Model Mavericks

Deep Learning Project

10th December 2023

Intro about the chosen topic

Image classification is a common task in Computer Vision and is one of the core fields within Deep Learning. With the emerging trend of using a few major models per domain with Transfer Learning for most tasks raises the question of whether training one's own models from scratch is still a legitimate method or not. In this Project, we attempt to try and get an intuition on what the answer might be.

The project's goal is comparing pre-trained and trained-from-scratch Neural Networks for image classification. First, we surveyed the relevant scientific papers (PyTorch Lightning CIFAR10 ~94% Baseline Tutorial , [1512.03385] Deep Residual Learning for Image Recognition) to get an overview of the topic. Following research, we set up our environments and began to create our Network. After training we set out to validate and measure the final model and ran an evaluation on the Transferred model as well.

CNN model from scratch

We tried to build a simple but powerful and efficient network.

For this, we used only 3 Convolutional and 2 Fully Connected layers. This model has less than 1.2 Million trainable parameters, which should be enough to tackle CIFAR-10.

We could have used AvgPool instead of MaxPool but we thought Max may be more efficient in this scenario: MaxPool can keep more prominent features during its pass, and CIFAR-10's small images mean that averaging can lead the training astray (because it might smooth out relevant, small edges).

Using ResNet18 with 'timm'

The task was to compare multiple approaches of classification. We first tested ResNet18 with the library 'timm'. Our codebase can be found on GitHub.

ResNet18 ([1512.03385] Deep Residual Learning for Image Recognition) is a Convolutional Network made by Microsoft Research. It's 18 layers deep with around 11 million parameters; it was trained on the ImageNet dataset of 1.2 million images of 1000 classes.

Timm (Pytorch Image Models (timm)) stands for 'PyTorch Image Models' and is a PyTorch helper library. It is a "collection of SOTA computer vision models, layers, utilities, optimizers, schedulers, [...] with ability to reproduce ImageNet training results", as per their introduction.

Thanks to this easy-to-use tool, the actual implementation was straightforward, the training itself resulted in an accuracy of around **83%** combined with the augmentations.

Data

To evaluate the models, we chose the well-known CIFAR-10 database. It consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The dataset is split into 50,000 training images and 10,000 test images.

Researchers often use CIFAR-10 as a benchmark; it has around 240 accepted submissions on PapersWithCode. Most SOTA models now use Vision Transformers, shifting away from CNNs entirely. While a ViT vs CNN evaluation would also be interesting, it is outside the scope of this paper.

Methods of training

1. Types of neural network chosen
    a. Both ResNet and our own model are convolutional neural networks. During a pass, the network's input is a Tensor of form NCWH (batch no, Channels, Width, Height).
2. When designing our network, we tried to use well known and tested ideas.
    a. We used 3 Convolutional Layers with a kernel size of 3, going from the input's 3 channels to 128;
    b. We used MaxPool to compress our representation;
    c. After two layers of fully connected layers, we output 10 probabilities for the 10 classes.
    d. We used manual HyperParameter optimization, mainly adjusting the learning rate.

Define CNN

```
class CustomCNN(nn.Module):
    def __init__(self):
        super(CustomCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(128 * 4 * 4, 512)
        self.fc2 = nn.Linear(512, 10)
```

Evaluation (results on test data, including visualization)

During our tests, ResNet18 produced 86% and our own model 83%.

ResNet18 converged faster on its accuracy, demonstrating that it's only finetuning what it already learned. This can be seen in its loss values approaching zero faster.

Our net's slower convergence and overall lower accuracy can be attributed to it being much smaller (a tenth of Resnet18), and, of course, training from randomly initiated weights.

Conclusions (what is the main result of your project)

During the Project, we explored various ways of doing Computer Vision with Deep Learning. Our original assumption and intuition of large, well established models dominating most Deep Learning subfields over from-scratch solutions was vindicated. Even over the top augmentations can't change that.

The method of from-scratch developments should thus be used for research, prototyping, or any other domains where well-established large models are unavailable for some reason (new or hard domain where no public work was done, Edge Computing etc).

Our Team

Hegedűs András	BCFU8E
Lakatos Bálint Gábor	T04ZVM
Sárossy János	GUSXLY