Transfer Learning for Object Recognition using Pretrained CNN Models

Adithya Sivakumar*, Amudhan A*, Rhea Sankar*
*Department of CSE, Sri Sivasubramaniya Nadar College of Engineering, Chennai, India
Project Repository

***Abstract*—Transfer learning has significantly improved object recognition performance through the use of pre-trained deep convolutional neural networks. In this study, we apply three pre-trained architectures—MobileNetV2, EfficientNetB0, and ResNet18—to the Caltech-101 dataset. We compare feature extraction and fine-tuning strategies and analyze the resulting classification performance across these models.**

***Index Terms*—Transfer Learning, Object Recognition, Caltech-101, MobileNetV2, EfficientNetB0, ResNet18, Deep Learning**

I. Introduction

Object recognition and differentiation have long been claimed as fundamental problems in the field of computer vision. In our work, we mitigate limitations inherent in training neural networks from scratch, such as high computational power requirements and excessive resource utilization, via transfer learning. Transfer learning makes use of models that are initialized with weights pre-trained on the ImageNet dataset.

In our study, we apply the concept of transfer learning to the Caltech-101 dataset and compare various convolutional architectures on the parameters of accuracy, generalization, and validity.

II. Dataset Description

In our work, we utilize the Caltech-101 dataset [1], which contains images from 101 object categories, with approximately 40–800 images per category. Most categories contain around 50 images. The average resolution of the images was 300x200 pixels.

III. Methodology

Our end-to-end pipeline consists of:

1) Dataset Preparation
2) Pretrained Model Loading
3) Feature Extraction
4) Fine-Tuning
5) Testing and Evaluation

A. Dataset Preparation

The dataset was organized by splitting it into three subsets—test, training, and validation. To meet the input requirements of the pre-trained model, the images were resized to 224 × 224 pixels. Generalization performance was improved by applying data augmentation techniques such as random flipping and rotation.

B. Pretrained Model Selection

We evaluated three pretrained architectures:

- **MobileNetV2** — Lightweight and efficient architecture optimized for mobile and embedded vision applications. Dataset split 66-33 (training and test respectively)
- **EfficientNetB0** — Compound scaling-based model balancing width, depth, and resolution. Dataset split 70-15-15(training, test, and validation respectively)
- **ResNet18** — Residual network architecture using skip connections to mitigate vanishing gradient problems. Dataset split 80-20(training and test respectively)

All models were initialized with ImageNet pretrained weights.

C. Feature Extraction

CNNs pre-trained on the ImageNet dataset were employed as fixed feature extractors in the feature extraction phase. The base of each architecture was initialized with pre-trained weights, and all convolutional layers were frozen. This was done to preserve low and mid-level visual representations. Only the classifier head was trained.

1) MobileNetV2: For MobileNetV2, the convolutional base was loaded without the top classification layer. All pre-trained convolutional layers were frozen during this phase. To reduce the spatial feature maps, a global average pooling layer was appended. This was followed by a dropout layer for regularization. Finally, a dense layer with softmax activation and 102 output units was added.

MobileNetV2 employs depth-wise separable convolutions, which decompose standard convolutions into depth-wise and point-wise operations. This design is considered advantageous as it reduces computational complexity while maintaining effective feature extraction capability. In the feature extraction phase, only the parameters of the newly added classification head were updated, while the network acted as a fixed high-level feature encoder.

2) ResNet18: In the case of ResNet18, all residual blocks were frozen, and the pre-trained convolutional backbone was initialized with ImageNet weights. To ensure homogeneity with the Caltech-101 class count, the original classification layer was replaced with a new dense layer.

ResNet18 utilizes residual skip connections, which facilitate stable gradient propagation and deep feature learning. In the feature extraction phase, progressively abstract representations were extracted by the residual layers, functioning as fixed hierarchical feature encoders. Only the parameters of the final classifier layer were optimized.

3) EfficientNetB0: For EfficientNetB0, the pre-trained backbone was loaded without its top classification layer. To retain pre-trained representations, all convolutional layers were frozen. Similar to the MobileNetV2 model, a custom classification head consisting of global average pooling, dropout regularization, and a dense softmax layer with 102 output units was appended.

EfficientNet employs compound scaling to balance network depth, width, and resolution. The multi-scale feature maps generated by the backbone in the feature extraction phase were mapped to class probabilities by the newly added classifier. Only the classification head was trained, while the pre-trained convolutional filters remained unchanged.

D. Fine-Tuning

In the second training phase, selected convolutional layers of each pre-trained backbone were unfrozen and optimized along with the classification head. To prevent large weight updates that could disrupt previously learned representations, a lower learning rate was used. Fine-tuning the models helped retain the general features learned from ImageNet.

1) MobileNetV2: For MobileNetV2, the convolutional backbone was unfrozen post the initial feature extraction phase. Stable optimization was ensured by performing fine-tuning using a reduced learning rate. The model adapted high-level semantic features to better capture dataset-specific patterns present in Caltech-101 by allowing gradient updates to propagate through the deeper layers. This refinement improved class discrimination while preserving the efficiency of the architecture.

2) ResNet18: In the case of ResNet18, the pre-trained residual blocks were unfrozen at this stage. To prevent overfitting and catastrophic forgetting, the model was retrained using a smaller learning rate. Significant improvement was observed in the alignment between the extracted features and the Caltech-101 class distribution by fine-tuning the residual layers to adjust their learned representations to the target dataset. During this adaptation process, the skip connections ensured stable gradient flow.

3) EfficientNetB0: Similar to the two prior models, for EfficientNetB0, the convolutional backbone was unfrozen and fine-tuned using a reduced learning rate. This allowed the compound-scale architecture to refine the multi-scale feature representations to be coherent with the characteristics of the Caltech-101 dataset. There was an observable improvement in the model's ability to distinguish between visually similar categories as a result of updating the higher-level convolutional filters. Strong generalization performance was still maintained.

E. Training Details

Models were trained using cross-entropy loss and the Adam optimizer. Performance was evaluated using classification accuracy on the validation set.

IV. Results and Discussion

A. EfficientNetB0

The EfficientNetB0 model was first trained with its pre-trained ImageNet weights on the Caltech-101 dataset to classify 102 types of images. The Adam optimizer, along with the binary cross-entropy loss function, was employed to calculate the gradient loss, and ReLU activation functions were used for the convolutional layers. The final output layer utilized the Softmax function to obtain the probability distribution across 102 classes. The initial training loop was performed without freezing the backbone layers, and the subsequent training loop was performed with frozen backbone layers for 10 epochs.

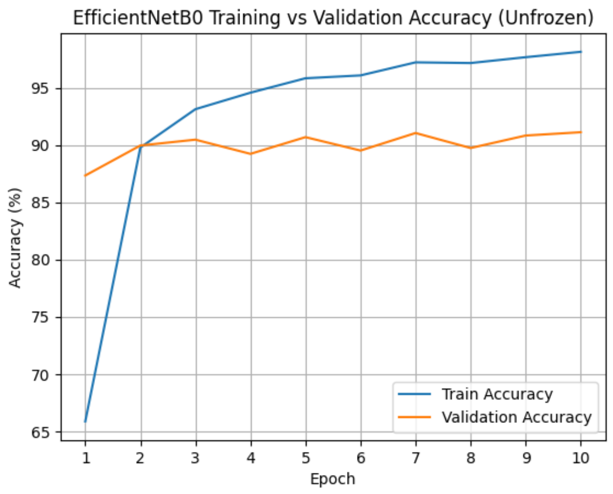

Fig. 1. EfficientNetB0 Training and Validation Accuracy (unfrozen)

Figs. 1 and 2. illustrate the training and validation accuracy scores under frozen and unfrozen conditions. In the fine-tuned setting, the model converged rapidly with training accuracy increasing at a constant rate, while the validation accuracy finally stabilized at around 0.9110. A gap of around 7 percent between the training and validation accuracy at the end shows that the model is moderately overfitting. In contrast, the model with frozen backbone layer showed a lower validation accuracy, around 0.960, showing that the model is failing to adapt without the backbone layers. But the final test scores revealed that the unfrozen model performed significantly better than the frozen model, scoring around 97.02 percent accuracy, while the frozen model scored around 88.87 percent.

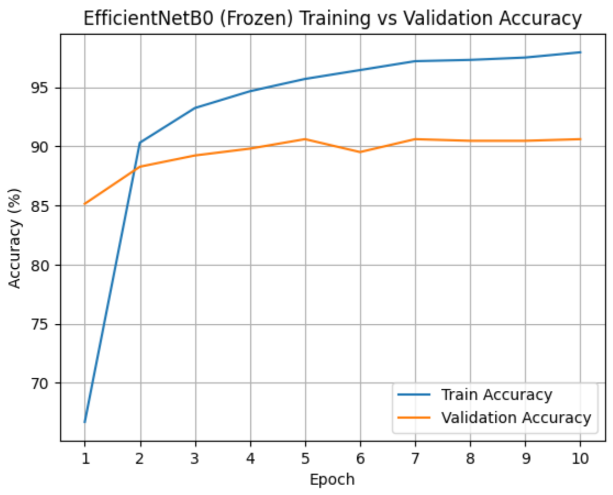

Fig. 2. EfficientNetB0 Training and Validation Accuracy (frozen)

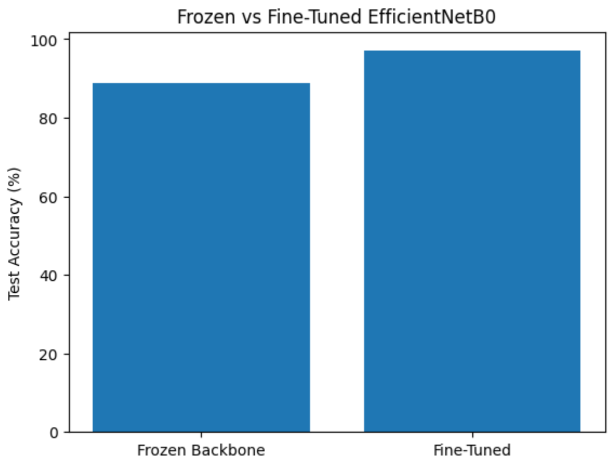

Fig. 3. EfficientNetB0 Test Accuracy Comparision

B. MobileNetV2

The Caltech-101 dataset was utilized to train the MobileNetV2 model to classify between 102 different types of items. To achieve the class probability distributions, the final classification layer was replaced with a dense layer consisting of 102 output units activated by Softmax after the model was initialized with the pretrained weights.

The Adam optimizer and the Sparse Categorical Cross-Entropy loss function were utilized for the multi-class classification problem. The layers in the backbone network took advantage of the ReLU activations.

Two phase training was conducted. Initially, the backbone was frozen and the newly added classification was trained for a duration of 10 epochs which allowed for the leverage of pretrained ImageNet features along with the adaption of the final decision layer to the Caltech dataset. Following this, the backbone was unfrozen and fine-tuned using a lower learning rate which enabled the filters to adapt to dataset-specific patterns which subsequently improved class discrimination performance.

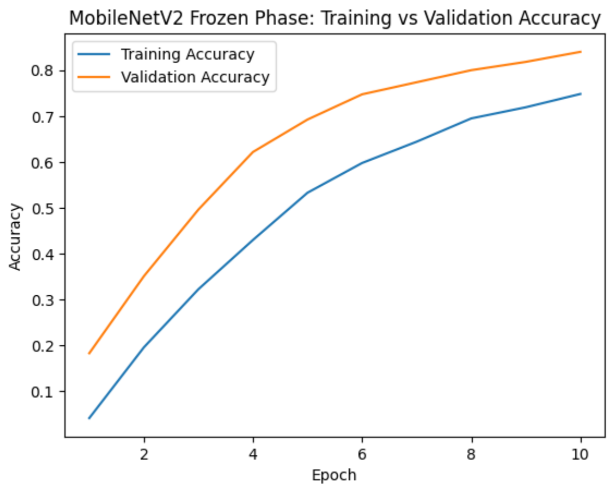

Fig. 4. MobileNetV2 Training and Validation Accuracy (frozen)

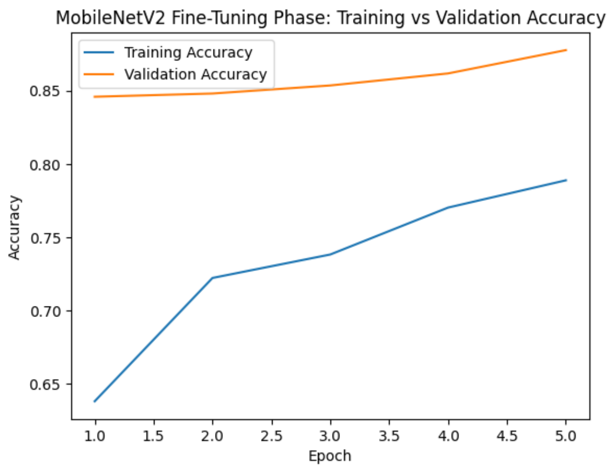

Fig. 5. MobileNetV2 Training and Validation Accuracy (unfrozen)

From Fig. 4 and Fig. 5, it was observed that during the frozen backbone period, both training and validation accuracy increased steadily across epochs, attaining approximately 74.8% and 83.9%, respectively. This indicates stable convergence and minimal overfitting. Notably, the validation accuracy remained slightly higher than the training accuracy, which can be attributed to data augmentation applied exclusively to the training set.

In the unfrozen phase, retraining with a reduced learning rate further improved performance. Validation accuracy increased from approximately 84% to 87.7%, while training accuracy reached nearly 79%. The upward trend in validation performance illustrates that adapting higher-level feature representations to the Caltech-101 dataset improves class discrimination and the lack of overfitting.

C. ResNet18

The ResNet18 model was also used to evaluate under the Caltech-101 dataset, with ImageNet pre-trained weights, initially. The final layer was modified to adapt to output probability of each of the 102 classes, using Softmax function . The model as evaluated with it's convolutional layer frozen and later, the convolutional layer unfrozen to better adapt to the dataset.

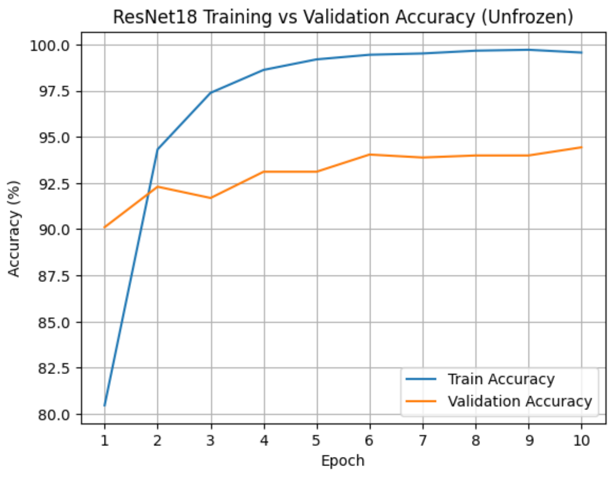

Fig. 6. ResNet18 Training and Validation Accuracy (unfrozen)

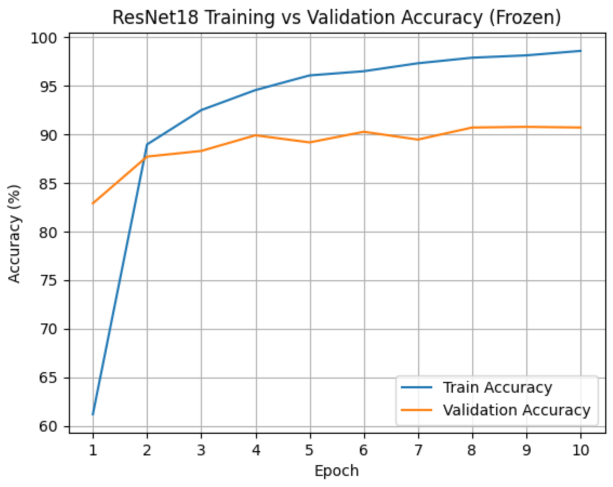

Fig. 7. ResNet18 Training and Validation Accuracy (frozen)

In the frozen model's training loop, the training accuracy slowly converged from around 61.19 percent to 98.56 percent at the end, while the validation score varied mostly around the 90 percent range, attaining the best value of 90.75 percent at the 9th epoch with no further improvement. The gap between the training and validation accuracy (around 8 percent) shows mild overfitting of the model.

The unfrozen model showed a better performance than the frozen version, starting from 90 validation accuracy which converged quickly to 94.43 percent. The training accuracy finally exceeded 99 percent. Considering the gap of 5 percent between the training and validation scores, the model only exhibited light overfitting. However, it showed an accuracy of 95.64 percent in the test run.

Overall, the unfrozen model performed better than the model with frozen convolutional layer, suggesting the effectiveness of updating the convolutional layer's weights in transfer learning tasks.

D. Comparative Analysis

TABLE I
MODEL PERFORMANCE COMPARISON ON CALTECH-101

Model	Frozen Val Acc (%)	Fine-Tuned Val Acc (%)	Test Acc (%)
MobileNetV2	83.9	87.7	82.13
EfficientNetB0	96.0	91.1	97.02
ResNet18	90.75	94.43	95.64

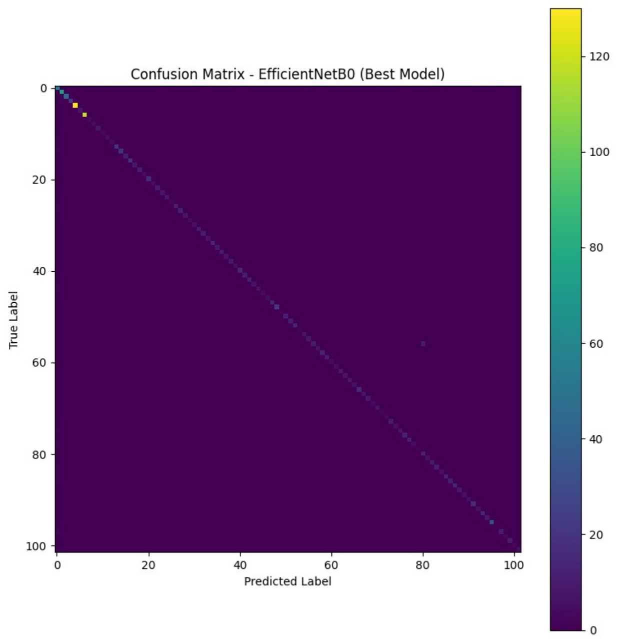

Fig. 8. Confusion Matrix of Fine-Tuned EfficientNetB0 on Test Set

The results indicate that the freezing–unfreezing strategy improved the overall performance of the models. EfficientNetB0 achieved the highest test accuracy of 97.02%, demonstrating superior feature adaptability. Following unfreezing, a similar performance gain was observed in ResNet18, which attained a validation accuracy of 94.43%. Although MobileNetV2 achieved a lower absolute accuracy, it reached 87.7% while requiring reduced computational resources, which may be preferable in resource-constrained settings. Overall, the findings suggest that selectively freezing and unfreezing pre-trained convolutional backbones significantly enhances classification performance compared to using them merely as fixed feature extractors.

V. Conclusion

This work has proven the effectiveness of transfer learning in object recognition tasks, particularly in scenarios that involve small datasets. Fine-tuning always improved the results across all the architectures that were tested. EfficientNetB0 produced stable results while maintaining a good trade-off between accuracy and efficiency, while ResNet18 showed stable and sound results. MobileNetV2 produced accurate results with relatively lower computational complexity.

Future research may include exploring deeper variants of the architectures and hyperparameter optimization.

References

[1] F.-F. Li, M. Andreeto, M. Ranzato, and P. Perona, “Caltech 101,” CaltechDATA, 2022. doi: 10.22002/D1.20086.