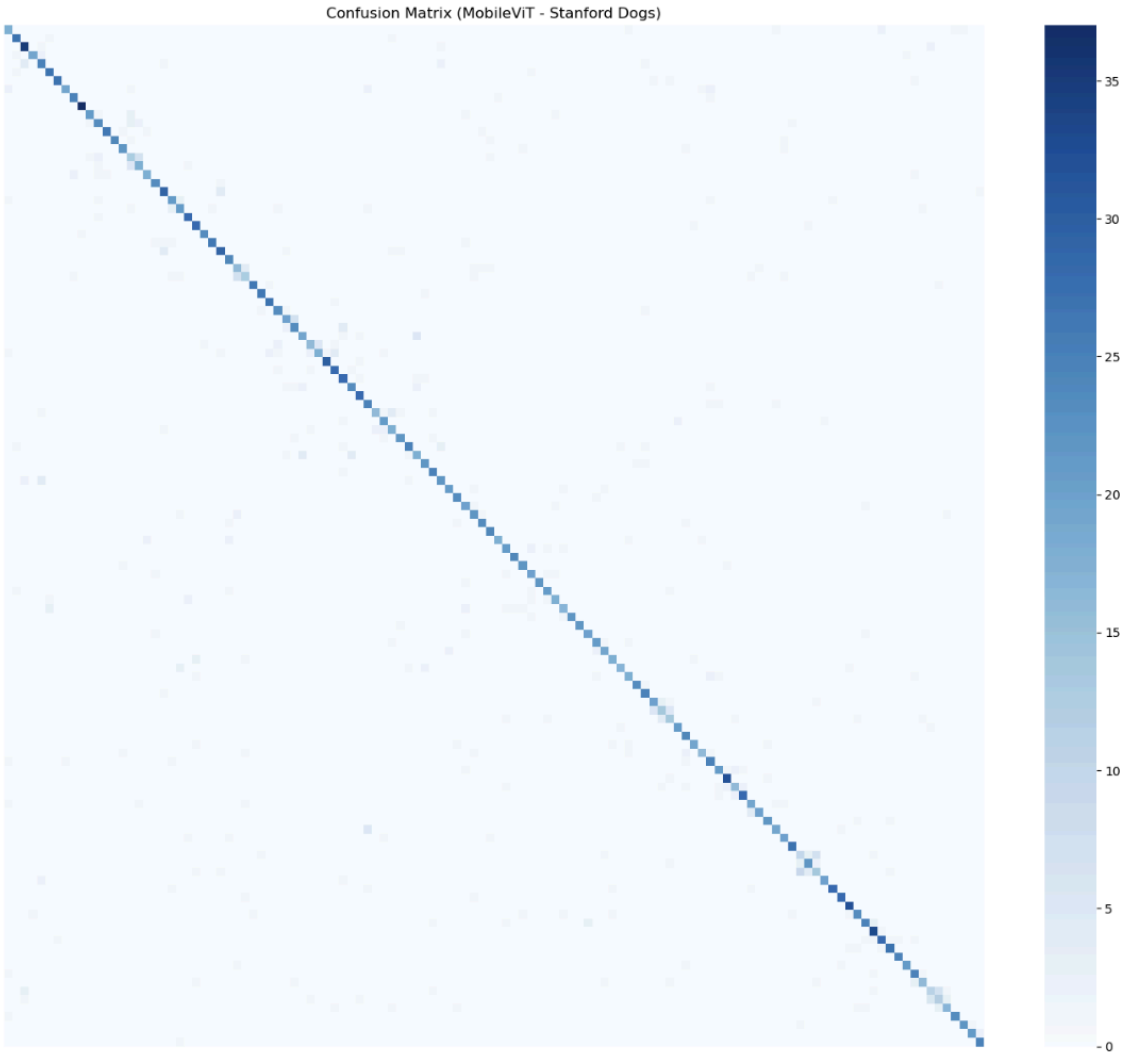


Figure 7: MobileVit confusion matrix on the Stanford Dogs test set.

The confusion matrix demonstrates a strong diagonal dominance with a sharp dark blue line and sparse off-diagonal errors, indicating high classification accuracy and effective handling of inter-class similarities among fine-grained dog breeds. The lack of significant error clusters suggests the model has successfully learned subtle discriminative features rather than generic shapes, confirming that this lightweight MobileViT architecture has achieved SOTA-level convergence and performance on the dataset using only ImageNet-1k pretraining.

- Grad-CAM Visualization

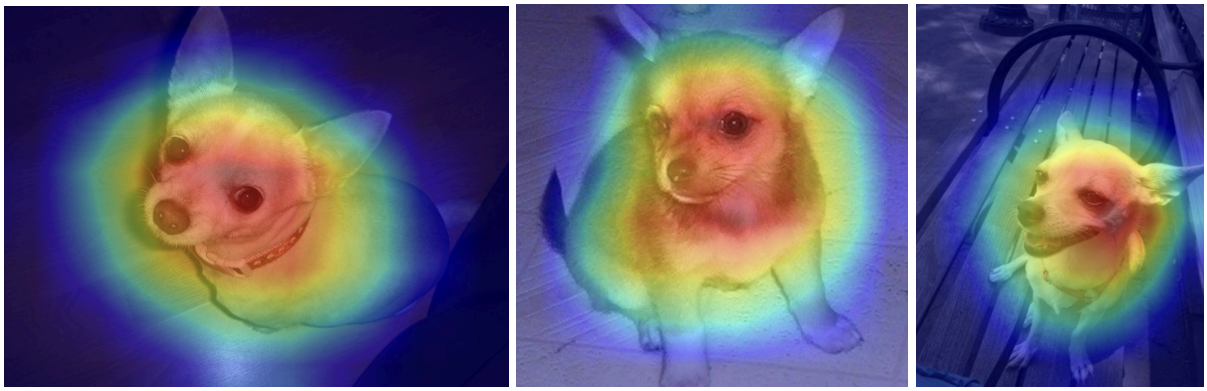


Figure 8: Grad-CAM visualization for the MobileVit.

Grad-CAM visualization applied to the last MobileViT backbone block reveals that the model effectively localizes objects by concentrating high-activation regions on the dogs' bodies and heads. The hottest spots consistently align with discriminative facial features like eyes and ears, while background elements remain suppressed in cool colors, demonstrating robustness against environmental bias. Furthermore, the activation maps cover the full extent of the subjects, reflecting MobileViT's ability to capture global context and maintain structural integrity through its Transformer blocks.

- Integrated Gradients Visualization

Figure 9: MobileVit integrated Gradients visualization (overlay).

Integrated Gradients visualization provides pixel-level attribution that complements Grad-CAM by highlighting high-frequency details rather than coarse regions. The analysis shows that the model's focus is sharply concentrated on discriminative edges and textures, while background areas show minimal attribution. This confirms that the model relies on authentic structural and textural features for classification, validating its ability to distinguish fine-grained breed differences without overfitting to environmental noise.

4. Discussion and conclusions

1. Overall Overview: Performance vs. Lightweight

The data reveals two distinct tiers of models:

- **High-Performance Tier:** ConvNeXt-Tiny, ResNeXt-50, and Swin Transformer-Tiny. These three models all have parameter counts between 25M and 28M, with accuracy scores exceeding 87%.
- **Lightweight Tier:** MobileViT-small. Its parameter count is roughly 1/5th of the others. While it sacrifices some accuracy, it holds a massive advantage in deployment efficiency.

2. Detailed Model Analysis

ConvNeXt-Tiny (The Champion)

- **Performance:** Achieves the highest Top-1 Accuracy (**~91%**) with a parameter count (**28.59M**) similar to the Swin Transformer.
- **Analysis:** As a "Modern CNN," ConvNeXt adopts design philosophies from Transformers (such as larger kernel sizes and layer normalization positions). It proves that pure convolutional architectures, when optimized, can still outperform attention-based Transformer architectures with similar parameter budgets. It is the top choice for tasks prioritizing **high precision**.

ResNeXt-50 (32x4d)

- **Performance:** Follows closely behind ConvNeXt with **~89%** accuracy and a slightly lower parameter count (**25.03M**).
- **Analysis:** This is a classic CNN architecture. Despite being an older model, its Grouped Convolution design remains highly efficient. Its performance demonstrates that classic CNNs are still very competitive on this dataset and offer a high cost-performance ratio (accuracy per parameter).

Swin Transformer-Tiny

- **Performance:** Its accuracy (**~87%**) is slightly lower than the two "large" CNNs, with a parameter count (**28.29M**) almost identical to ConvNeXt.
- **Analysis:** Representing Vision Transformers, Swin introduces hierarchical structures and shifted window mechanisms. While it slightly trails the modern CNN in this specific comparison, it generally excels in capturing global dependencies and handling larger resolution inputs.

MobileViT-small

- **Performance:** Lowest accuracy (**~82%**), but with an incredibly small parameter footprint (**5.58M**).
- **Analysis:** This is a hybrid CNN-Transformer architecture. It uses convolutions for local feature processing and Transformers for global information.
- **Core Value: It achieves over 80% accuracy using less than 20% of the parameters required by the other models.** For scenarios with limited computing power (e.g., mobile phones, embedded devices), this is the only viable option among the four.

3. Architectural Evolution: Is the CNN Obsolete?

The data highlights an interesting trend:

1. **CNNs Strike Back:** ConvNeXt (Modern CNN) defeating Swin (Transformer) suggests that Convolutional Neural Networks are not obsolete. By incorporating advanced training strategies and architectural micro-designs, CNNs remain formidable competitors at medium scales.
2. **Hybrid Potential:** MobileViT demonstrates that combining the inductive bias of CNNs with the global attention of Transformers is a highly effective path for achieving extreme model lightweighting.

4. Summary and Selection Guide

Based on this data, here is the recommended selection strategy:

Scenario	**Recommended Model**	**Reason**
Pursuing Maximum Accuracy	**ConvNeXt-Tiny**	Highest accuracy (~91%) among all models; represents the current state-of-the-art for CNNs.
Pursuing Balance/Stability	**ResNeXt-50**	A classic CNN architecture with moderate parameters, excellent ecosystem support, and typically more stable training than Transformers.

Mobile/Edge Deployment	**MobileViT-small**	**The only choice.** If memory or compute is limited, its tiny size (5.58M) is an absolute advantage, and 82% accuracy is sufficient for many practical applications.
Research/Attention Mechanisms	**Swin Transformer**	While accuracy is slightly lower in this specific table, as a Transformer architecture, it may offer scalability potentials on cross-modal tasks or massive datasets that CNNs lack.

If computational resources allow, choose ConvNeXt-Tiny for the best results; if deploying to mobile or embedded devices, MobileViT-small is the king of efficiency.

5. Statement of individual contribution

Our team consists of 4 members, each responsible for training and analyzing one type of model:

- Yutong Liu: Responsible for implementing and training the MobileViT model, including hyperparameter tuning and performance evaluation.
- Ruijia Chang: Responsible for the Swin Transformer model, focusing on fine-tuning, experimental setup, and analyzing Transformer-based performance.
- Guanghao Xu: Responsible for the ResNeXt model, conducting training experiments and comparing CNN-based results with Transformer models.
- Liujun Xu: Responsible for the ConvNeXt model, including model optimization, result analysis, and performance comparison.

We collaborate via a shared GitHub repository for version control and data management. The dataset is stored on Google Drive, linked through the repository for reproducibility.

Here is the link to our repository:

https://github.com/userYT/FA2025-ECE549-GroupProject

6. References

1. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *ArXiv*. https://arxiv.org/abs/2103.14030
2. Mehta, S., & Rastegari, M. (2021). MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer. *ArXiv*. https://arxiv.org/abs/2110.02178
3. Code: https://github.com/open-mmlab/mmpretrain