# Assignment 3 Part 2 Report | Haider Ali Lokhand | a1894658

## Introduction of the Baseline Model

This code is an example of training a convolutional neural network (CNN) for image classification using PyTorch and the torchvision library. It goes through the entire process, from loading the dataset, defining the model architecture, training the model, evaluating its performance, and even calculating the FLOPs (floating point operations) required by the model. The dataset contains a total of 6270 images corresponding to the name of animal types with 151 classes. The baseline model contains a simple convolutional neural network architecture built on pytorch that uses torchvision library for image transforms and utils for data loaders, data splitting and data visualization. A baseline CNN architecture is used with 4 convolutional layers with ReLU activation function and 1 fully connected layer with ReLU activation followed by an output layer with 151 nodes with Softmax activation for classification. The code defines functions to detect and set the device to either GPU (if available) or CPU. It also defines a class `DeviceDataLoader` to move data batches to the chosen device. The `fit` function is defined to train the model for a specified number of epochs. It uses stochastic gradient descent (SGD) or another optimizer (like Adam) to update the model's parameters. The `evaluate` function calculates validation loss and accuracy. The training and validation progress is stored in the `history` list. The training loop iterates over epochs, performing training and validation steps in each epoch. The `plot_accuracies` and `plot_losses` are used to plot accuracy and loss curves during training. The model's performance is evaluated on the test set using the `evaluate` function and the `test_loader`. The model efficiency is determined by ratio of accuracy to GFlops (Computational cost - the number of floating-point operations (FLOPs) required by the model) which is calculated by a script (`FLOPs_counter.py`) that calculates FLOPs for the baseline model using a sample input.

## Ablation Study Table for Experimentation and Results

| Model(Method) | Base Model | Test Model 1 | Test Model 2 | Test Model 3 | Transfer Learning 1 | Transfer Learning 2 | Transfer Learning 3 | Transfer Learning 4 |
|---|---|---|---|---|---|---|---|---|
| Optimizer | Adam | Adam | Adam | Adam | Adam | Adam | Adam | Adam |
| Loss Function | cross_entropy | cross_entropy | cross_entropy | multi_margin_loss | cross_entropy | cross_entropy | cross_entropy | cross_entropy |
| Learning Rate | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 |
| Test Loss | 5.263 | 3.288 | 3.266 | 0.27 | 0.409 | 0.465 | 0.496 | 0.637 |
| Batch Size | 16 | 32 | 32 | 32 | 16 | 16 | 16 | 16 |
| Max no. of Epochs | 10 | 15 | 15 | 15 | 5 | 5 | 5 | 5 |
| Number of FLOPs | 0.69G | 0.69G | 0.96G | 0.69G | 8.17G | 15.61G | 3.59G | 0.76G |
| Test Accuracy (%) | 35.68 | 43.88 | 43.36 | 38.05 | 96.40 | 96.25 | 95.0 | 93.28 |

| Model(Method) | Base Model | Test Model 1 | Test Model 2 | Test Model 3 | Transfer Learning 1 | Transfer Learning 2 | Transfer Learning 3 | Transfer Learning 4 |
|---|---|---|---|---|---|---|---|---|
| Efficiency Score | 39.74/0.69 = 51.71 | 43.88/0.69 = 63.59 | 43.36/0.96 = 45.17 | 38.05/0.69 = 55.145 | 96.40/8.17 = 11.80 | 96.25/15.61 = 6.166 | 95.0/3.59 = 26.46 | 93.28/0.76 = 122.74 |

## Explanation of the Methods

1. **Test Model 1** –Applied Dropout Regularization with probability 0.2 in layers 3 and 4. Added Vertical Flip and Random Rotation in Data Augmentation, Increased the batch size to 32. The changes were made due to the overfitting of the model, and added a mode of regularization in the layers. Finally trained the model for 15 epochs. The accuracy improved slightly (Loss and accuracy Curve shows potential overfitting and vanishing gradients) and increased the efficiency score which is the maximum in all models.
2. **Test Model 2** – Applied Dropout Regularization and batch normalization to the convolutional layers keeping data augmentation same as the previous model. Also, implemented a wider and deeper model. The overall accuracy improved slightly but the efficiency score reduced because the change in accuracy was very low as compared to change in parameters.
3. **Test Model 3** – Applied multi_margin_loss function with the same model listed in the Test Model 1. The efficiency score improved compared to the base model but less than test model 1.
4. **Transfer Learning 1, 2, 3 and 4** – Applied transfer learning using the resnet50, resnet101 and resnet18 models and EfficientNet-B0 model respectively which are proven to perform well in the ImageNet dataset and most widely used model for multiclass image classification tasks.

## Limitations/Conclusion

The ResNet models offer top accuracy due to their unique approach of using skip connections or "residual blocks" to combat the vanishing gradient issue. Among these, ResNet18 performed best on the test set, albeit with lower efficiency. The compound scaling approach of EfficientNet involves finding the optimal combination of these factors using a compound coefficient. When optimizing for a production system with limited resources, a model with a good efficiency score might be suitable. However, If the task requires high precision, we might need to accept a lower efficiency score for better accuracy. It is a clear choice that favoring both efficiency and accuracy, Transfer Learning 4 is the final model. Moreover, this model can be seamlessly employed across various devices, as it obviates the need to train the entire network anew. Leveraging pre-existing weights results in improved accuracy. Further enhancements can be implemented to bolster efficiency, such as incorporating L1 or L2 regularization, optimizing learning rates and batch sizes, and exploring Ensemble methods.