A Comparative Study of Pre-Trained Image Models for Computer Vision

Saleh Ahmad
BS - AI
National University of Computer and Emerging Sciences
Islamabad, Pakistan
salehahmad2106@gmail.com

***Abstract*—Convolutional neural networks (CNNs) have been shown to be very effective for image classification. However, they can be computationally expensive to train from scratch. Finetuning is a technique that can be used to improve the performance of a pretrained CNN model on a new dataset. In this paper, I compare a CNN trained from scratch and finetune two pretrained CNN models, ResNet50 and EfficientNet, on the Intel Image Classification Dataset. I find that finetuning can significantly improve the performance of pretrained CNN models.**

I. Introduction

CNNs have been the backbone for computer vision since the dawn of imagery and machine learning. CNNs are great problem solvers and they can scale enough to solve problems with varying complexity, but do we need to train a CNN for every problem/dataset? This is where transfer learning and finetuning comes in. Finetuning is basically retraining an already trained i.e. pretrained model on an another dataset.

By leveraging a pre-trained model's knowledge of general features and patterns learned from extensive data, fine-tuning significantly reduces the amount of labeled data required for training, accelerates convergence, and enhances the model's performance on domain-specific tasks. This transfer learning approach saves time, computational resources, and yields more robust and accurate results in various computer vision applications, such as object detection, image classification, and segmentation.

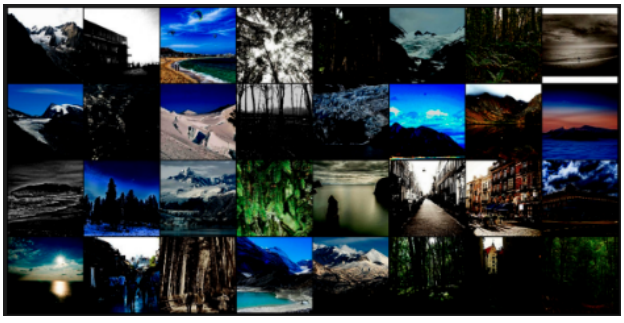

Fig. 1. Dataset Sample Batch

II. Modules and Frameworks

The frameworks and modules used for the paper are:

- Pytorch
- Torchvision
- Numpy
- Matplotlib

III. Dataset Description

The dataset contains around a total of 25000 images. 14000 images in train, 3000 in test and 7000 in prediction. The images are labelled into one of the 6 target labels i.e. buildings, forest, glacier, mountain, sea, street. The images are all of the same 150 x 150 dimensions.

IV. Github Repository

Github repo with the jupyter notebook source files and the dataset: Github

V. Convolutional Neural Network

A. Methodology

The methodology of the provided **Convolutional Neural Network (CNN)** class can be summarized as follows. The network architecture consists of three convolutional layers (conv1, conv2, and conv3) followed by max-pooling layers (pool) to extract hierarchical features from input images. Each convolutional layer uses rectified linear unit (ReLU) activation functions to introduce non-linearity.

After the convolutional and pooling layers, the output is flattened into a one-dimensional tensor and passed through two fully connected layers (fc1 and fc2). The first fully connected layer (fc1) has 256 output features, and the final fully connected layer (fc2) produces the network's output with 6 classes, representing the classification result. The ReLU activation function is also applied to the fully connected layers to introduce non-linearity in the final stage of the network.

This CNN architecture is designed for image classification tasks, with the number of input channels set to 3 (for RGB images) and a specific output shape tailored to the target

classification problem.

The preprocessing includes transformation of images which include resizing images to 150x150, converting to tensor and normalizing the images.

B. Results

Model	Train Acc	Test Acc	Train Loss	Test Loss
CNN	17	18	6.4	1.08

VI. Residual Neural Network (ResNet-50)

A. Methodology

The second model that I utilized was **Wide ResNet-50-2** model from the PyTorch torchvision library as a starting point. This model is a variant of the Wide ResNet-50 model, which has been pre-trained on the ImageNet dataset. The ”2” in the name signifies that the model has roughly double the number of channels in each layer compared to the standard Wide ResNet-50.

To accommodate my specific task, I modified the last layer of the pre-trained model. The original output layer was designed for 1000 classes (the number of classes in ImageNet), but my task required a different number of classes corresponding to our image labels i.e. 6. Therefore, I replaced the last layer with a new fully connected layer with an output size equal to the number of classes in our dataset.

After fine-tuning, I evaluated the performance of our model on a test set from the Intel Image Dataset. The results from this evaluation provide insights into how well my model generalizes to unseen data and its effectiveness in solving our specific task.

The preprocessing includes transformation of images which include resizing images to 150x150, converting to tensor and normalizing the images.

B. Results

Model	Train Acc	Test Acc	Train Loss	Test Loss
CNN	57	73	1.1	0.65

VII. EfficientNet

A. Methodology

The third model that I utilized was **EfficientNet-V2-S** model from the PyTorch torchvision library as the base model for third analysis. The **EfficientNet-V2-S** model is a state-of-the-art convolutional neural network architecture that has been pre-trained on a large dataset, providing us with a robust feature extractor for our image classification task.

I then fine-tuned it on the intel dataset. During the fine-tuning process, the last layer of the pre-trained model, which originally outputted to the default number of classes, was modified to match the number of classes in our dataset. This modification allowed us to repurpose the pre-trained model for my specific classification task.

The performance of the fine-tuned model was then evaluated on the test set from the for accuracy and loss results. The results from this evaluation provide insights into how well our fine-tuned model generalizes to unseen data and its effectiveness in classifying images in the Intel Image Dataset.

The preprocessing includes transformation of images which include resizing images to 150x150, converting to tensor and normalizing the images.

B. Results

Model	Train Acc	Test Acc	Train Loss	Test Loss
CNN	71	80	0.8	0.5

References

[1] O’shea, Keiron, and Ryan Nash. An Introduction to Convolutional Neural Networks. 2015.

[2] He, Kaiming, et al. Deep Residual Learning for Image Recognition. 10 Dec. 2015.

[3] Tan, Mingxing, and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. 2019.