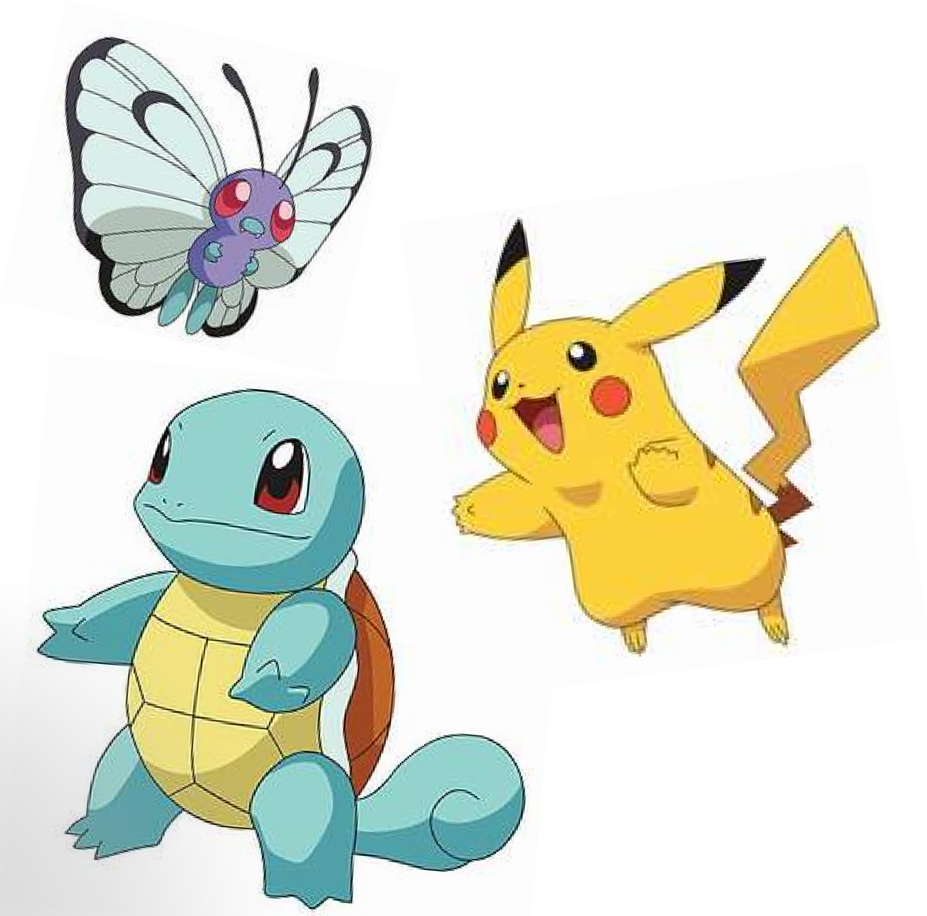
POKEMON CLASSIFICATION



Liel Layney , Michael Lurya , Omri Arie

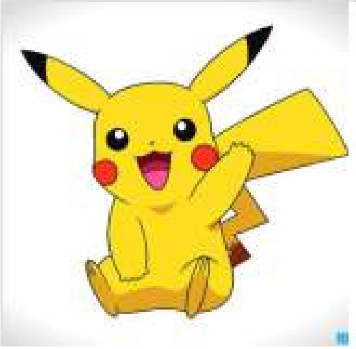
Pokémon Image Classification

The motivation Imagine an app that instantly recognizes any Pokémon—from trading cards to fan art—so collectors can auto-catalog and track their collections. By solving the fine-grained classification challenge posed by 151 visually similar species, this project lays the groundwork for any application needing fast, accurate, and scalable image recognition in both entertainment and real-world domains.

The goal of this project is to accurately classify Pokémon images into their respective species, leveraging deep learning techniques.

Data Collection & Description

Dataset Source:



Pokemon gen1 151 classes's images classification

All 151 classes's images of Pokemon for classifications.

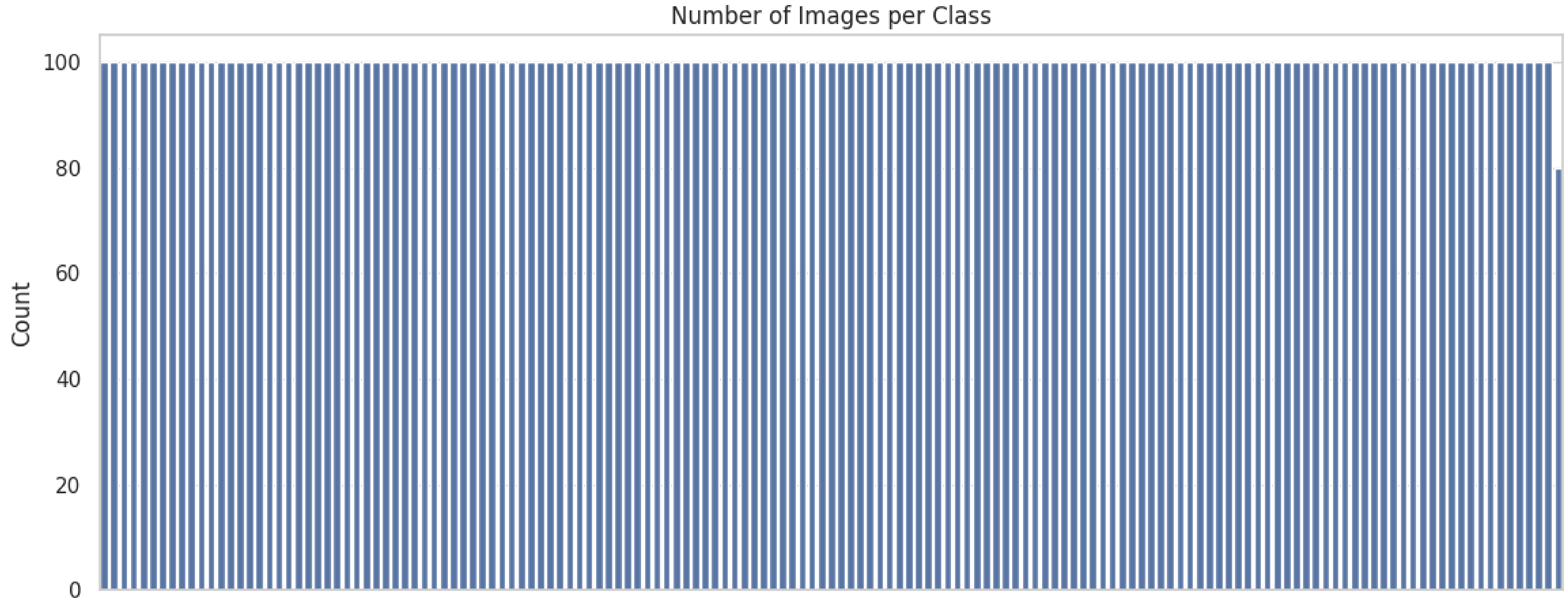
kaggle.com

Dataset Details:

- **Total images: over 15,100**
- **Classes (Pokémon species): 151**
- **The data comes from a wide variety of sources**

Exploratory Data Analysis

Class Distribution



The distribution of the images is balanced

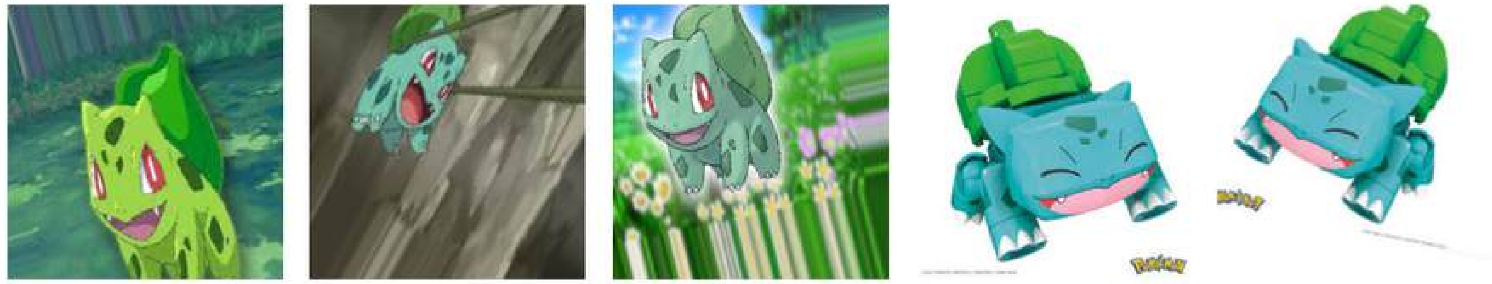
Main Challenges

Pokémon evolution

Pokémon image texture

Pokémon image source

The tilt of the images

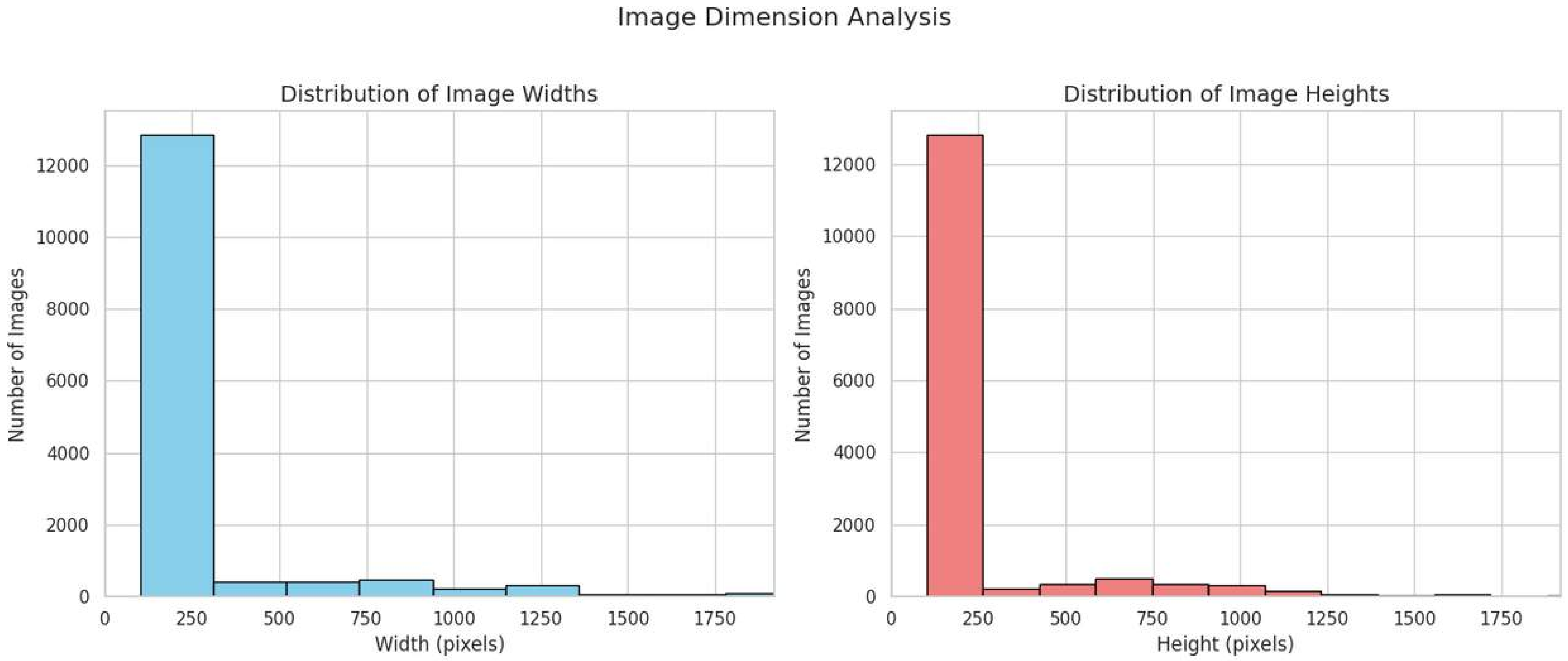


Variability in Image Dimensions

```
Most common image sizes:
(240, 240): 12264 images
(1280, 720): 144 images
(225, 225): 136 images
(300, 168): 54 images
(1920, 1080): 48 images
(734, 1024): 36 images
(500, 500): 35 images
(646, 646): 35 images
(600, 600): 34 images
(600, 825): 34 images
```

Image Size Analysis

Most images size is 240x240 pixels



so we choose to resize all the images to 240x240 pixels

Preprocessing Pipeline

1. Resize images to consistent dimensions.
2. Normalize pixel values to [0, 1] range.
3. Split dataset: Training: 70% , Validation: 15% , Testing: 15%

Model 1 – Basic CNN

Architecture:

- Convolutional Layers:
  - 3× Conv2D → ReLU → MaxPool
    - Filters: 32 → 64 → 128
- Fully Connected Layers:
  - Flatten → Dense(256) → ReLU → Dropout(0.5) → Dense(151)

Training Setup:

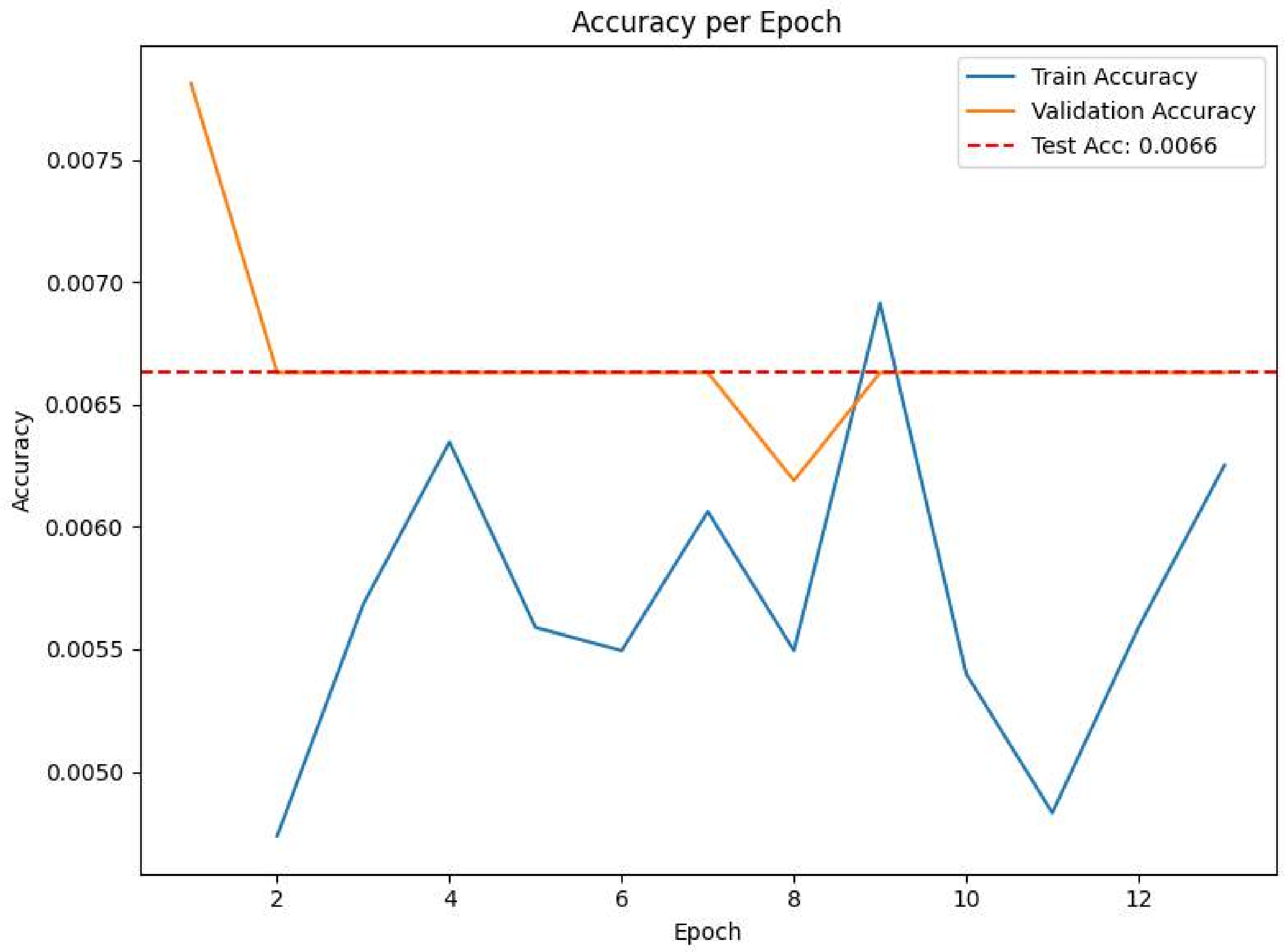
- Optimizer: Adam (learning rate = 0.001)
- Loss Function: Cross-entropy loss
- Epochs: Up to 25 (Early stopping)

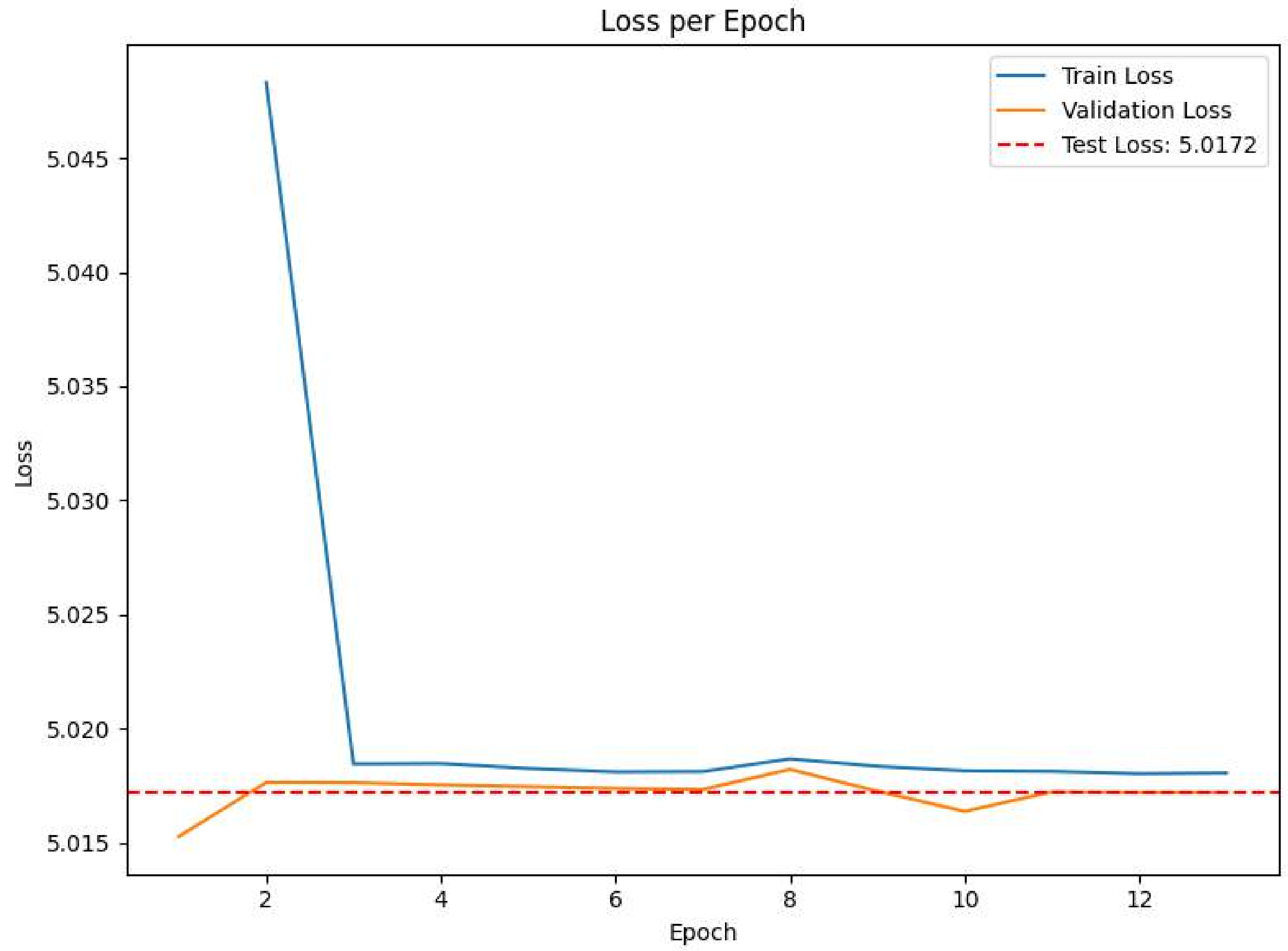
Results:

- Test Accuracy: 0.66%

Observations:

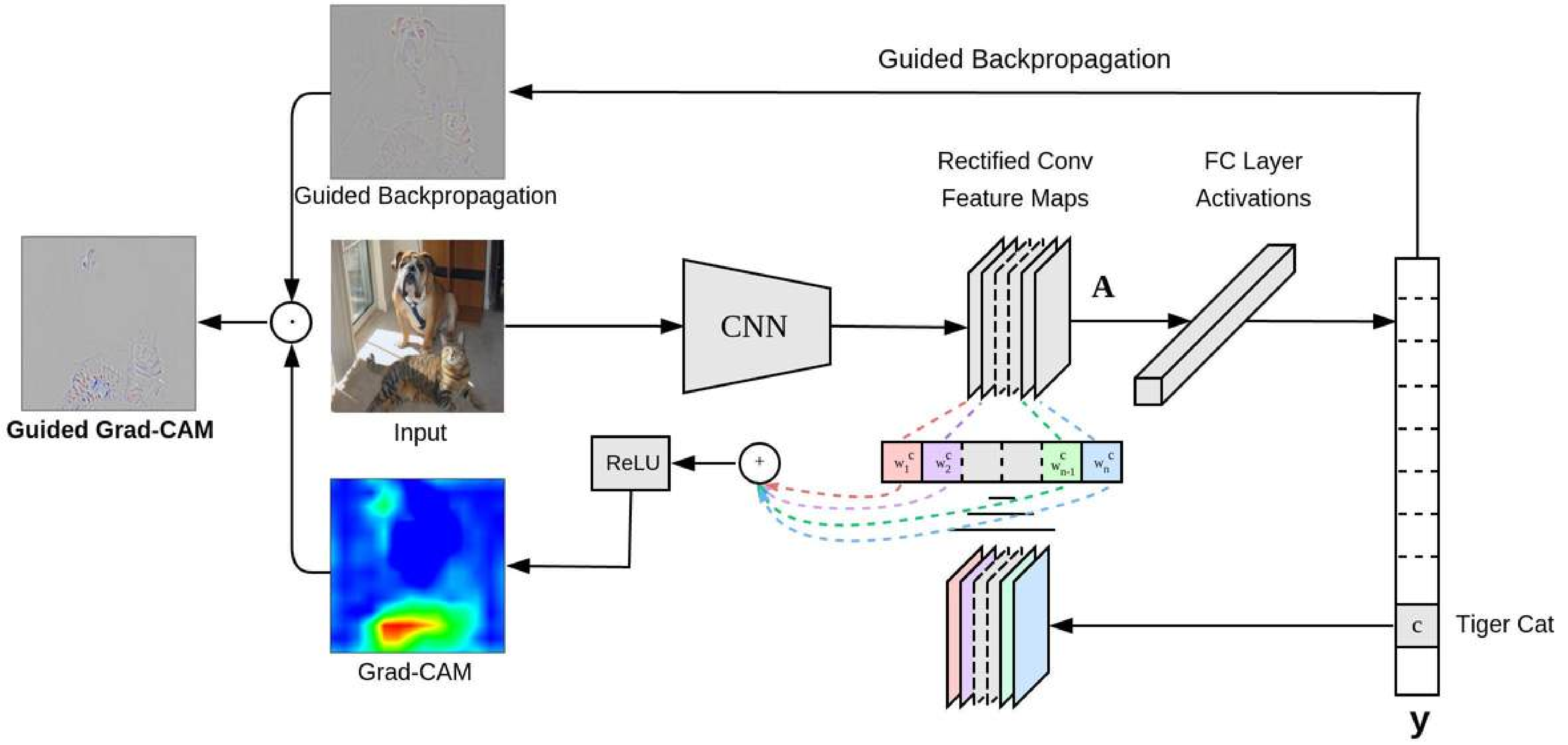
Not Good initial learning, lets try another model.

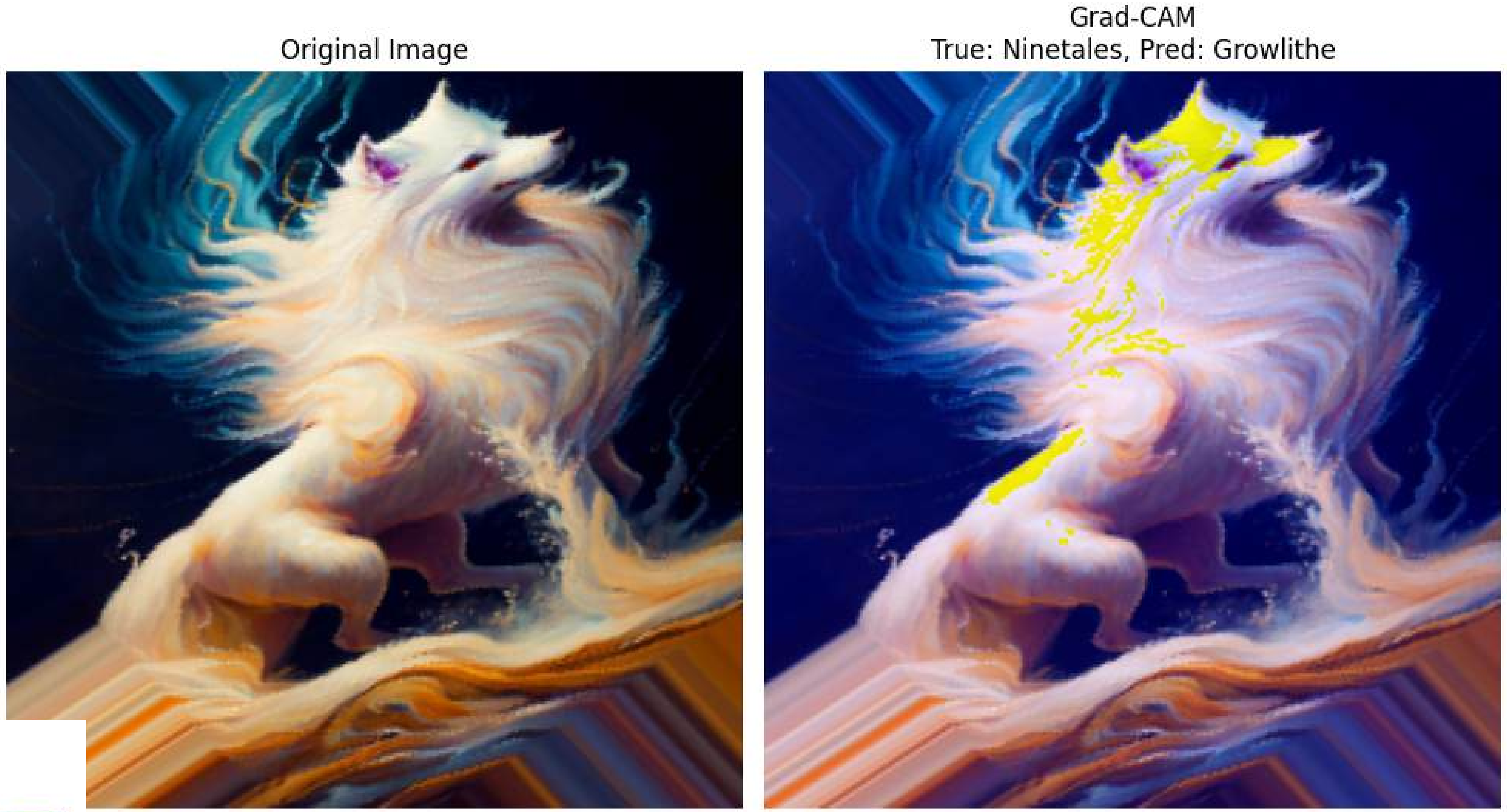


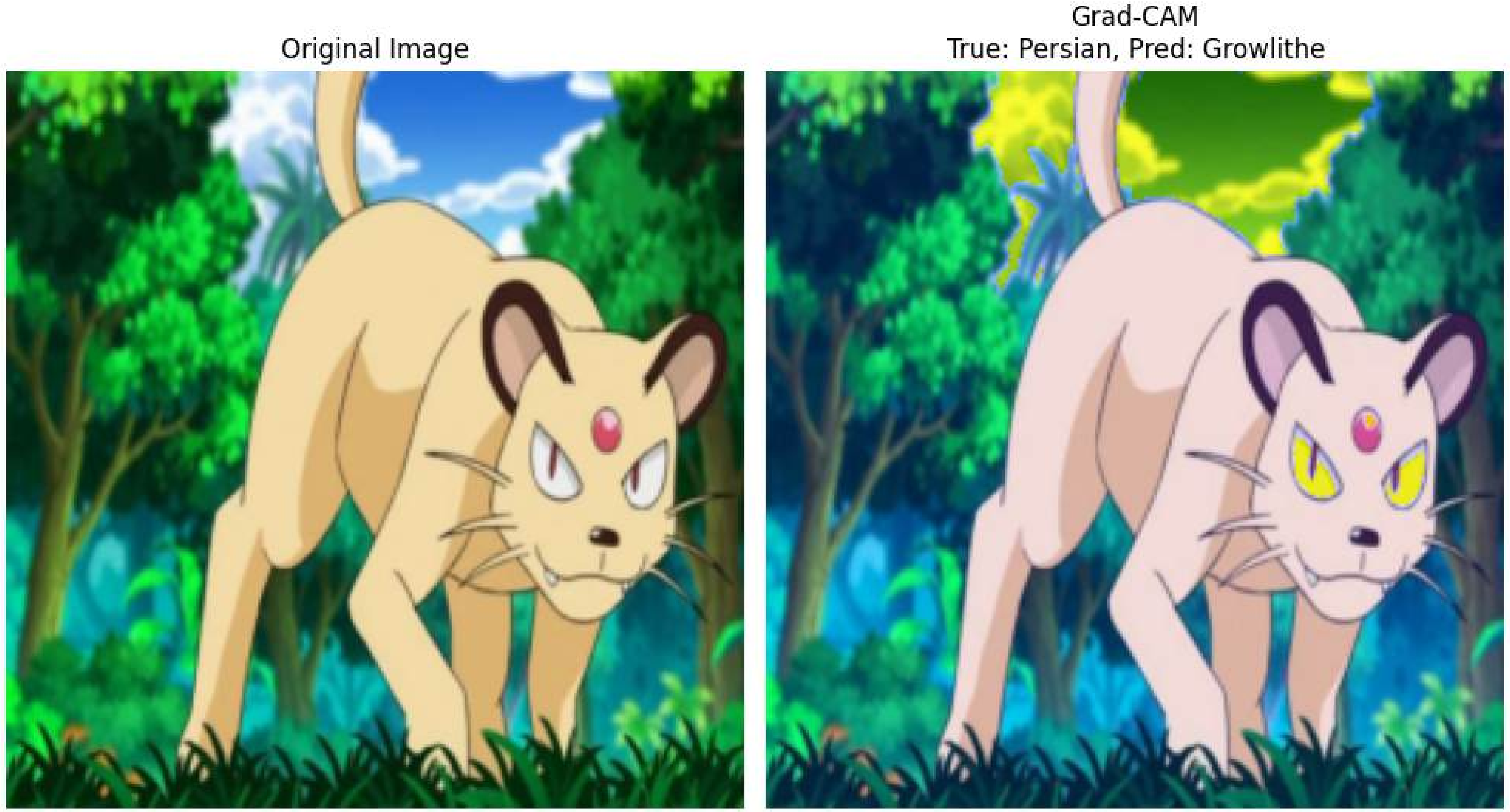


Interpreting CNN Decisions with Grad-CAM

In our project, we used Grad-CAM (Gradient-weighted Class Activation Mapping) to visualize and validate the regions of each input image that our CNN considers most important for its predictions. Grad-CAM works by backpropagating the class score through the network to compute gradients with respect to the feature maps of a chosen convolutional layer, averaging those gradients to obtain channel weights, and then combining the weighted feature maps into a coarse heatmap. We then upsample and overlay this heatmap on the original image to highlight, for example, the exact object parts or lesion boundaries that drive the model's decision.



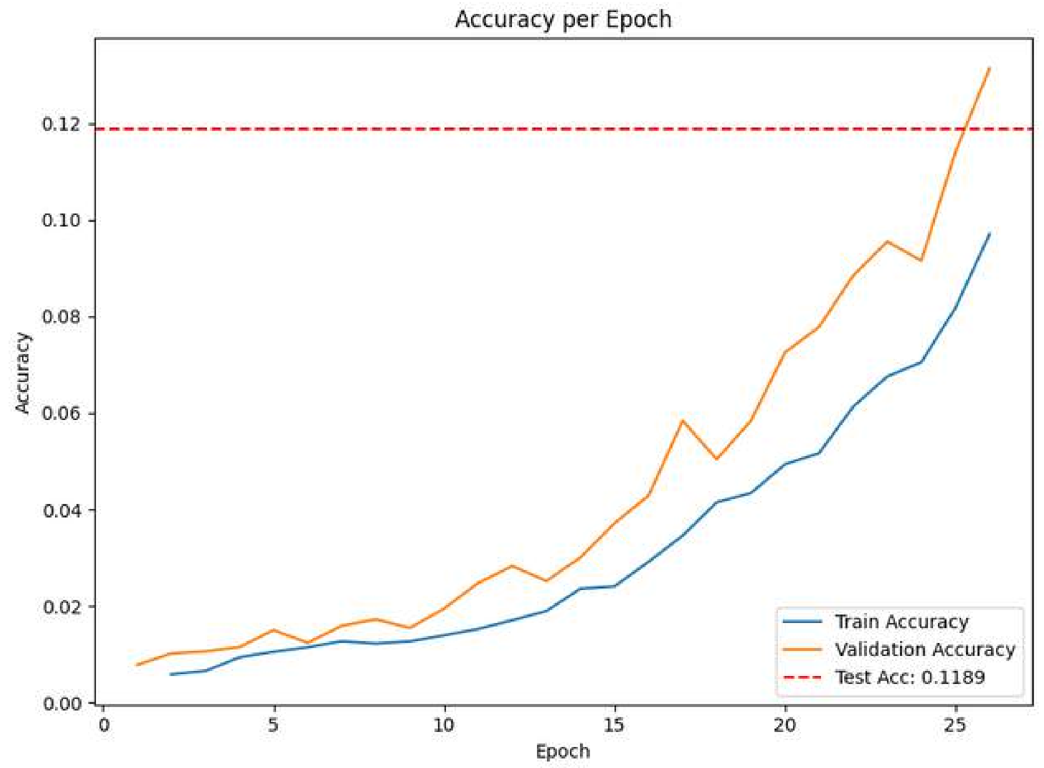

Original Image
Grad-CAM
True: Ninetales, Pred: Growlithe


Original Image
Grad-CAM
True: Persian, Pred: Growlithe

Model 2 – Custom CNN Modern (Residual CNN)

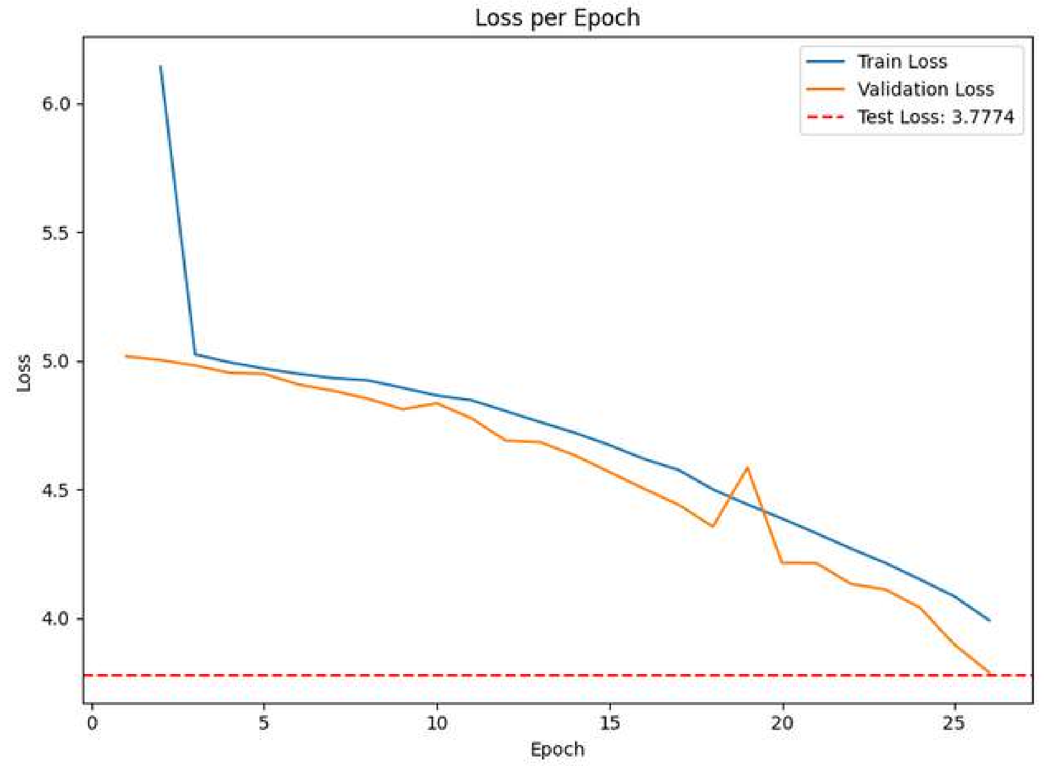
Architecture :

- **4 Convolutional Stages:**
  - **Increasing filters (64 → 128 → 256 → 512)**
  - **Each stage: Two convolutional layers + Batch Normalization + ReLU activation**
  - **Residual connections within each stage for better gradient flow**
  - **MaxPooling after each stage for dimensionality reduction**
- **Residual Blocks: Improve training stability and accuracy by mitigating vanishing gradients.**



Classifier Layers:

- **1024 neurons → ReLU → Dropout (0.4)**
- **512 neurons → ReLU → Dropout (0.4)**
- **Final layer for classification (output size: number of classes)**



Training Setup:

- **Same as the previous**

Results:

- **Test Accuracy: 11.89%**

Observations:

- **there is improvement over baseline performance.**
- **Improved generalization to validation set, but still not good enough.**

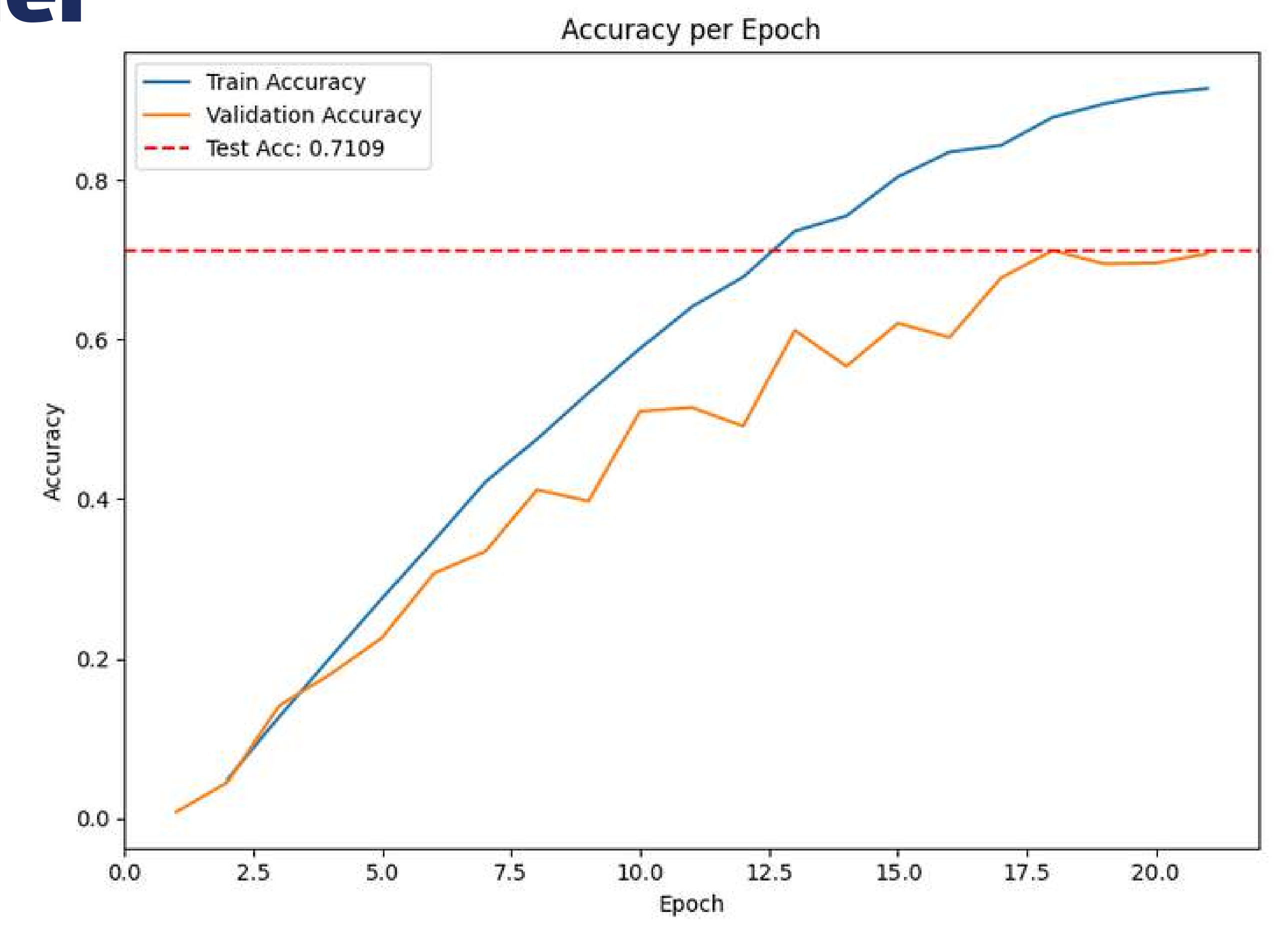

Original Image
Grad-CAM
True: Ninetales, Pred: Cubone

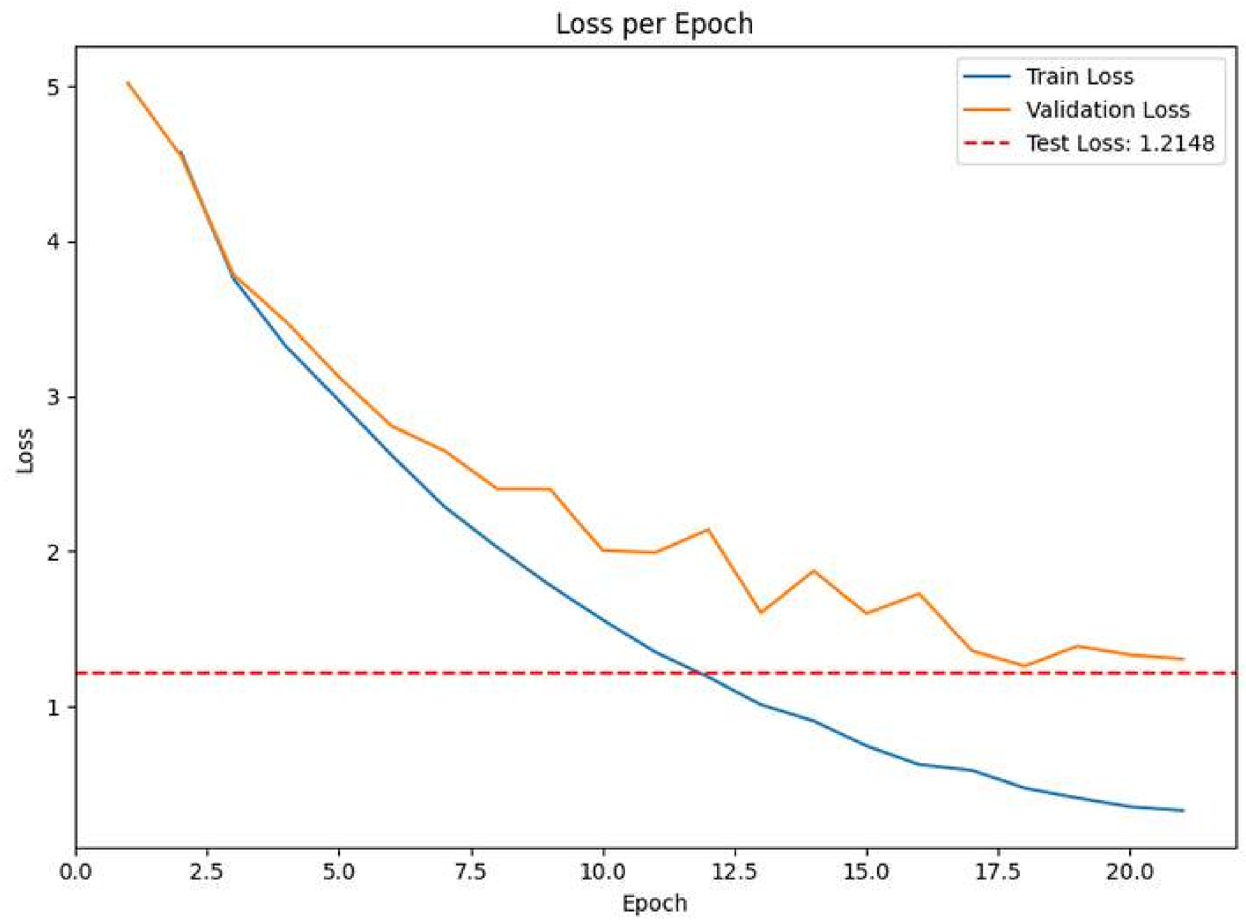

Original Image
Grad-CAM
True: Persian, Pred: Hypno

Model 3 – ConvMixer Model

Architecture:

- ConvMixer (Hybrid CNN model inspired by Vision Transformers):
  - Patch Embedding Layer (Stem):
    - Convolution (patch size=10, stride=10)
    - ReLU activation and Batch Normalization
  - ConvMixer Blocks (Depth = 8):
    - Depthwise convolution (kernel size=5) with residual connection
    - Pointwise convolution (kernel size=1)
    - Batch Normalization and ReLU activations
  - Global Average Pooling and Linear Classification Layer





Training Setup:

- Same as the previous

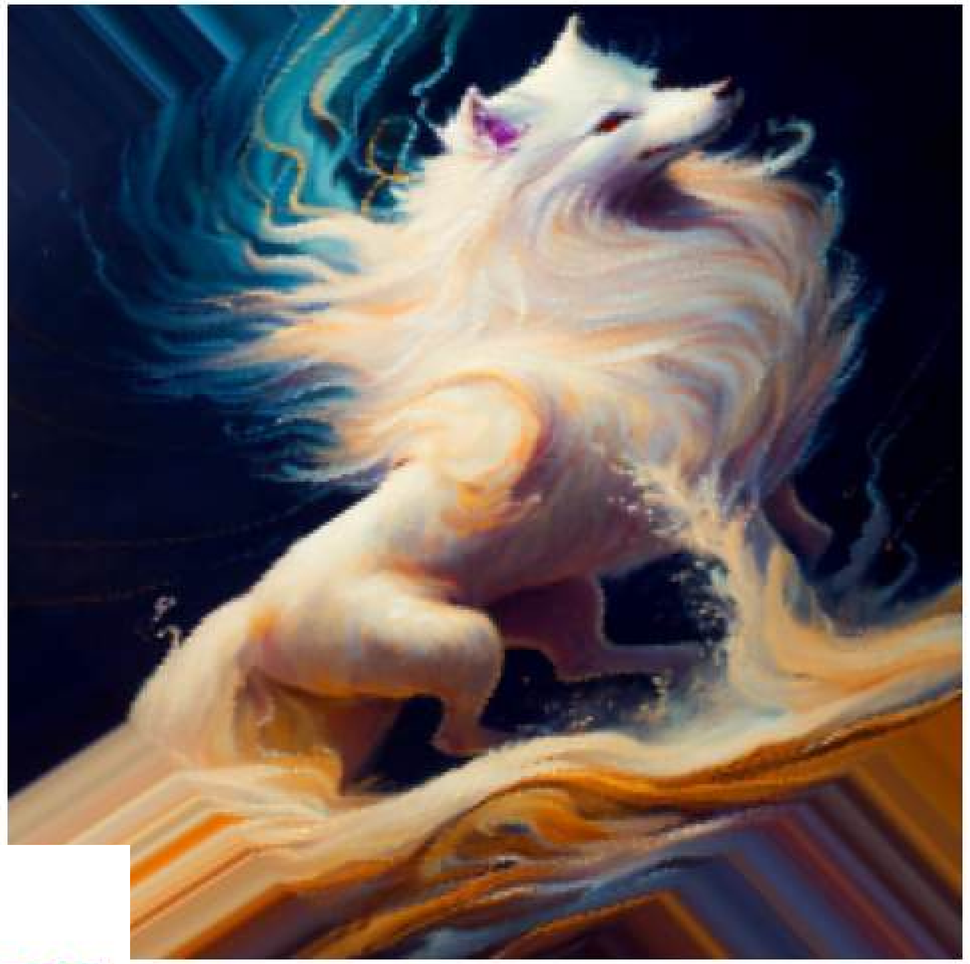
Results:

- Test Accuracy: 71.09%

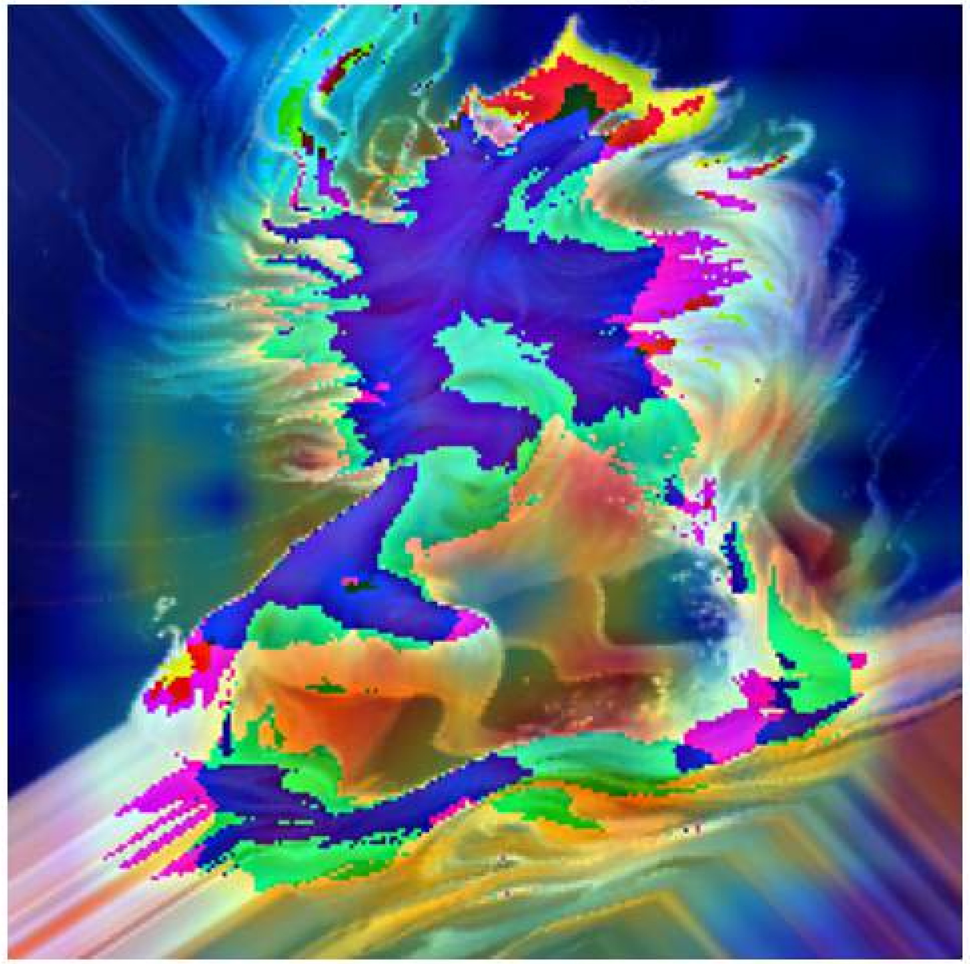
Observations:

demonstrating robust feature extraction and strong generalization.

Original Image



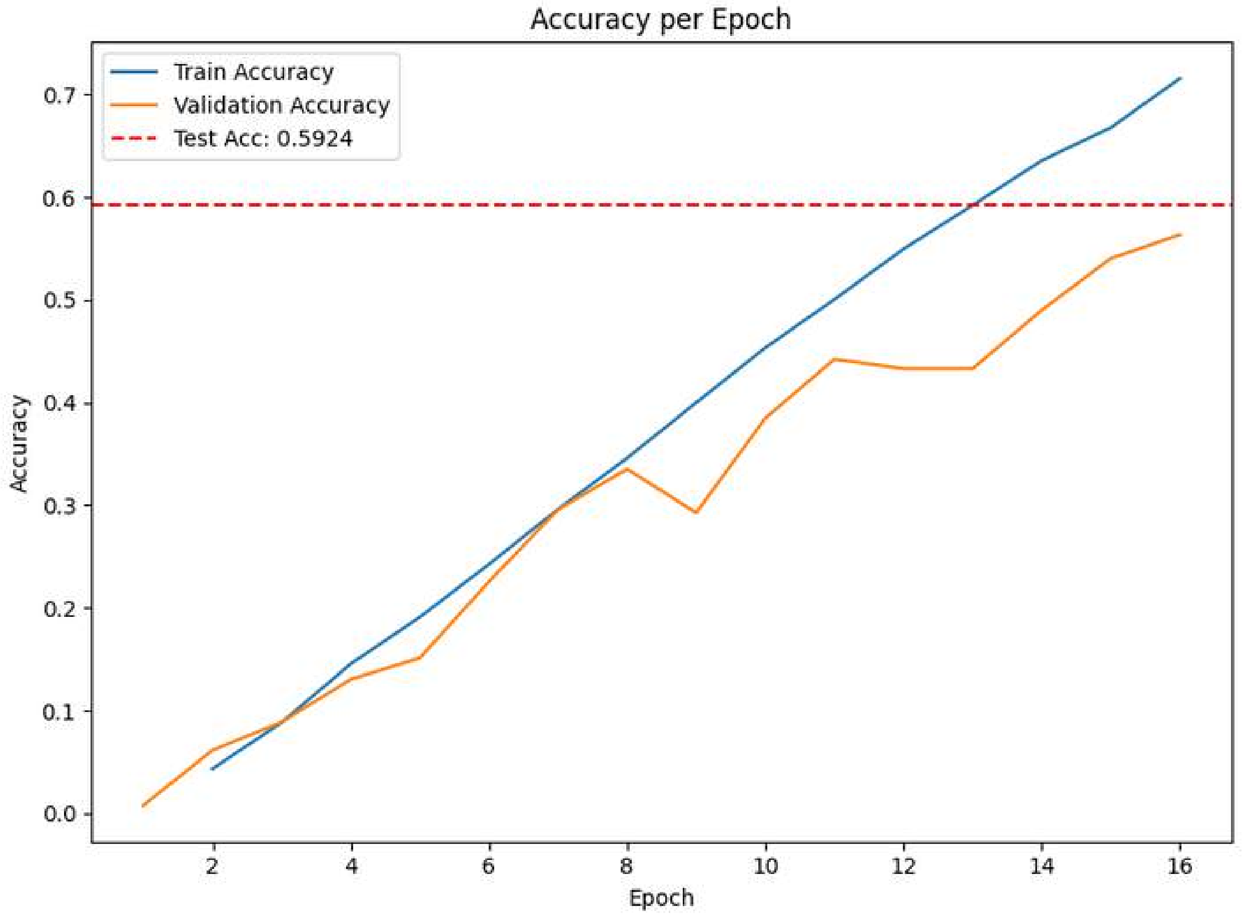
Grad-CAM
True: Ninetales, Pred: Magneton



Original Image

Grad-CAM
True: Persian, Pred: Persian

Model 4 – ResNet18 From Scratch



Architecture:

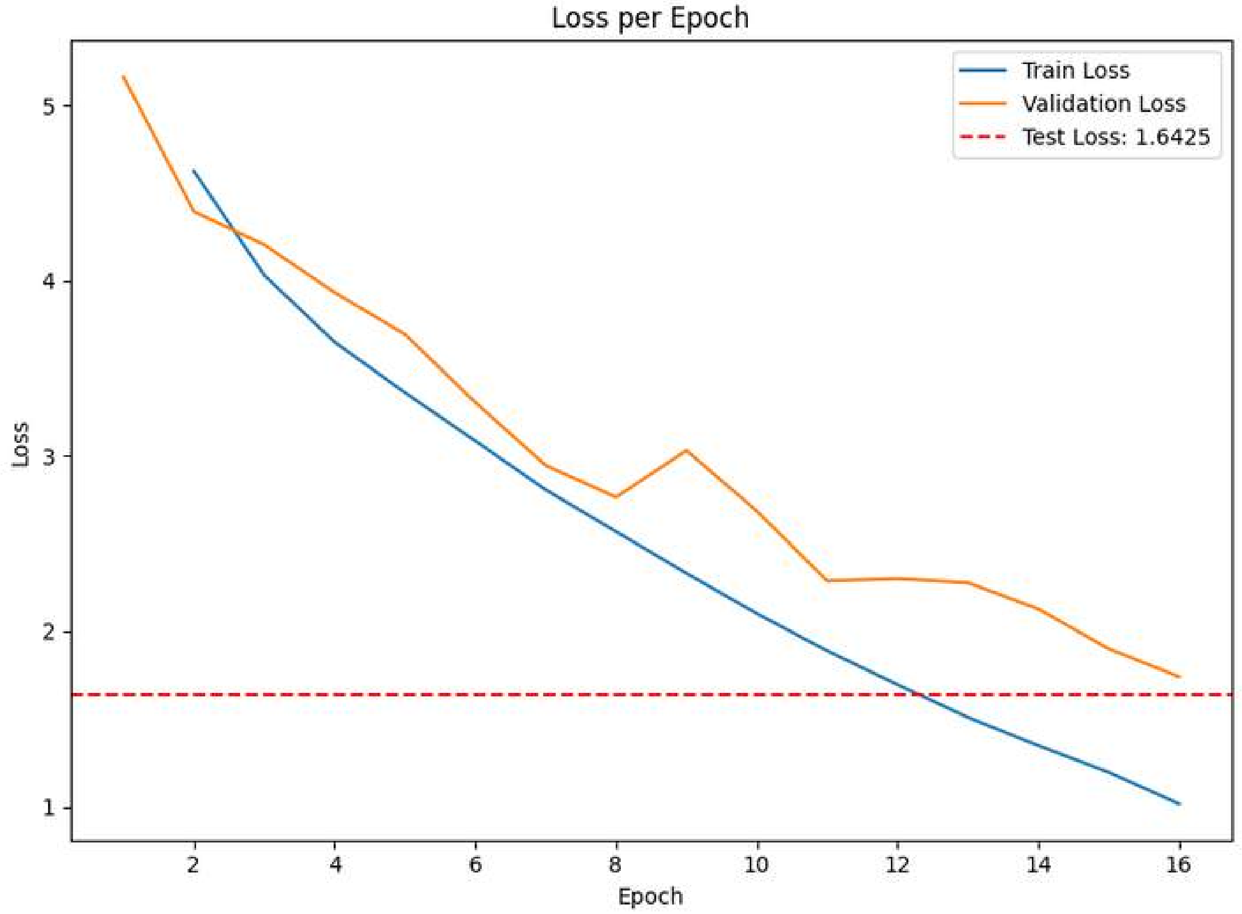
- Base Model: ResNet-18 without pretrained weights
- Modification: Final fully-connected layer replaced to output num_classes (151)

Training Setup:

- Same as the previous

Results:

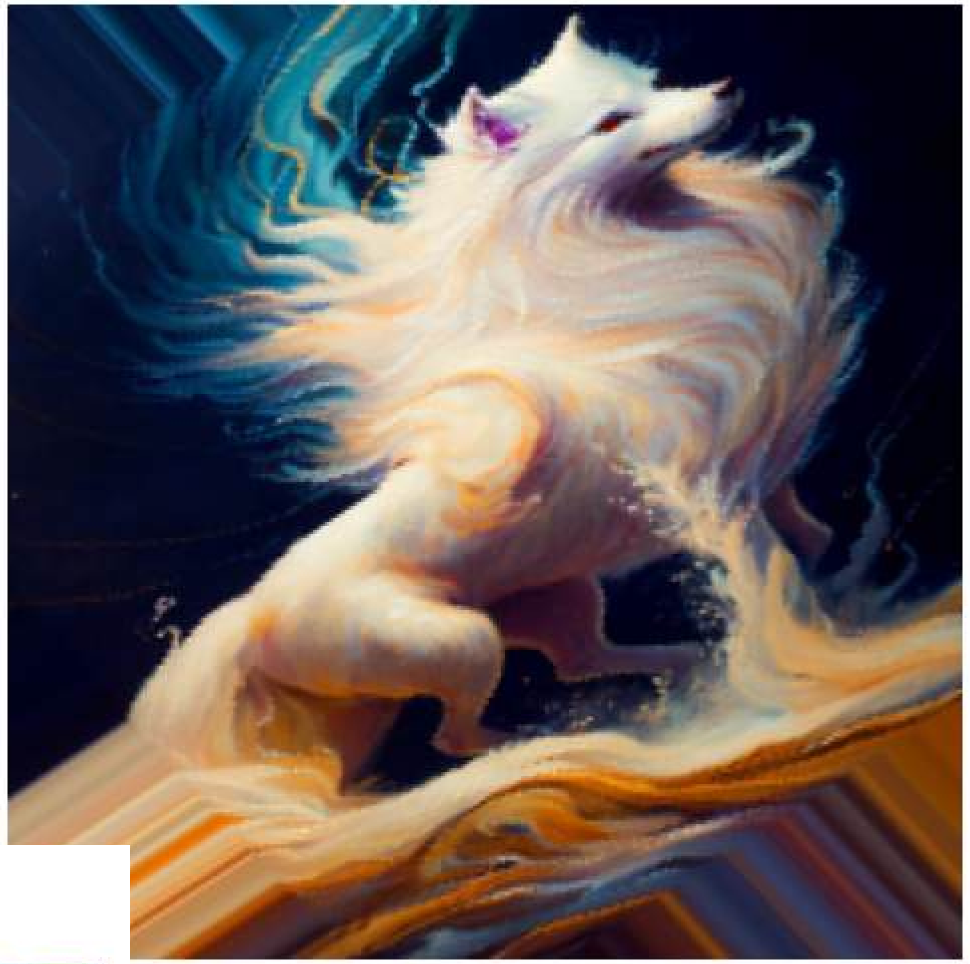
Test Accuracy: 59.24 %



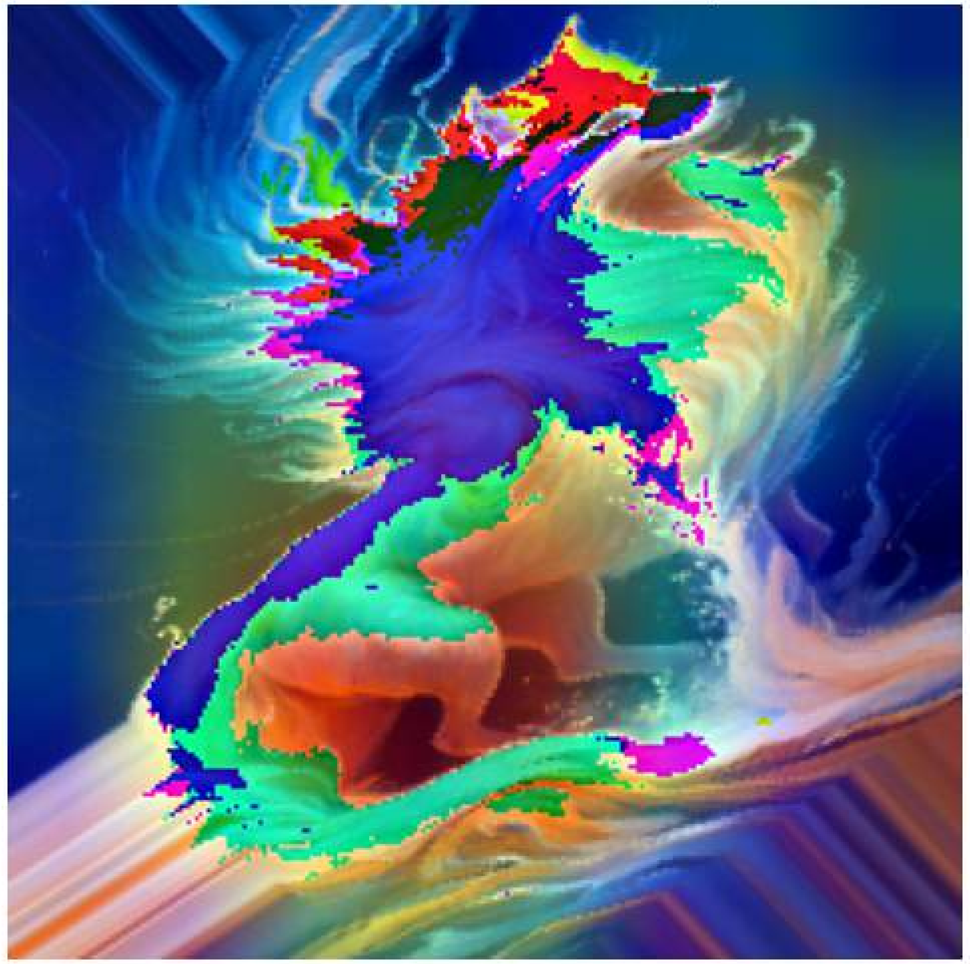
Observations:

This model showing that while deep residual architectures are powerful, training without pretrained weights limits peak performance.

Original Image



Grad-CAM
True: Ninetales, Pred: Primeape



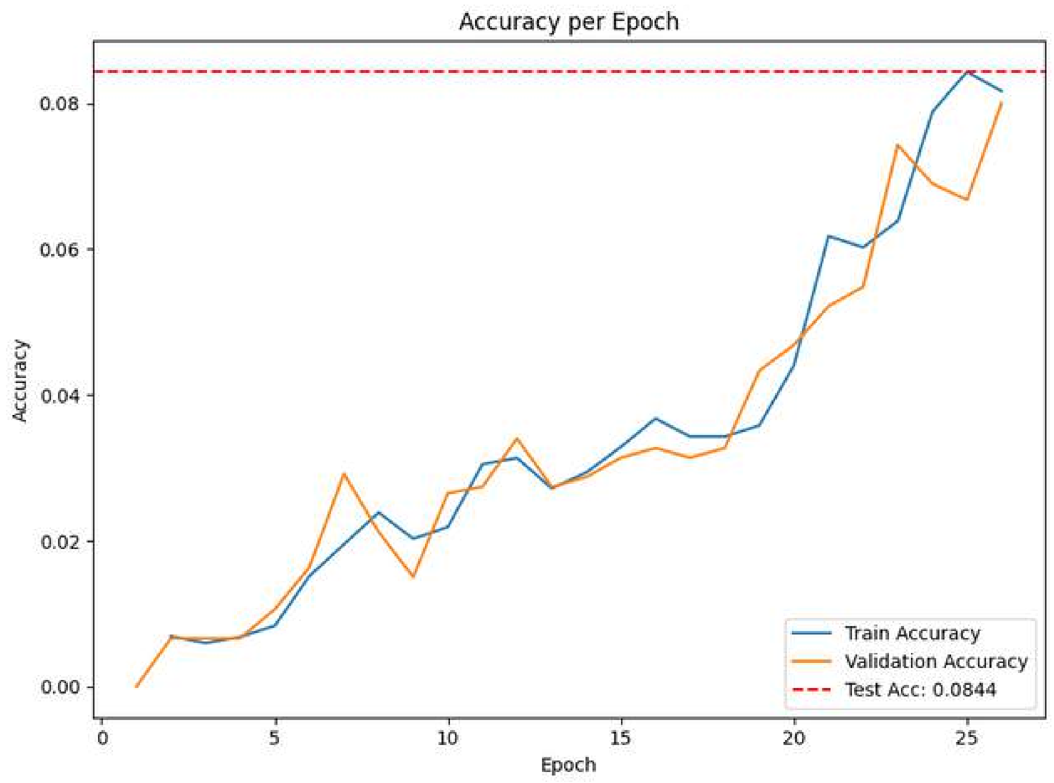
Original Image

Grad-CAM
True: Persian, Pred: Persian

Model 5 – Vision Transformer (ViT) From Scratch

Architecture:

- Base Model: vit_b_16 without pretrained weights
- Modification: Replaced classification head with Linear(in_features → num_classes)

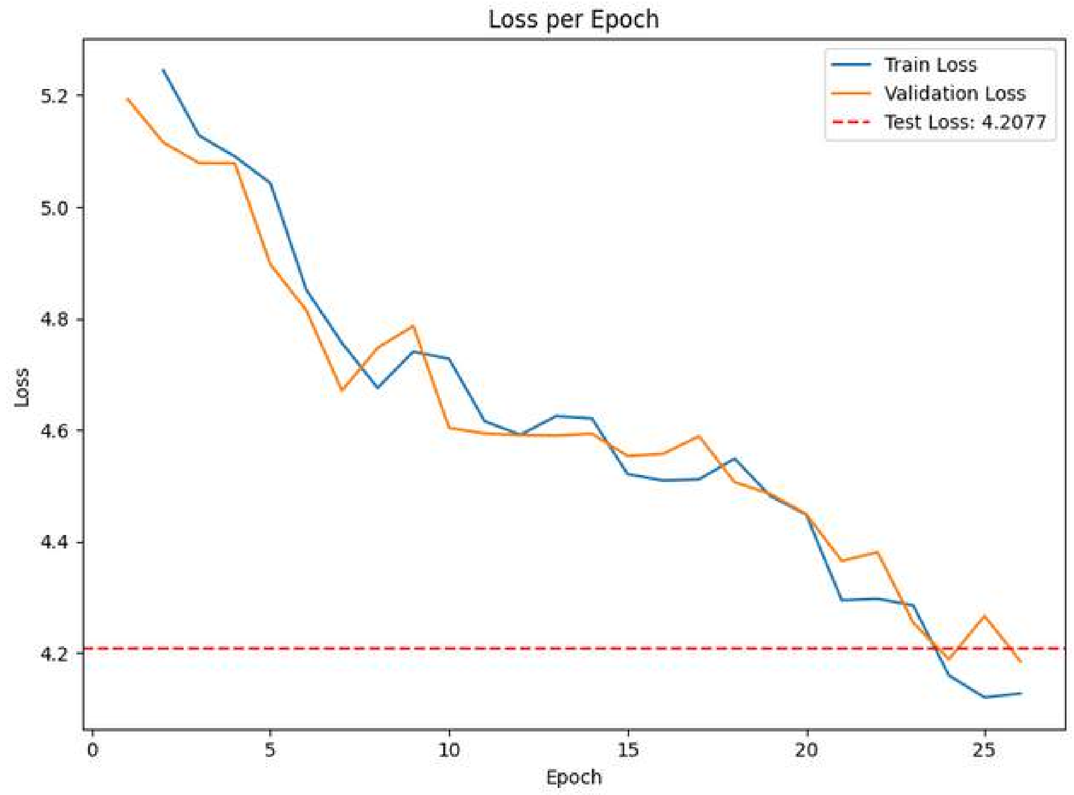


Training Setup:

- Same as the previous

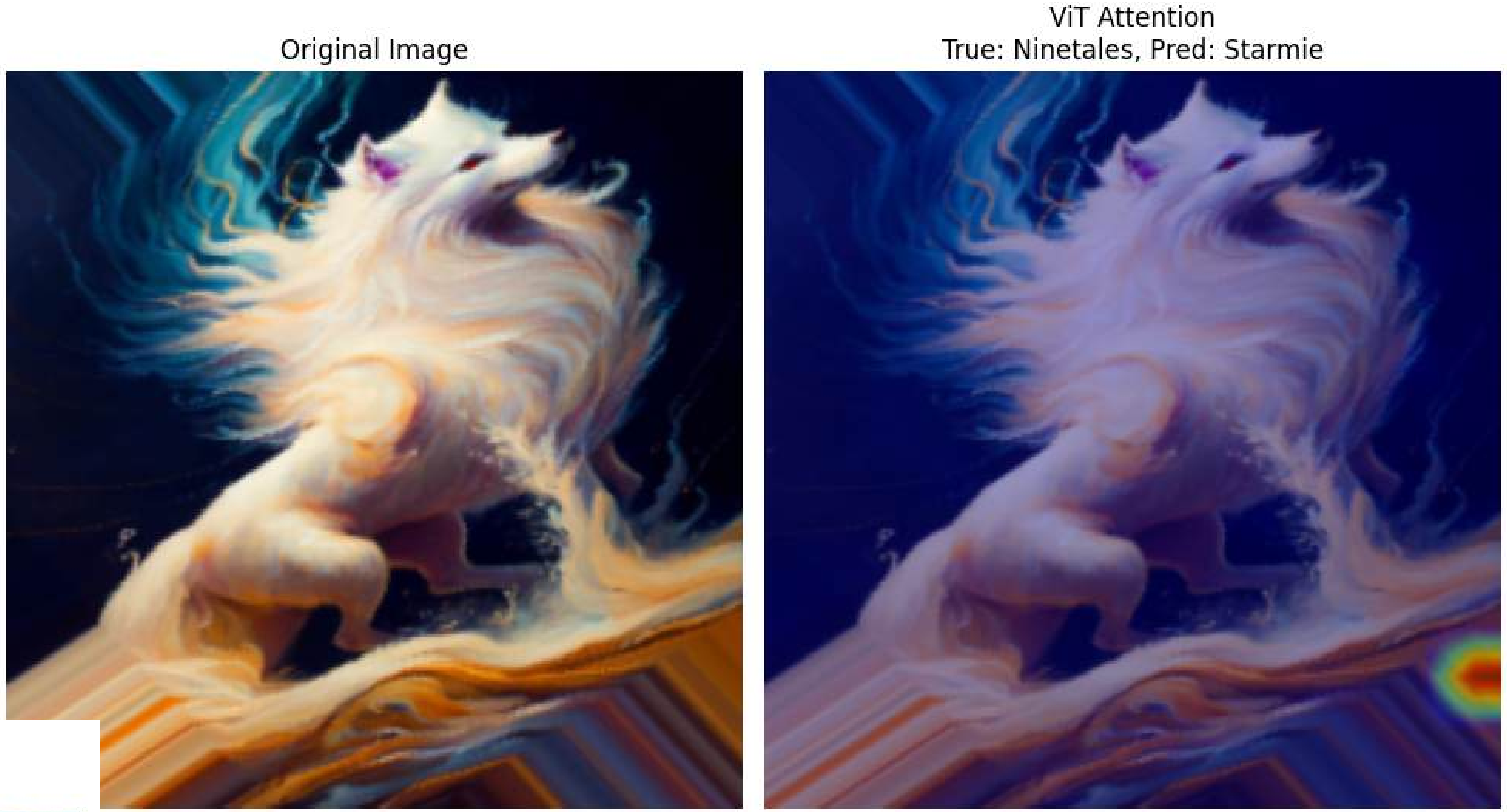
Results:

Test Accuracy: 8.44 %



Observations:

This model indicating that pure transformers require large-scale pretraining or extensive data augmentation to perform well on smaller, specialized datasets.


Original Image
ViT Attention
True: Ninetales, Pred: Starmie


Original Image
ViT Attention
True: Persian, Pred: Oddish

Conclusion

Since the two models trained from scratch did not produce satisfactory results, we will now move on to fine-tuning.

Experiment 1

In this experiment we will pretrained features and stages unfreezing for each model

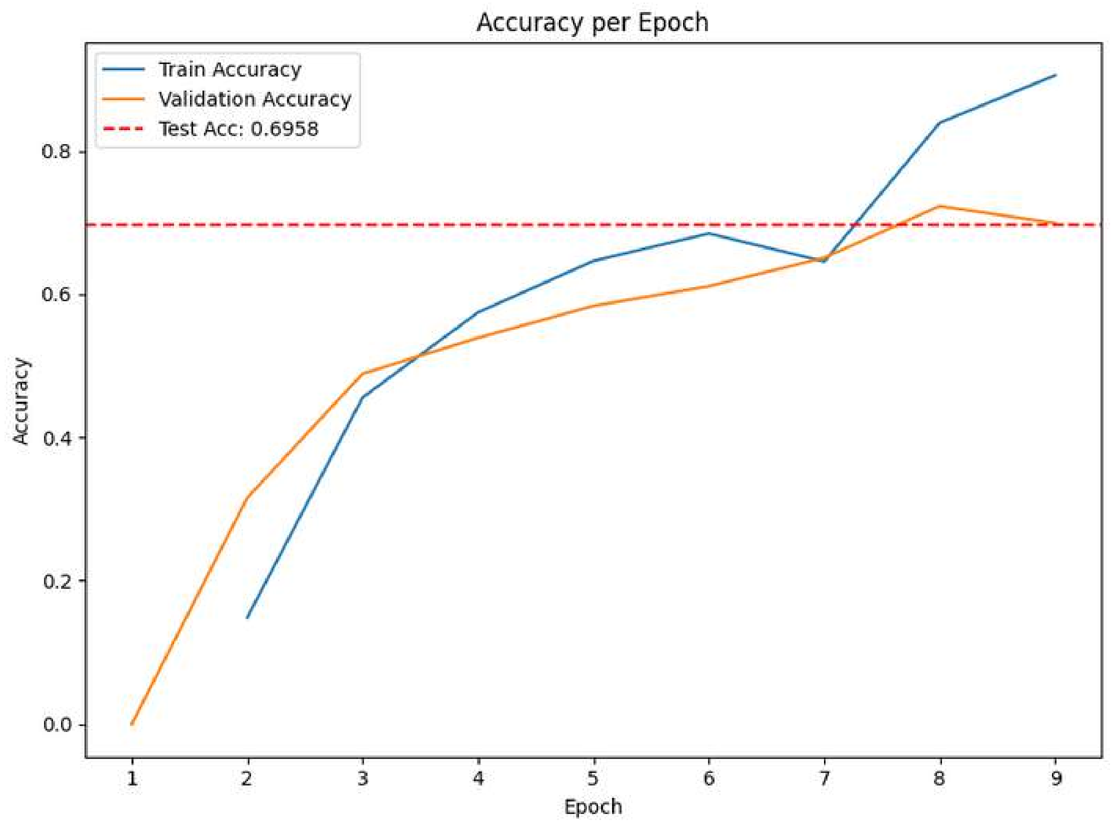
1. **ResNet18 Fine-Tuned**
2. **Pretrained ViT Staged Fine-Tune**

Models 6 – ResNet18 and Pretrained ViT Fine-Tuned

Architecture:

- Base Model: ResNet-18/ViT with ImageNet pretrained weights
- Final Layer: Replaced with Linear(in_features → 151)
- Fine-Tuning Schedule:
  - Epochs 0–3: Freeze backbone, train only final FC layer
  - Epochs ⩾4: Unfreeze entire network for full fine-tuning

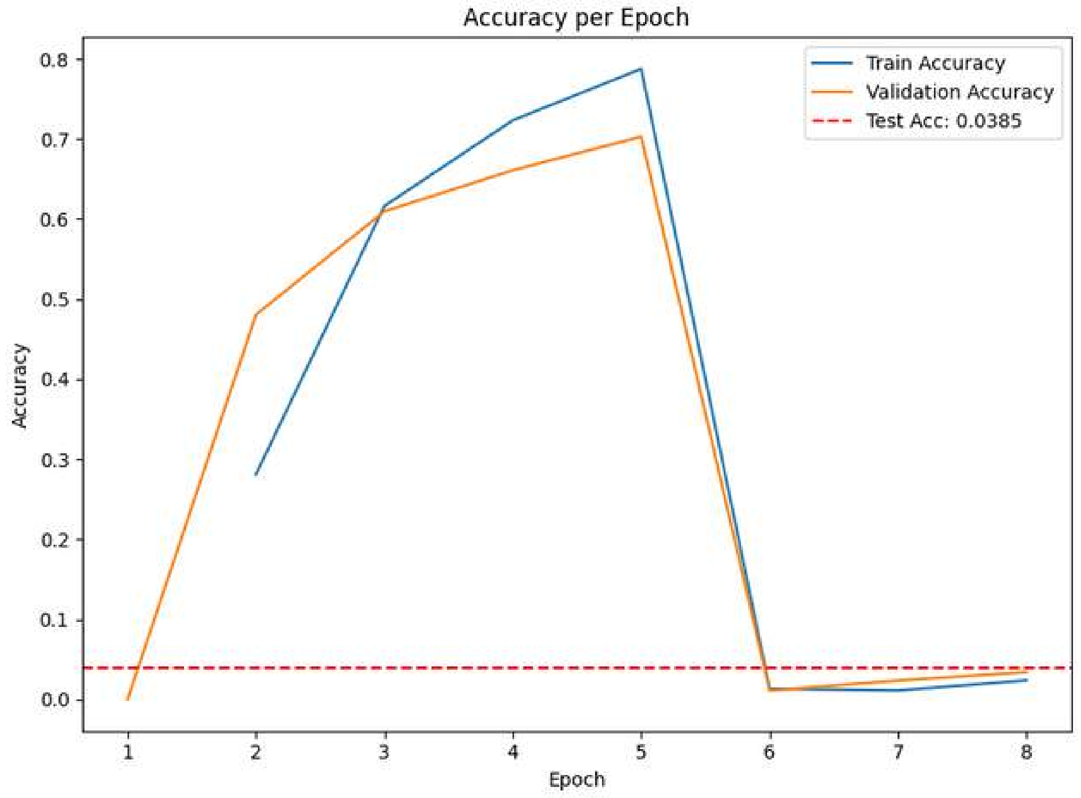
ResNet18 Accuracy



Training Setup:

- Same as the previous

ViT Accuracy



Results:

ResNet18 Test Accuracy: 69.58 %

ViT Test Accuracy: 3.85 %

Observations:

The ResNet18 demonstrating the power of pretrained features and staged unfreezing for effective domain adaptation.

The ViT struggled on this dataset, highlighting the need for large-scale or domain-specific pretraining for pure transformer models.

ResNet18 From Scratch


Original Image

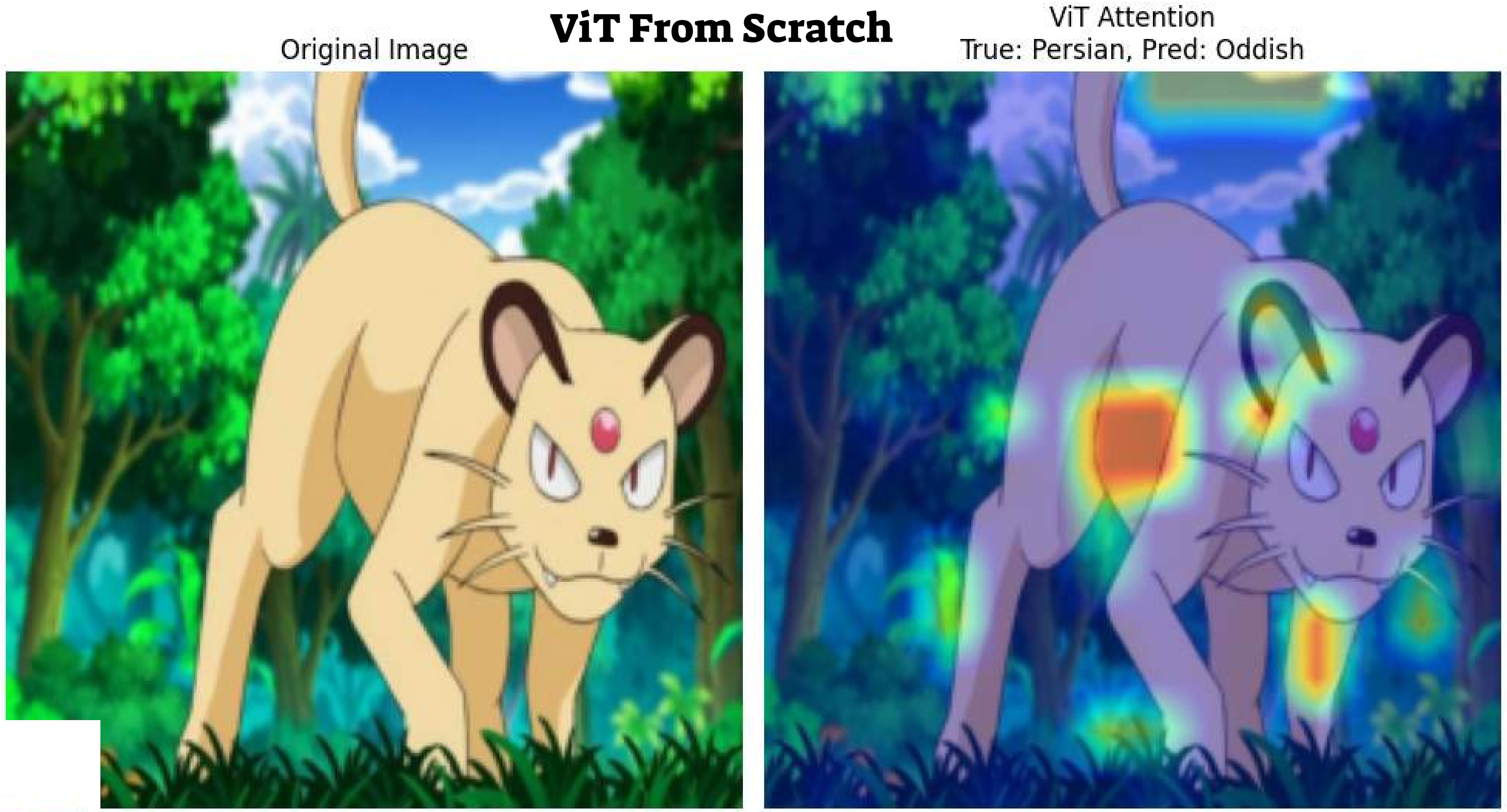

Grad-CAM
True: Persian, Pred: Persian

ResNet18 Fine-Tuned


Original Image


Grad-CAM
True: Persian, Pred: Persian

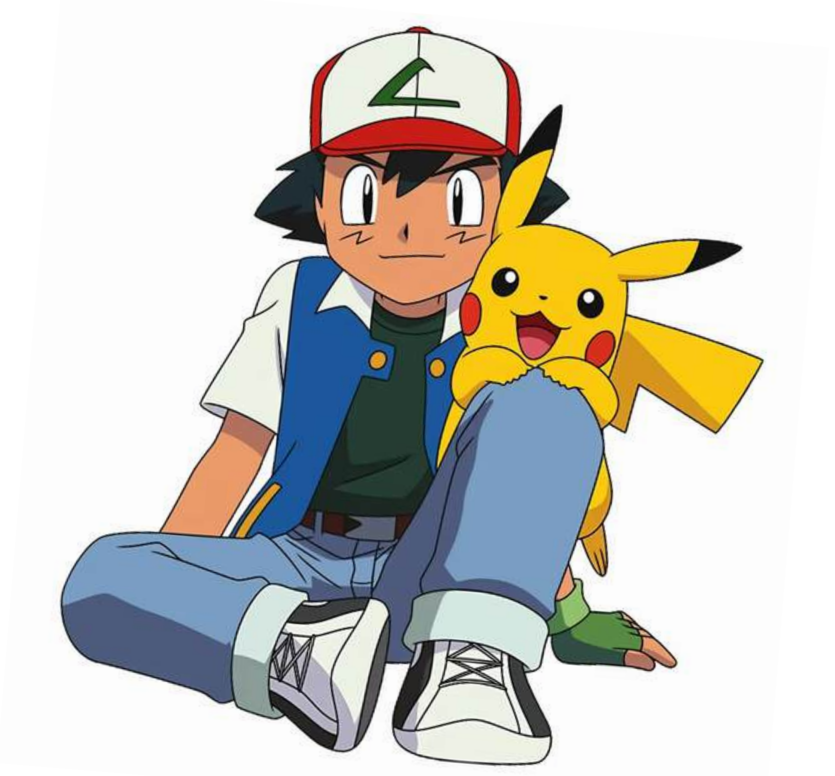
ViT From Scratch



ViT Fine-Tuned

Feature Extraction Pipeline with Pretrained ResNet-18

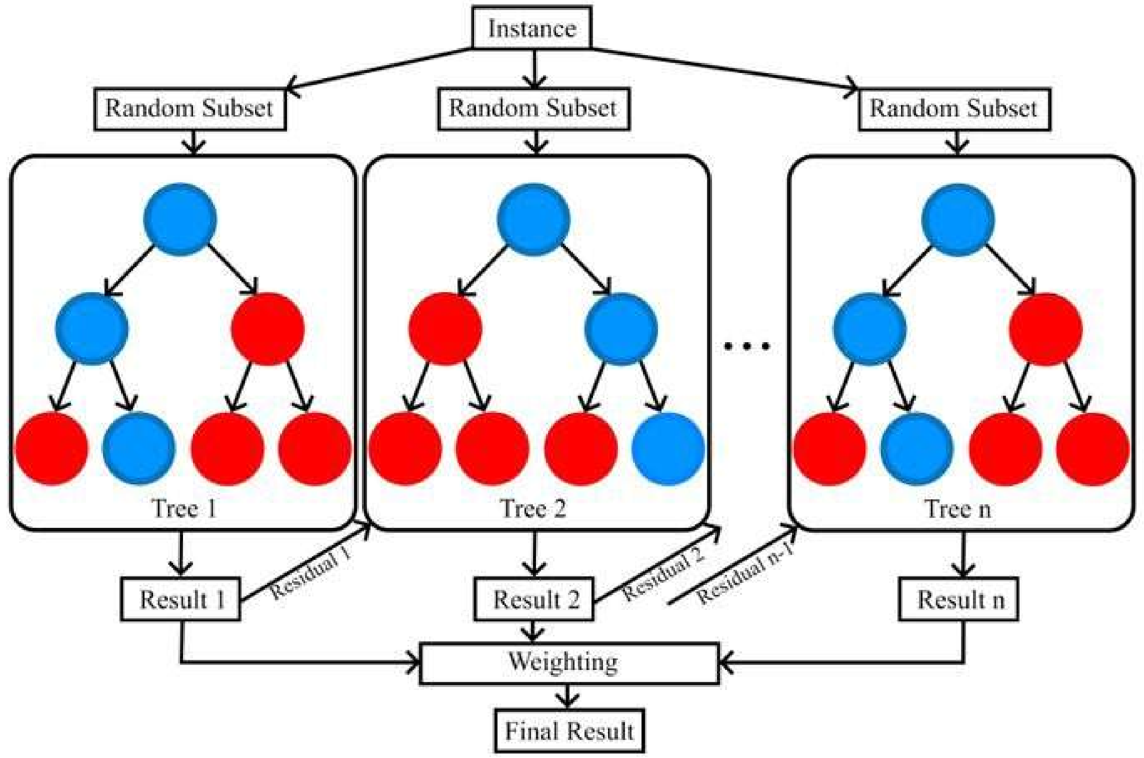
- **Backbone: ResNet-18 pretrained on ImageNet, and fine-tuned on our data.**
- **Output Features: 512-dim vector per image via global avg-pool**
- **Purpose: This feature-extraction step uses a pretrained ResNet-18 to transform each image into a 512-dimensional embedding that encapsulates its high-level visual characteristics. By decoupling expensive CNN passes from classifier training, it lets us quickly experiment with and train classical ML models on fixed-size inputs, leveraging powerful pretrained representations for better performance on our specialized dataset.**



Model 7 – XGBoost on ResNet Features

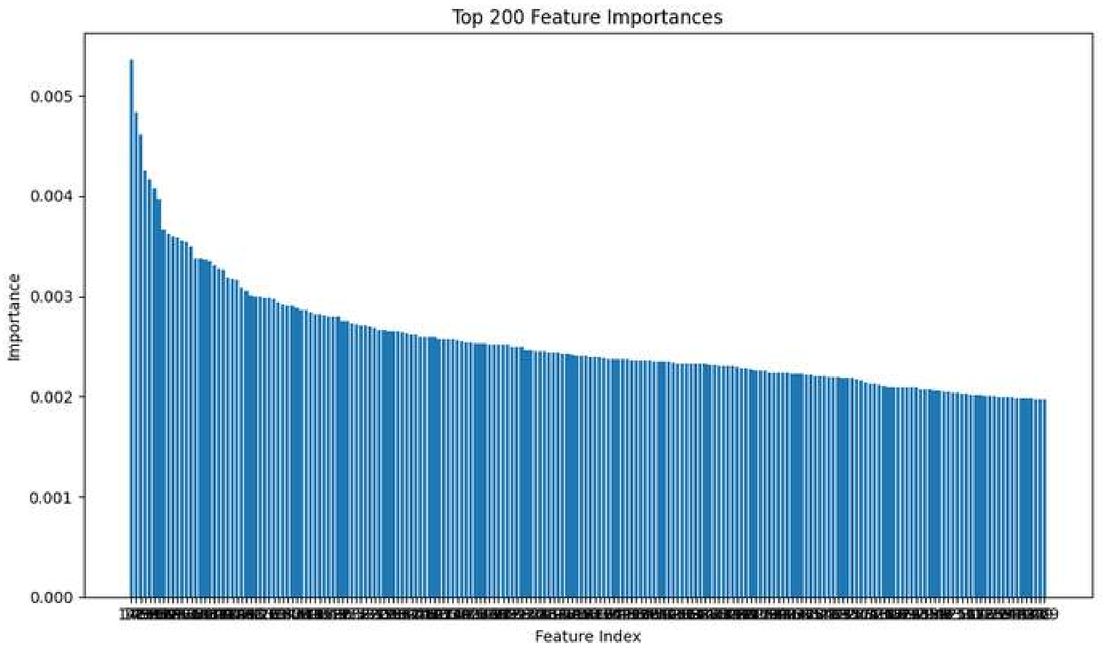
Architecture:

- Feature Input: 512-dim ResNet embeddings
- Classifier: XGBClassifier
- Configurations Tried:
  - n_estimators=100, max_depth=10
  - n_estimators=200, max_depth=16
  - n_estimators=300, max_depth=24



Results: Test Accuracy:

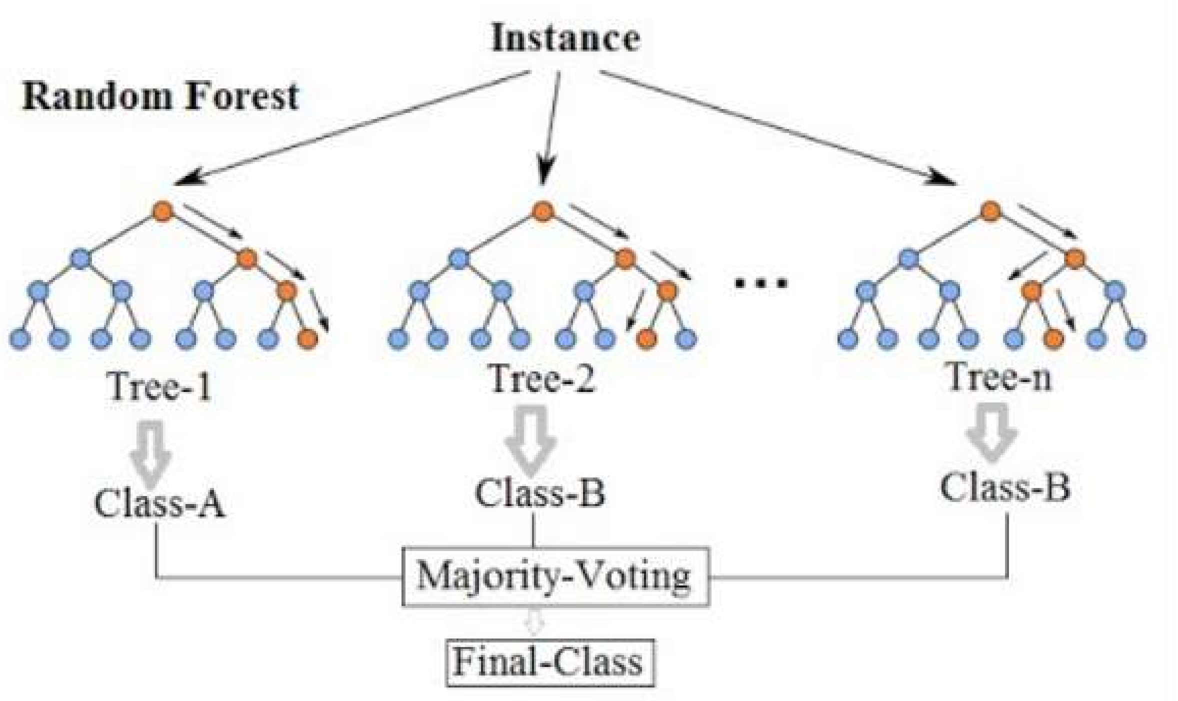
- n=100, d=10 : 41%
- n=200, d=16 : 42%
- n=300, d=24 : 42%



Observations:

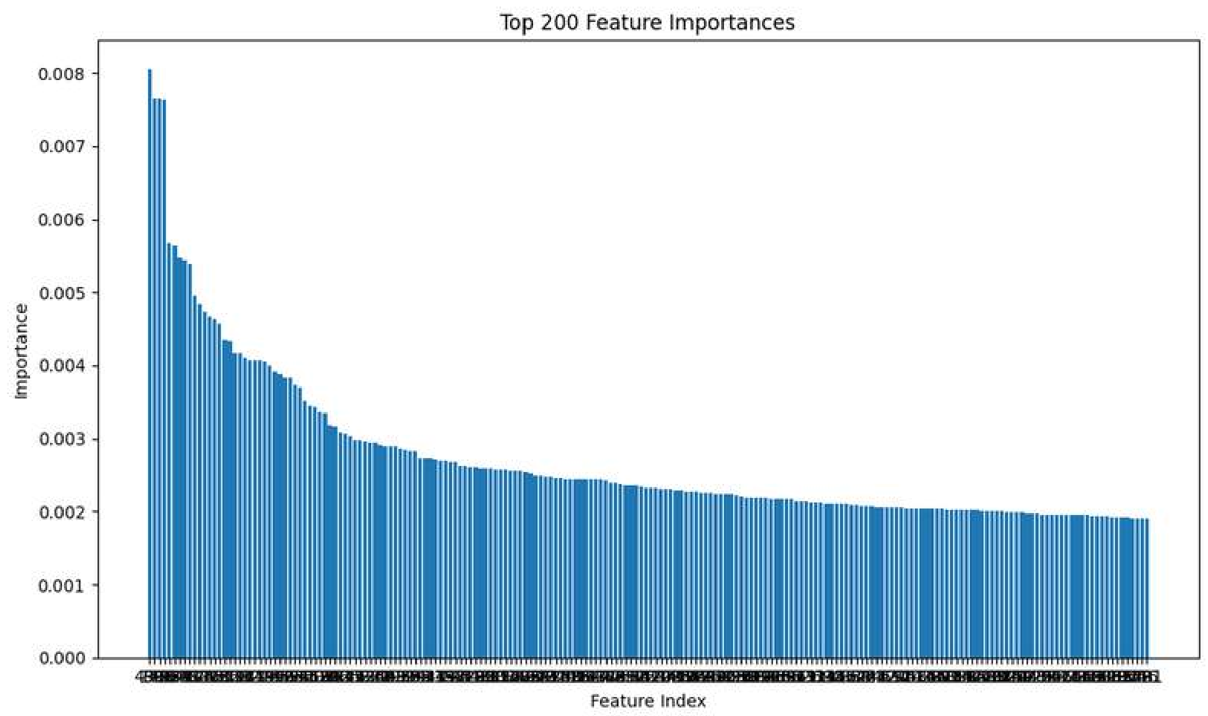
- Achieved ~42% accuracy across deeper configurations, indicating diminishing returns beyond moderate model complexity.
- Captures non-linear interactions in ResNet features but struggles with fine-grained distinctions in 151 classes.

Model 8 – Random Forest on ResNet Features



Architecture:

- Feature Input: 512-dim ResNet embeddings
- Classifier: RandomForestClassifier
- Configurations Tried:
  - n_estimators=100, max_depth=10
  - n_estimators=200, max_depth=16
  - n_estimators=300, max_depth=24



Results: Test Accuracy:

- n=100, d=10 : 25%
- n=200, d=16 : 36%
- n=300, d=24 : 43%

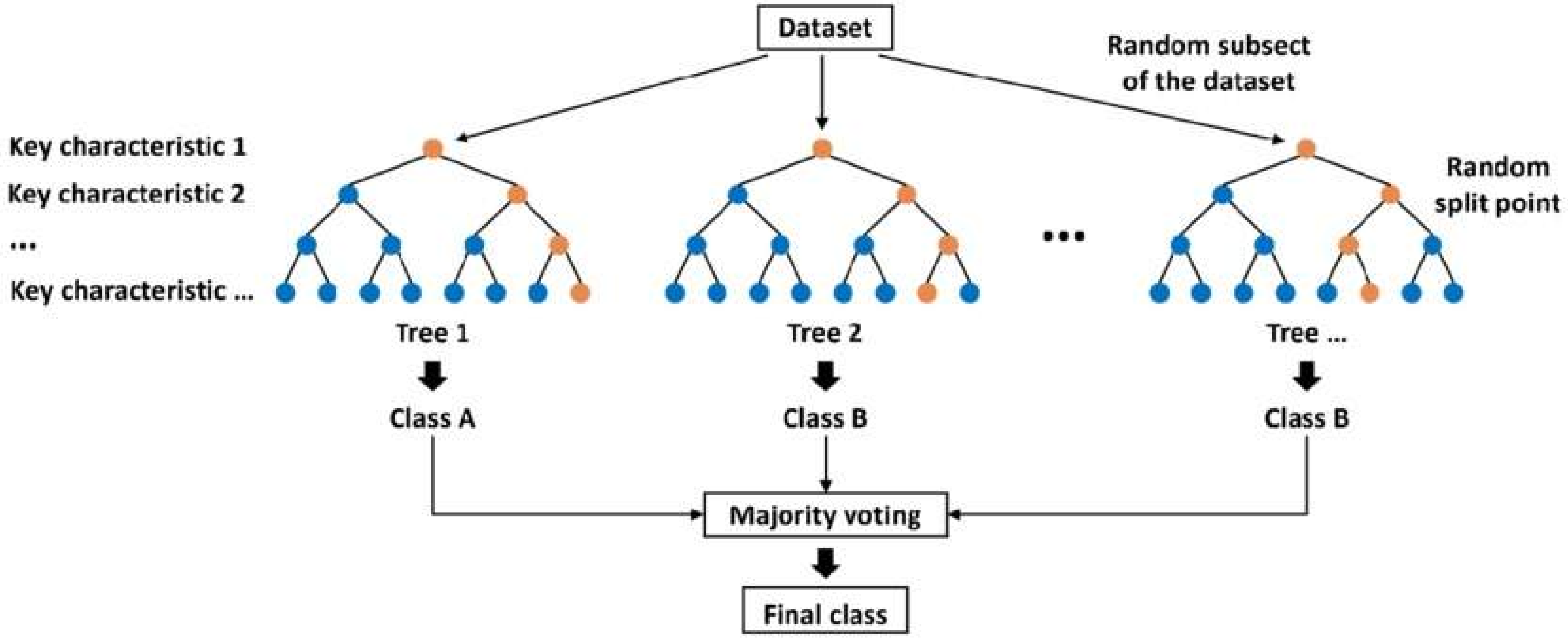
Observations:

- Performance improved steadily with depth and more trees, peaking at 43%, showing that ensembling helps but still underfits complex feature patterns.
- Slower gains suggest limited ability to leverage all discriminative signals from high-dimensional embeddings.

Model 9– Extra Trees on ResNet Features

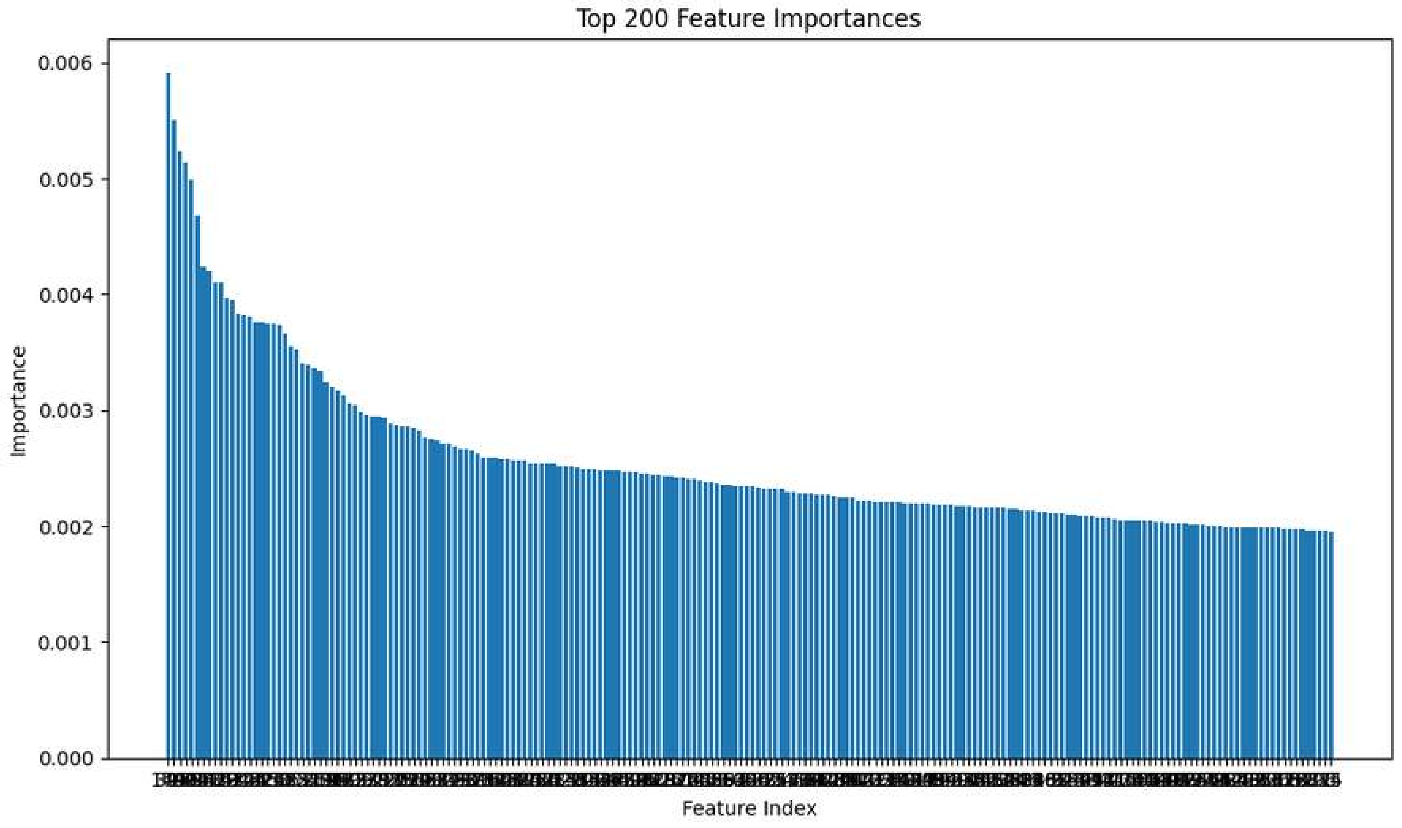
Architecture:

- Feature Input: 512-dim ResNet embeddings
- Classifier: ExtraTreesClassifier
- Configurations Tried:
  - n_estimators=100, max_depth=10
  - n_estimators=200, max_depth=16
  - n_estimators=300, max_depth=24



Results: Test Accuracy:

- n=100, d=10 : 31%
- n=200, d=16 : 43%
- n=300, d=24 : 46%



Observations:

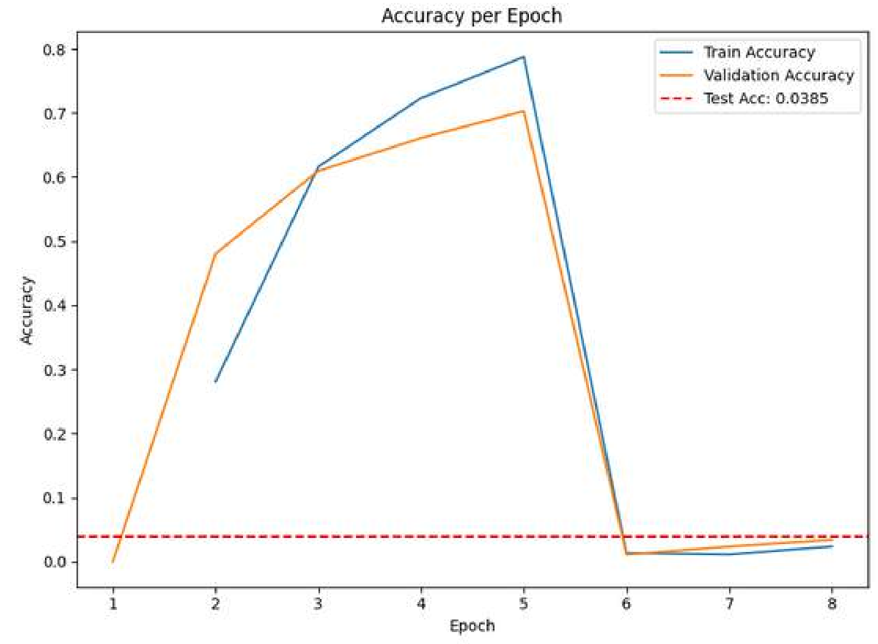
- Significant jump to 46% accuracy at n=300, d=24, demonstrating that extreme randomness plus deep trees can exploit subtle feature variations.
- The aggressiveness of feature and split randomness in Extra Trees appears well-suited to these high-level embeddings and many classes.

Conclusion for experiment 1

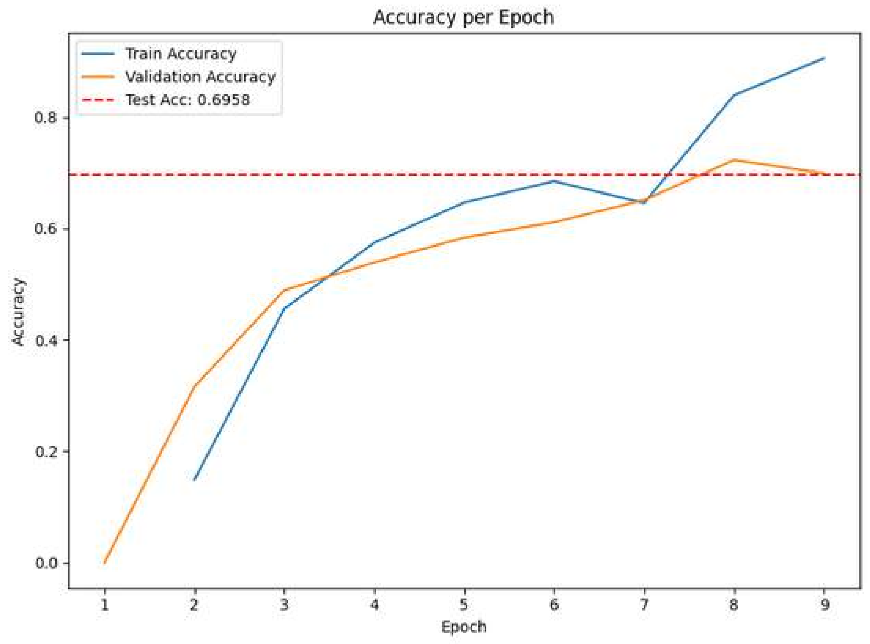
Fine-tuning ResNet-18 clearly yielded strong improvements, whereas fine-tuning the ViT failed to produce satisfactory results and even degraded performance.

Model	Test Accuracy	Num of parm
XGBoost	41.00%	n_estimators=100, max_depth=10
	42.00%	n_estimators=200, max_depth=16
	42.00%	n_estimators=300, max_depth=24
Random Forest	25.00%	n_estimators=100, max_depth=10
	36.00%	n_estimators=200, max_depth=16
	43.00%	n_estimators=300, max_depth=24
Extra Trees	31.00%	n_estimators=100, max_depth=10
	43.00%	n_estimators=200, max_depth=16
	46.00%	n_estimators=300, max_depth=24

ViT



Resnet-18



Experiment 2

In this experiment we will use ResNet-18 with ImageNet pretrained weights without unfreezing depth.

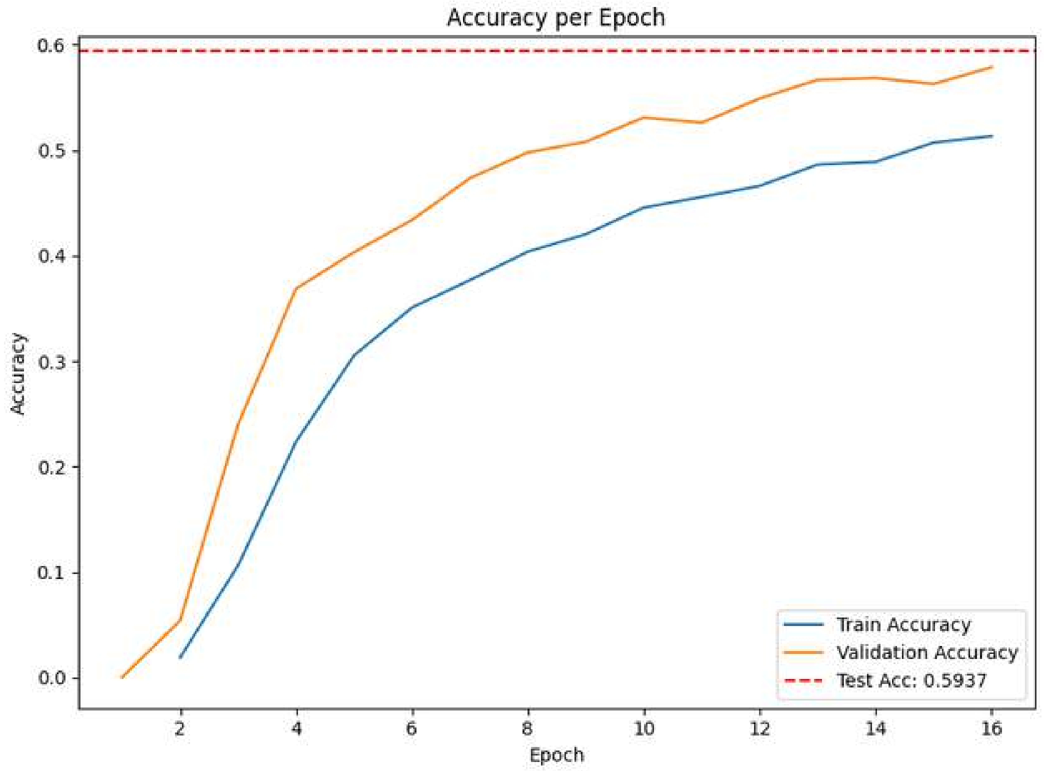
1. **ResNet18 Fine-Tuned with Frozen Deep Head**
2. **Pretrained ViT Staged Fine-Tune with Frozen Deep Head**

Models 10 – ResNet-18 and Vit Fine-Tuned with Frozen Deep Head

Architecture:

- Base Model: ResNet-18/ViT with ImageNet pretrained weights
- Backbone: Fully frozen (all conv layers non-trainable)
- Classifier Head: Deeper, multi-layer head
  - Linear(in_features→512) → ReLU → Dropout(0.4)
  - Linear(512→256) → ReLU → Dropout(0.2)
  - Linear(256→num_classes)

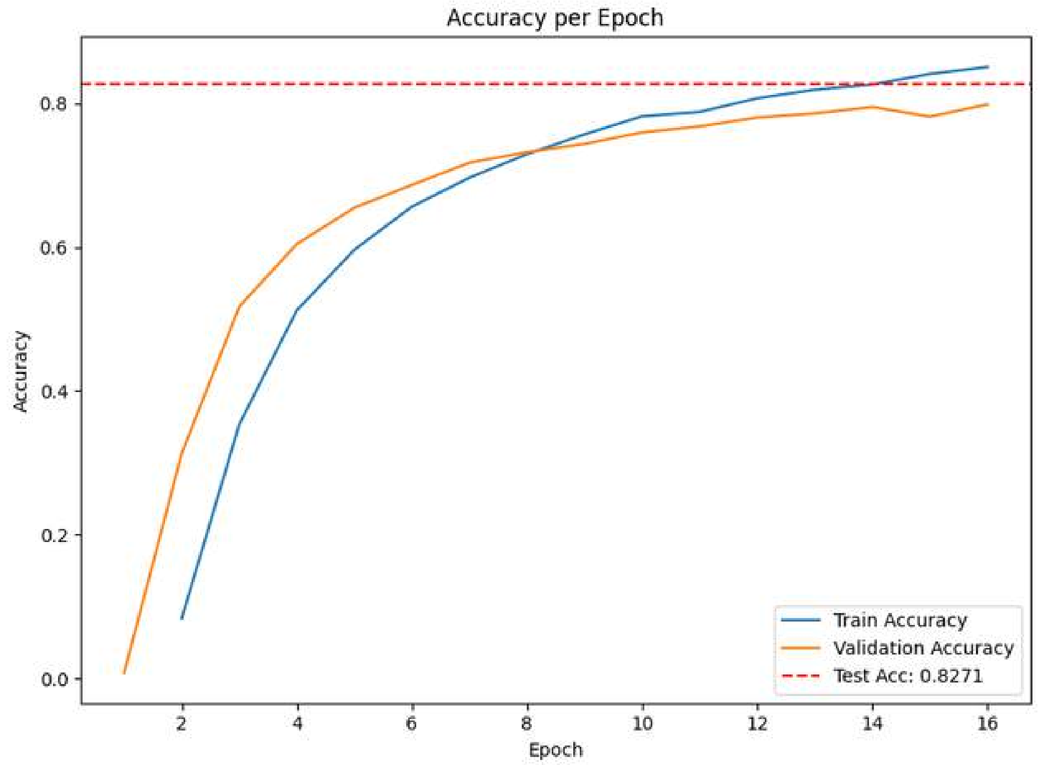
ResNet18 Accuracy



Training Setup:

- Same as the previous

ViT Accuracy



Results:

ResNet-18 Test Accuracy: 59.37 %

ViT Test Accuracy: 82.71 %

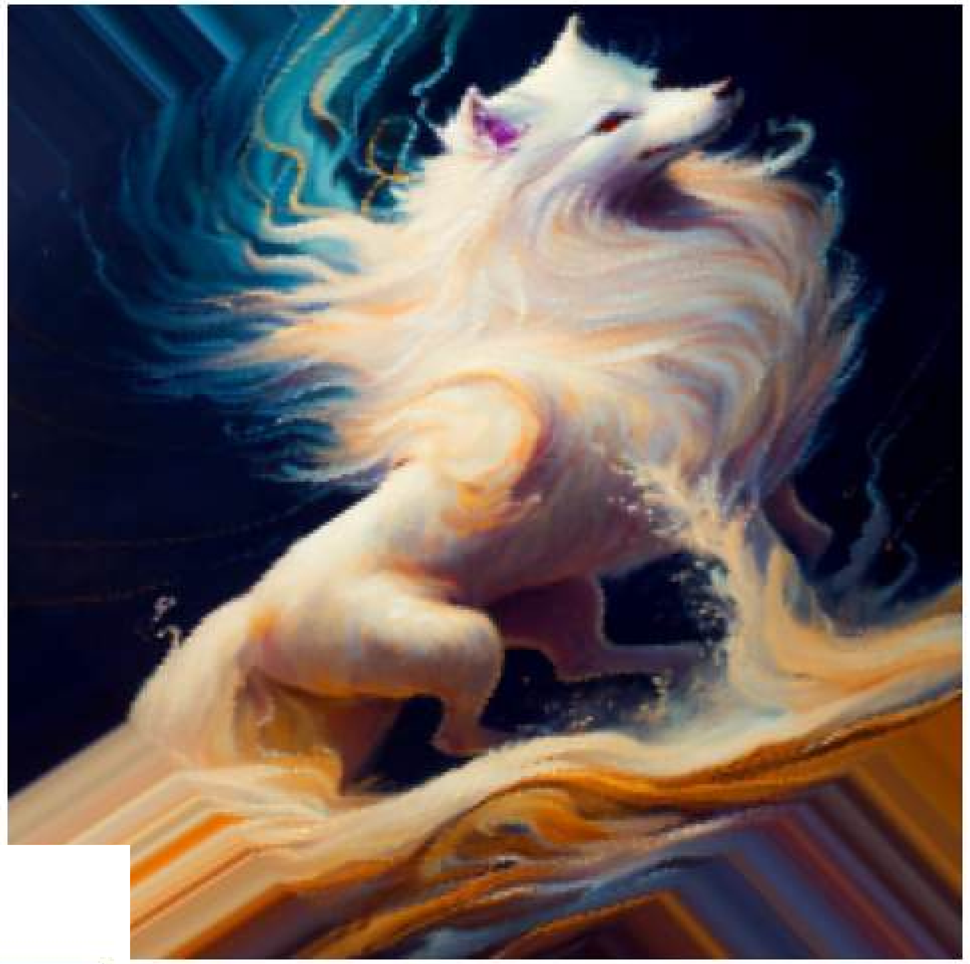
Observations:

ResNet-18 yielding moderate performance.

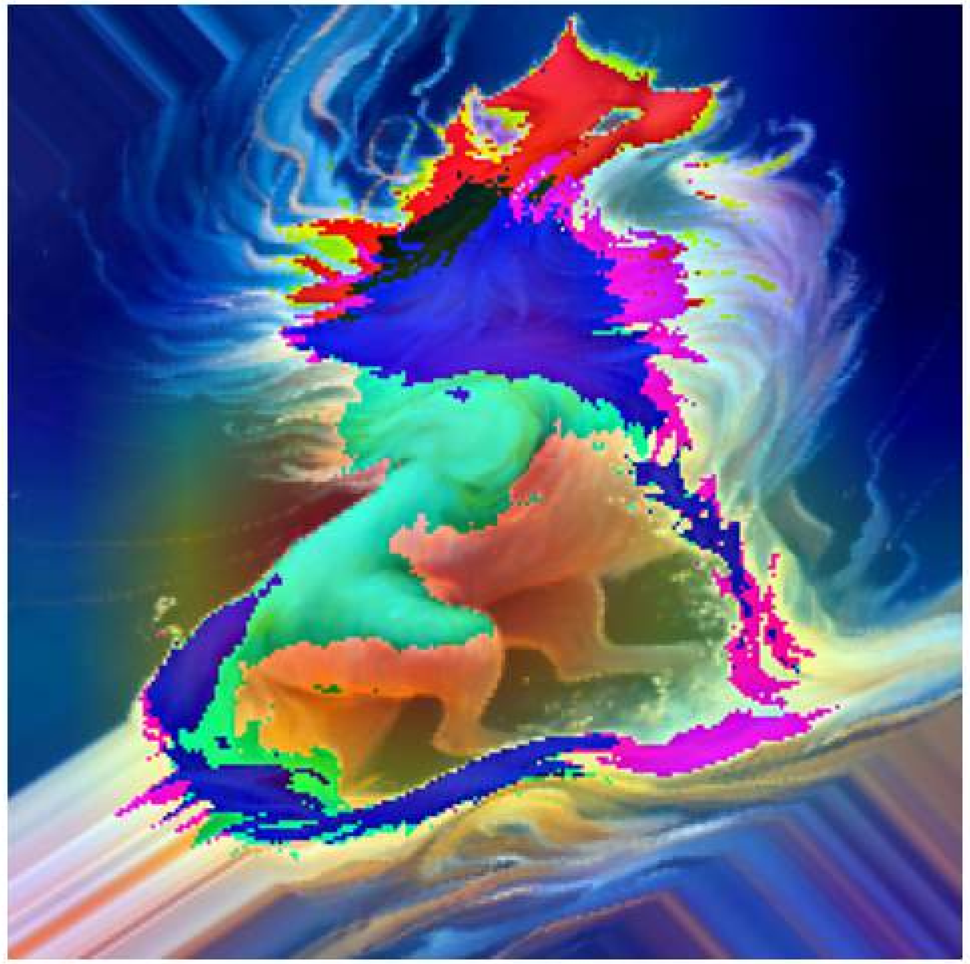
ViT yields excellent performance—showcasing that a powerful pretrained feature extractor can outperform full fine-tuning on limited data.

Experiment 1

Original Image



Grad-CAM
True: Ninetales, Pred: Arcanine



Experiment 2

Original Image

Grad-CAM
True: Ninetales, Pred: Articuno

Experiment 1

Original Image

ViT Attention
True: Persian, Pred: Clefable

Experiment 2

Original Image

ViT Attention
True: Persian, Pred: Persian

New Feature Extraction Pipeline with Fine-Tuned ResNet-18

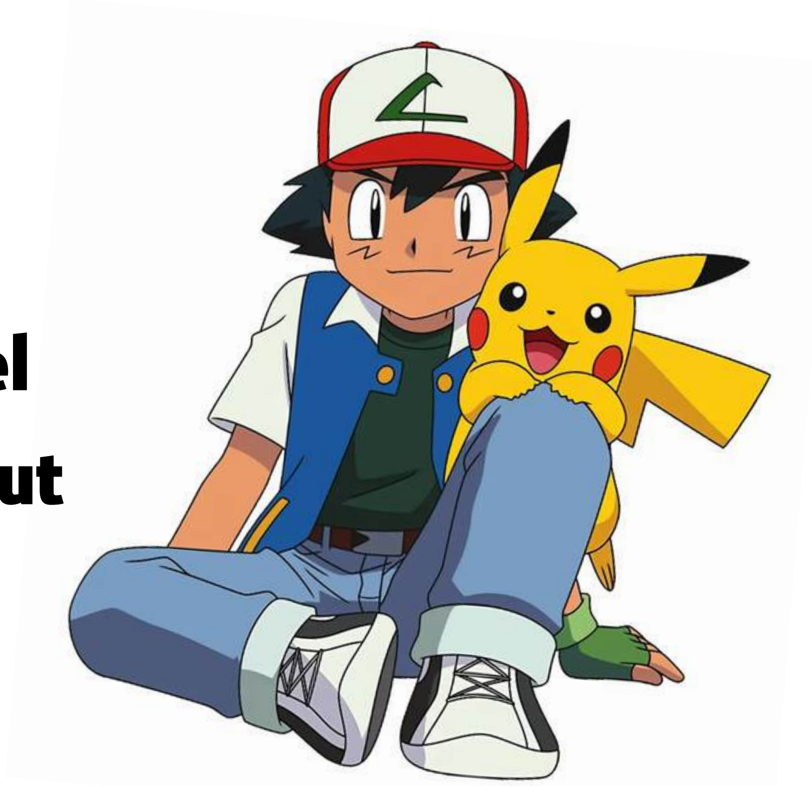
- Base Model: ResNet-18 with pretrained weights and a frozen deep head (backbone weights remained frozen during training)

- Backbone: All convolutional layers up to global pooling

- Custom Head Hook:
    - Captures activations after the 2nd ReLU in the deep FC head
    - Uses a forward hook to save intermediate features

- Output Features: Features collected at relu2 stage for each image

- Purpose: Leverage a fine-tuned classification model to extract task-specific, mid-level embeddings—providing richer, domain-adapted features for downstream tasks without retraining the full network.

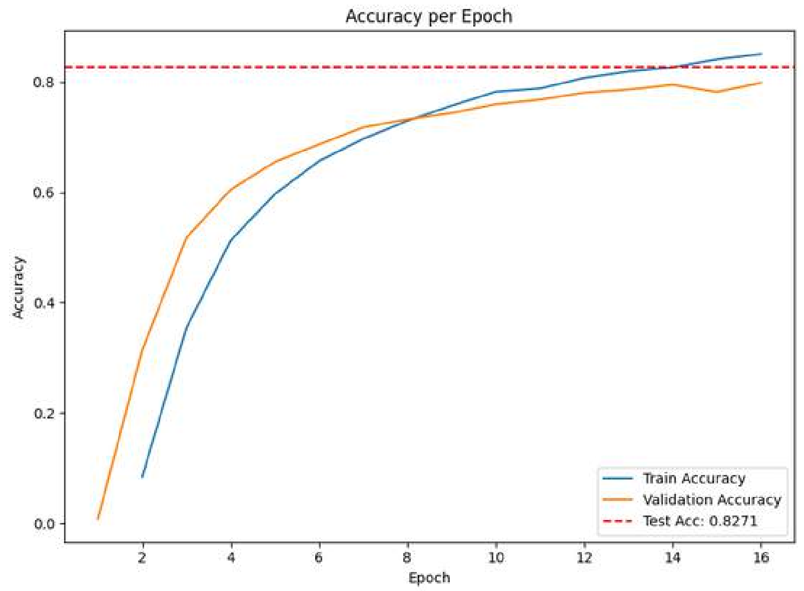


Conclusion for experiment 2

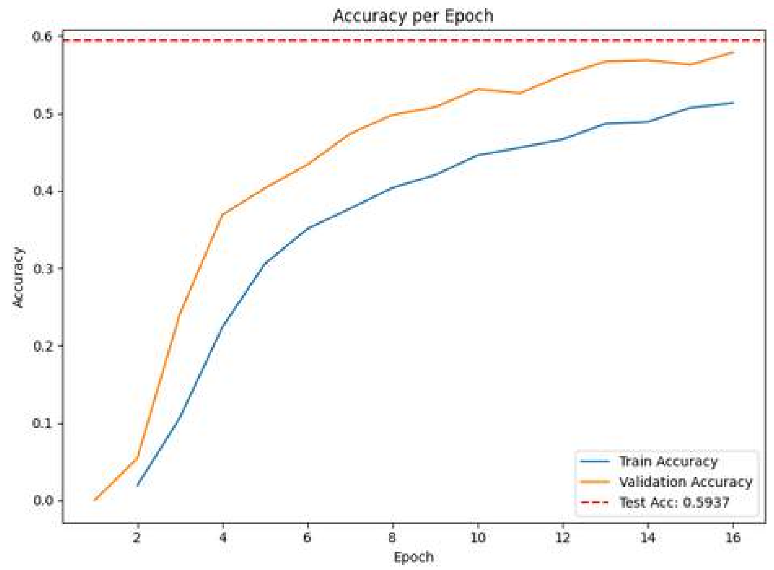
Experiment 2 achieved superior performance in both loss and accuracy, and the tree-based models trained using its settings also outperformed the previous trial, highlighting the deeper significance of its approach.

Model	Accuracy exp1	Accuracy exp2	Num of parm
XGBoost	41.00%	57.00%	n_estimators=100, max_depth=10
	42.00%	57.00%	n_estimators=200, max_depth=16
	42.00%	58.00%	n_estimators=300, max_depth=24
Random Forest	25.00%	55.00%	n_estimators=100, max_depth=10
	36.00%	62.00%	n_estimators=200, max_depth=16
	43.00%	63.00%	n_estimators=300, max_depth=24
Extra Trees	31.00%	56.00%	n_estimators=100, max_depth=10
	43.00%	64.00%	n_estimators=200, max_depth=16
	46.00%	66.00%	n_estimators=300, max_depth=24

ViT



Resnet-18



Summary of model results

Model	Test Accuracy	Num of parm
Basic CNN	0.66%	29.6M
Custom CNN Modern (Residual CNN)	11.89%	129M
ConvMixer Model	71.09%	700K
ResNet18 From Scratch	59.24%	11.3M
Vision Transformer (ViT) From Scratch	8.44%	85.9M
ResNet18 Fine-Tuned (experiment 1)	69.58%	11.3M
Pretrained ViT Staged Fine-Tune (experiment 1)	3.85%	85.9M
XGBoost on ResNet Features (experiment 1)	42.00%	n_estimators=200, max_depth=16
Random Forest on ResNet Features (experiment 1)	43.00%	n_estimators=300, max_depth=24
Extra Trees on ResNet Features (experiment 1)	46.00%	n_estimators=300, max_depth=24
ResNet18 Fine-Tuned with Frozen Deep Head (experiment 2)	59.37%	11.6M
Pretrained ViT Staged Fine-Tune (experiment 2)	82.71%	86.4M
XGBoost on Fine-Tuned ResNet Head Features (experiment 2)	58.00%	n_estimators=300 , max_depth=24
Random Forest on Fine-Tuned ResNet Head Features (experiment 2)	63.00%	n_estimators=300 , max_depth=24
Extra Trees on Fine-Tuned ResNet Head Features (experiment 2)	66.00%	n_estimators=300 , max_depth=24

Summary

This project benchmarked 15 models across 151 Pokémon classes. ConvMixer led experiment 1 with 71.09% accuracy using just 0.7 M parameters, closely followed by fine-tuned ResNet-18 (69.58%, 11.3 M parameters) and Extra Trees on its head features (66%). In experiment 2, a frozen-head ViT achieved 82.71% (86.4 M parameters), showing that powerful pretrained backbones combined with rich downstream heads can excel. Overall, modern convolutional hybrids offer the best efficiency–performance tradeoff, while tree-based classifiers on task-adapted embeddings provide strong, interpretable results with minimal retraining.

Thank you!

Questions?