Exploring Transfer Learning Strategies for Multi-Label Classification

Layanne El Assaad
Neural Networks Deep Learning
Columbia University
New York, NY
layanne.a@columbia.edu

Abstract

This project explores transfer learning for multi-label image classification of animals, where each image is labeled with both a superclass and a subclass. I compare a baseline CNN to pretrained ResNet18 and EfficientNet-B0 models using linear, partial, and full fine-tuning. To handle novel classes at test time, I design a dual-head architecture with Mixup augmentation for regularization. Results show that full fine-tuning of EfficientNet achieves the best accuracy and generalization, demonstrating the strength of transfer learning in hierarchical label settings.

1 Introduction

Transfer learning enables efficient learning on new tasks by leveraging pretrained models. In vision tasks, models like ResNet and EfficientNet have proven effective, especially when data is limited. This project investigates transfer learning for hierarchical, multi-label classification—assigning both superclass and subclass labels to animal images.

A key challenge is generalizing to novel subclasses during testing, a scenario common in real-world applications like biodiversity monitoring. I implement a dual-head model that predicts both label types and evaluate three fine-tuning strategies: linear, partial, and full. The goal is to assess how each approach handles generalization, especially in the presence of distribution shifts and open-set conditions. Mixup regularization is applied to enhance robustness. This study offers insights into effective transfer learning for complex, low-resource classification tasks.

2 Dataset Overview

This project utilizes a multi-label image classification dataset consisting of approximately 6,600 color images, each sized 64×64 pixels. Every image is annotated with two hierarchical labels:

A superclass from one of three categories: bird, dog, or reptile.

A corresponding subclass, totaling 88 fine-grained categories (e.g., hawk, chihuahua, mud turtle).

These subclass labels are organized under their respective superclasses. For instance:

1. Hawk is a subclass of bird
2. Toy terrier is a subclass of dog
3. Mud turtle is a subclass of reptile

Note: The test set is provided without ground truth labels and is only used for leaderboard submissions.

Preprint. Under review.

Although the training set includes examples for seen superclasses and their subclasses, the test set may include novel superclasses and subclasses that do not appear in training. As a result, the model must be capable of open-set recognition, predicting a "novel" label when necessary. This makes generalization to unseen classes a central challenge.

To better simulate real-world settings, the superclass distribution in the training set is imbalanced, with some superclasses having many more examples than others. For instance, many subclasses, like that of the Komodo dragon or tree frog, have twice as many samples as the others.

3 Methodologies

3.1 Data Preprocessing

In this project, all input images are uniformly resized to a resolution of 6464 pixels using $transforms.Resize$. This ensures compatibility with both the custom CNN architecture and pretrained models ($ResNet18$ and $EfficientNet - B0$) which expect fixed-size inputs.

To enhance the model's ability to generalize and reduce overfitting, a sequence of data augmentations is applied only to the training set. The augmentations are implemented using $torchvision.transforms$ and include:

1. $RandomHorizontalFlip()$: Horizontally flips the image with a probability of 0.5. This helps the model learn invariance to object orientation.
2. $RandomRotation(10)$: Randomly rotates the image within ±10 degrees to simulate viewpoint variations.
3. $ColorJitter(brightness = 0.2, contrast = 0.2, saturation = 0.2)$: Randomly adjusts brightness, contrast, and saturation to simulate lighting changes and improve robustness to color variation.

After augmentation, all images are normalized using: $mean = [0.5, 0.5, 0.5]$ and $std = [0.5, 0.5, 0.5]$ This normalization scales pixel values from the $[0, 1]$ range (after ToTensor) to approximately $[-1, 1]$, which improves training stability and convergence speed.

A custom $TransferLearningDataset$ class is implemented, which handles:

1. Loading image paths from the filesystem ($train\_images$ directory).
2. Reading annotations from $train\_data.csv$, and label metadata from $superclass\_mapping.csv$ and $subclass\_mapping.csv$.
3. Dynamically mapping integer indices to class names (and vice versa) for both superclasses and subclasses.
4. Returning a tuple per sample:

$$(image\ tensor, superclass\ index, superclass\ label, subclass\ index, subclass\ label)$$

The data set is partitioned into the following:

1. Training set: approximately 90% of the labeled images (used for model learning)
2. Validation set: approximately 10% of the labeled data (used for hyperparameter tuning and evaluation)

3.2 Models Used

In this project, we experiment with three different neural architectures to evaluate performance on hierarchical image classification and generalization to novel classes. Each model is adapted to produce dual-head outputs for both superclass and subclass predictions. All models employ a dual-head architecture that simultaneously predicts both the superclass (one of four: bird, dog, reptile, or novel) and the subclass (one of 89: 88 known subclasses plus a novel class). This structure enables coarse-to-fine classification while explicitly accounting for unseen categories during inference.

We explore three fine-tuning strategies: (1) Linear probing, where all pretrained weights are frozen except the classification heads; (2) Partial fine-tuning, where only the last block and classifier are trainable; and (3) Full fine-tuning, where all layers are updated. These strategies are applied to both ResNet18 and EfficientNet-B0.

3.2.1 CNN Baseline: Custom 3-Block ConvNet

We begin with a custom convolutional neural network (CNN) built from scratch using three convolutional blocks. Each block consists of a convolutional layer, ReLu activation, Batch normalization, and Max Pooling.

This is followed by two fully connected layers, and two parallel linear heads:

- One for predicting the superclass label (output size = 4)
- One for predicting the subclass label (output size = 88)

A CNN was used because it provides a lightweight baseline for comparison against larger pretrained models and trains quickly and can highlight when performance gains come from data/model complexity rather than better learning.

3.2.2 ResNet18: Pretrained with Linear / Partial / Full Fine-Tuning

The ResNet18 architecture is a well-known residual network with 18 layers, pretrained on ImageNet. The original classification head of ResNet is removed and replaced with a shared fully connected layer and two classification heads for super- and subclass predictions.

ResNet18 was used because it is a strong general-purpose backbone that captures mid-level and high-level visual features. Furthermore, The residual connections in ResNet help mitigate the vanishing gradient problem, improving deep network training stability. Also, Pretraining on ImageNet enables effective transfer learning, especially for small datasets like ours.

3.2.3 EfficientNet-B0: Pretrained with Linear / Partial / Full Fine-Tuning

EfficientNet-B0 is a modern convolutional architecture that scales depth, width, and resolution in a principled way. Like ResNet18, the model is pretrained on ImageNet and adapted with a hared linear projection layer after the backbone and two classification heads (superclass and subclass).

EfficientNet was used because it offers better parameter efficiency and higher accuracy per FLOP compared to ResNet. It is well-suited to low-resolution inputs (64×64) and smaller training datasets. Also, Tte model often achieves great results on fine-grained classification tasks due to its scalable feature representation.

By evaluating all three architectures under different training regimes, we are able to benchmark performance against a simple CNN baseline and study the effectiveness of fine-tuning on learned representations from ImageNet when adapting to a hierarchically labeled dataset.

3.3 Loss and Regularization

To train the dual-head models for simultaneous superclass and subclass prediction, we use the CrossEntropyLoss for both outputs. Specifically, the loss from each head is computed independently and summed to form the final objective:

$$\mathcal{L}_{total} = \mathcal{L}_{super} + \mathcal{L}_{sub}$$

During training, we also apply Mixup regularization, a data-level augmentation strategy that improves generalization and robustness to overfitting. At each batch, two random images and their labels are mixed using a convex combination controlled by a sampled coefficient λ:

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \quad \tilde{y} = \lambda y_i + (1 - \lambda)y_j$$

This is applied jointly to both superclass and subclass labels, effectively encouraging the model to interpolate between classes and reducing sensitivity to noisy inputs.

Additionally, to prevent overfitting and promote smoother weight updates, L2 regularization (via weight decay) is incorporated into the optimizer, with a small decay factor applied to all trainable parameters.

Mixup and weight decay stabilize training, especially under class imbalance and open-set generalization conditions.

4 Novelty Detection

To address the presence of unseen (novel) categories in the test set, we implement a confidence-based novelty detection mechanism during inference. Since the test data may contain images from both novel superclasses and novel subclasses, the model must be able to abstain from assigning them to any of the known (seen) classes.

During prediction, softmax probabilities are computed for both superclass and subclass logits. If the model's maximum softmax confidence for a prediction is below a predefined threshold, the image is labeled as "novel" for that particular task (superclass or subclass).

Confidence Threshold:
A softmax confidence threshold of

$$\texttt{CONFIDENCE\_THRESHOLD} = 0.6$$

is applied to both outputs. Predictions with maximum softmax probability below this value are considered uncertain and are labeled as novel.

Novelty Labels:

- `NOVEL_SUPER_IDX` = 3
  This assumes known superclasses are indexed as 0: bird, 1:dog, 2:reptile, 3:"novel".
- `NOVEL_SUB_IDX` = 89
  This assumes that subclass indices range from `0` to `88`, and `89` corresponds to the "novel" subclass.

5 Evaluation Metrics

Model performance is evaluated across four key metrics:

- **Seen Superclass Accuracy:** Accuracy on superclass labels present in the training set.
- **Novel Superclass Accuracy:** Accuracy on unseen superclass labels encountered only at test time.
- **Seen Subclass Accuracy:** Accuracy on subclass labels present during training.
- **Novel Subclass Accuracy:** Accuracy on subclass labels not seen during training.

These are computed using the `validate_seen_vs_novel(...)` function, which uses superclass and subclass index sets from the training data to distinguish between seen and novel categories.

6 Results

I trained each model for 10 epochs and tracked loss and accuracy on both superclass and subclass tasks. The curves in Figure 1 show consistent convergence across models, with ResNet18 with full fine-tuning achieving lowest training and validation losses across epochs, while EfficientNet linear has the highest training and validation losses.

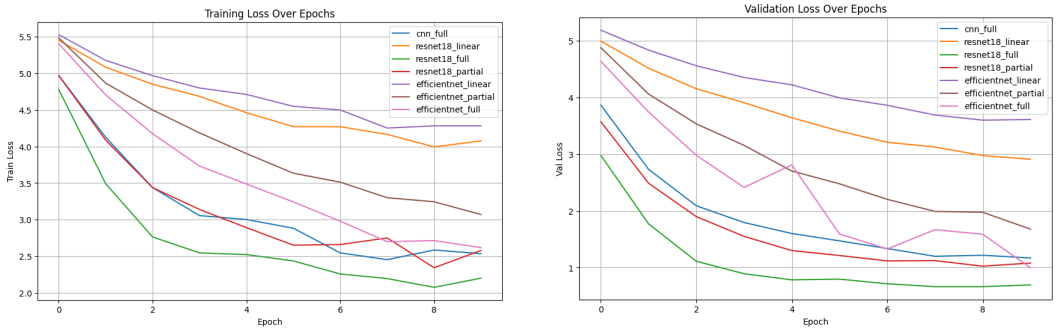
ResNet18 (full) and EfficientNet (full) outperform their linear and partial variants. Surprisingly, CNN shows has competitive convergence overall despite training from scratch.

Figure 1c shows that all models exceed 85% superclass accuracy. ResNet18 (full) and EfficientNet (full/partial) reach about 99%. The CNN baseline performs well (97%), while linear probing variants for both ResNet18 and EfficientNet plateau below 90%. Subclass results in Figure 1d showcase

a larger gap between the models, where ResNet18 (full) achieves the highest accuracy (88.7%), followed by ResNet18 (partial) and EfficientNet (full) (80.6%). CNN reaches 76.2%, while linear variants underperform, staying below 38%. Table **??** reports final validation accuracy. Since the training set contained no novel categories, all models yield 0% accuracy on novel classes, as expected.

Model	**Seen Superclass Accuracy (%)**	**Seen Subclass Accuracy (%)**
CNN Baseline	96.50	76.15
ResNet18 - Linear	86.49	37.68
ResNet18 - Partial FT	99.05	74.88
ResNet18 - Full FT	99.05	**88.71**
EfficientNet - Linear	86.49	29.25
EfficientNet - Partial FT	97.62	57.23
EfficientNet - Full FT	**99.05**	80.60

Table 1: Validation accuracy on seen superclasses and subclasses. Novel class accuracies are omitted since the training set contains no novel categories; as expected, models default to 0% accuracy on novel classes.



(a) Training Loss Over Epochs

(b) Validation Loss Over Epochs

(c) Superclass Accuracy Over Epochs

(d) Subclass Accuracy Over Epochs

Figure 1: Training and evaluation metrics for each model and fine-tuning strategy. Full fine-tuning of pretrained models achieves the lowest loss and highest accuracy across tasks.

7 Analysis

The most significant improvements appear in subclass prediction, which is inherently more fine-grained. Full fine-tuning of ResNet18 achieves the best subclass accuracy (88.71%) (as well as superclass accuracy and lowest losses), outperforming all other models. EfficientNet (full) also performs strongly (80.60%). This highlights that enabling all layers to adapt to the new dataset allows the model to specialize in subtle subclass distinctions, especially important when training data is limited.

Both ResNet18 and EfficientNet with linear probing (i.e., frozen backbone) underperform across all metrics, particularly in subclass accuracy (37.68% and 29.25% respectively), and perform far worse than the CNN baseline. This suggests that fixed pretrained features are not sufficient for transfer to our domain, which involves small resolution images (64×64) and a different label structure from ImageNet.

Partial fine-tuning (last block + heads) improves over linear probing and reaches competitive performance without updating the entire network. ResNet18 (partial) achieves 74.88% subclass accuracy, which is close to the CNN baseline, while EfficientNet (partial) improves to 57.23%. This strategy offers a good compromise when computational constraints or overfitting risks are concerns.

Surprisingly, the CNN trained from scratch performs well (76.15% subclass accuracy), surpassing all partial and linear transfer variants. This suggests that for small images and moderate data sizes, a compact architecture can learn robust representations—especially with Mixup augmentation enhancing generalization.

All models achieve high superclass accuracy (above 96%) after a few epochs, with relatively little variation between methods. This indicates that coarse-level class boundaries (e.g., bird vs. dog vs. reptile) are easier to learn, even with frozen backbones. Therefore, improvements from transfer learning are most evident in fine-grained subclass prediction.

Figures 1a and 1b show that ResNet18 (full) consistently exhibits the lowest losses. Validation curves also show minimal overfitting, indicating that regularization (e.g., Mixup) are effective. In contrast, linear variants converge slowly and plateau early, reinforcing the need for feature adaptation.

Fine-tuning all layers of a pretrained model provides the best performance, especially for subclass classification. ResNet18 (full) is the overall top performer, followed by EfficientNet (full), while linear probing consistently underperforms. The CNN baseline remains a surprisingly competitive choice, particularly when resources are constrained.

8 Test Time Inference

At test time, predictions are made using the `predict_with_novelty()` function, which performs forward passes on batches of test images using the trained model. The prediction pipeline includes both superclass and subclass outputs, with softmax confidence scores applied to each.

To handle potential novel categories in the test set, a softmax confidence threshold is applied. Specifically:

- `CONFIDENCE_THRESHOLD` is set to 0.6.
- If the maximum softmax confidence for a prediction is below this threshold, the sample is labeled as `novel`.
- For superclass predictions, the novel class is assigned index `3`, assuming the seen classes are indexed as `0: bird`, `1: dog`, `2: reptile`.
- For subclass predictions, the novel class is assigned index `87`, assuming training subclasses are indexed from 0 to 86.

The test dataset is loaded using a custom `CustomTestDataset` class, which loads test images from a directory, applies preprocessing transformations consistent with training, sorts image filenames numerically to preserve order.

Predictions are obtained by passing batches through the model and computing softmax probabilities. If the confidence of a predicted label is below 0.6, it is replaced with the corresponding novel index.

Predictions are saved in CSV files with the following format:

```
image, superclass_index, subclass_index
```

Each row corresponds to one test image, and the predictions reflect either the most confident known class or the designated novel index based on the thresholding rule.

The `test_all_models()` function automates inference across all model variants, applying the same novelty-aware inference logic. Each model's predictions are saved to a separate CSV file (e.g., `predictions_resnet18_full.csv`) for downstream evaluation or submission.

9 Conclusion

This project evaluated transfer learning strategies for hierarchical multi-label image classification under limited data and open-set conditions. We compared a custom CNN with pretrained ResNet18 and EfficientNet-B0 models using linear, partial, and full fine-tuning.

Results show that full fine-tuning consistently yields the best performance, with ResNet18 achieving the highest subclass accuracy (88.71%). EfficientNet-B0 also performs strongly when fully fine-tuned. In contrast, linear probing underperforms, highlighting the importance of adapting pretrained features to the target domain.

Notably, the custom CNN trained from scratch achieved competitive results (76.15%), outperforming all linear and partial fine-tuning variants. This suggests that with regularization (e.g., Mixup), lightweight models can remain effective on small datasets.

Overall, the findings confirm that transfer learning is most effective when the entire model is fine-tuned, enabling better adaptation to fine-grained subclass distinctions—especially important for real-world scenarios involving novel or imbalanced classes.