Animal Classification Competition Report

Introduction

This report details our approach to enhancing the baseline convolutional neural network for classifying images of 151 different animal species. The dataset consists of 6270 RGB images distributed across 151 folders, each representing a different animal category. Our primary goal was to improve the classification accuracy of the model while considering the computational efficiency. The final model gains **94.06% test accuracy** and **2103.04 accuracy/GFLOPs ratio**, which **increases the accuracy around 57.97% (2.61 times better)** and **boosts the efficiency for 20.71 times better than the baseline model.**

Understanding of task and baseline model

The task is to categorize each animal image into one of the 151 categories. The provided baseline code includes essential steps such as loading and analysing the dataset using torchvision, defining a simple convolutional neural network (CNN), and training and testing the network. The baseline model achieved a test accuracy of 36.09% with a computational cost of 0.3555 GFLOPs. The baseline model is a simple CNN composed of four convolutional layers and a fully connected layer. The first convolutional layer (Conv1) has 64 filters with a kernel size of 5x5 and a stride of 1. The next three convolutional layers (Conv2 to Conv4) each use 128 filters with a kernel size of 3x3 and a stride of 1. Following the convolutional layers, a fully connected layer maps the flattened feature maps to 151 classes. Each convolutional layer is followed by a ReLU activation function and a max-pooling layer with a kernel size and stride of 2x2. The final fully connected layer outputs the softmax probabilities for each class. The input size is reduced to 112 pixels through transformations such as resizing, random horizontal flipping, and center cropping.

The baseline model, while simple and effective for initial exploration, has limitations in terms of capacity and generalization. To address these limitations, we explored several advanced techniques including data augmentation, transfer learning, fine-tuning, and optimization of training parameters.

Modifications and Improvements

Data Augmentation

We applied several data augmentation techniques to increase the diversity of the training dataset and reduce overfitting. The transformations included resizing (224), random horizontal flipping, random rotation, colour jittering, and random resized cropping. And normalizes the tensor image with the specified mean and standard deviation values commonly used for pre-trained models on ImageNet. These augmentations helped the model generalize better to unseen data.

Transfer Learning and Fine-Tuning

We leveraged pre-trained models to utilize existing knowledge from larger datasets like ImageNet. Specifically, we used ResNet18, MnasNet0_5 and ShuffleNet_v2_x0_5 models. Transfer learning enabled us to achieve higher accuracy with less training time and computational resources. In the fine-tuning approach, we froze the pre-trained model's layers and only trained the final fully connected layer to adapt it to our specific dataset. This method helps to maintain the learned features from the pre-trained model while adapting to our task.

Learning Rate Optimization

We optimized the learning rate to improve the training efficiency of each model. For ResNet18 models, we used a learning rate of 0.001. For MnasNet0_5, we used a higher learning rate of 0.01, and even higher learning rate for ShuffleNet to better suit their lightweight architecture and achieve faster convergence.

Efficiency Calculation

Efficiency is defined as the ratio of accuracy (%) to GFLOPs. We reported the computational cost and efficiency for each model, highlighting the trade-off between accuracy and computational resources.

Ablation Study

From the following results of ablation study, we found that using data augmentation with transfer learning (Resnet18) and do full training can significantly improve the accuracy from 36% to 75%. However, by freezing the initial layers of pre-trained model and fine-tuning can further improve the accuracy till 96%. Besides, via using smaller model with

higher learning like MnasNet and ShuffleNet, the accuracy is not as good as the ResNet18 (with 94.06%) but very close, it instead dramatically increases the efficiency, which we will discuss in the next section. In conclusion, the final best model achieves 94.06%, which **gains 57.97% accuracy increase compared to baseline (2.61 times better).**

Method	Data Augmentation	Transfer learning	Fine tuning	Learning rate	Accuracy (%)	GFLOPs	Efficiency (acc%/GFLOPs)
Baseline	*No*	*No*	*No*	*0.001*	*36.09*	*0.3555*	*101.53*
Resnet18	*Yes*	*Yes*	*No*	*0.001*	*75.61*	*1.8261*	*41.40*
Resnet18	*Yes*	*Yes*	*Yes*	*0.001*	*96.09*	*1.8261*	*52.62*
MnasNet0_5	*Yes*	*Yes*	*Yes*	*0.001*	*93.95*	*0.1187*	*791.56*
MnasNet0_5	*Yes*	*Yes*	*Yes*	*0.01*	*94.84*	*0.1187*	*799.05*
ShuffleNet	*Yes*	*Yes*	*Yes*	*0.02*	*94.06*	*0.0447*	*2103.04*

Efficiency and Trade-offs

To reduce computational cost and improve the trade-off between accuracy and efficiency, we focused on model selection and learning rate optimization:

1. **ResNet18 (Fine-Tuning)**: This model achieved a high accuracy of 96.25%, but the computational cost was quite high at 1.8261 GFLOPs, which is approximately 5.5 times greater than the baseline model. While the accuracy is impressive, the high GFLOPs indicate a significant computational load.

2. **MnasNet and ShuffleNet**: To address the high computational cost of ResNet18, we firstly selected MnasNet0_5, which is designed for mobile and resource-constrained environments. Initially, MnasNet0_5 provided a much lower GFLOPs of 0.1187, which is about one-third of the baseline model's computational cost. However, the accuracy dropped slightly to 93.95%. Then we tried the ShuffleNet_v2_x0_5 model which cost only 0.0447 GFLOPs (about one eighth of baseline model), and the accuracy can still remain around 94%.

3. **Learning Rate Optimization**: To improve the performance of MnasNet0_5, we increased the learning rate from 0.001 to 0.01. This adjustment helped boost the accuracy to 94.84%, and when we apply even bigger learning rate 0.02 on ShuffleNet, the accuracy can remain at 94.06% and makes the efficiency reach the astonishing 2100+, striking a better balance between accuracy and computational efficiency. The higher learning rate allowed the model to converge faster and achieve higher accuracy with minimal computational cost.

By implementing these strategies, we significantly enhanced the classification accuracy while managing and, in some cases, reducing the computational cost. These methods ensured a better trade-off between accuracy and efficiency, as evidenced by the high efficiency score of the ShuffleNet_v2_x0_5 model. In conclusion, the final ShuffleNet model achieves **2103.04 acc%/GFLOPs ratio, which is 20.7 times better than the baseline model.**

Limitations and Conclusions

This study has several limitations:

- Models were trained for only 10 epochs, more epochs could improve performance and provide better comparison.
- MnasNet and ShuffleNet are not the smallest model; we can explore more lightweight models for better results.
- Hyperparameter tuning was limited, and more data augmentation weren't applied due to time constraints.
- Models showed signs of overfitting, additional regularization may be needed.
- The final model highly relies on the pre-trained model, and we only fine-tuned the fully connected layers, which may not always capture domain-specific knowledge like animals, maybe fine tune more layers help.

In conclusion, while the fine-tuned ResNet18 model achieved the highest accuracy, **ShuffleNet_v2_x0_5** offered the best balance between accuracy and computational efficiency. By leveraging transfer learning, fine-tuning, and learning rate optimization, we significantly enhanced the classification accuracy and efficiency. This underscores the value of pre-trained models and efficient architectures in high-performance image classification tasks. Future work could focus on extended training, more comprehensive hyperparameter tuning, and advanced data augmentation to further optimize model performance.