Final Project Report transfer learning 2023

Md Esti
University of Vaasa
Wolffintie 32, 65200 Vaasa
md.esti@student.oulu.fi

Abstract

This deep learning project primarily focuses on using the transfer learning method in image classification, where the target goal is to improve performance in remote sensing applications with small datasets via transfer learning. In this project, certain procedures have been used to deal with the outcome or goal. Certain methods such as splitting datasets or organizing datasets, preprocessing, normalization, and using pre-trained models have been used throughout the project to reach the required outcome of the project. Furthermore, it also includes pretraining the model on the miniImageNet dataset and then fine-tuning it on the selective or updated portion EuroSAT dataset.

Finally, going through deep analysis, this project achieves its goal in applying transfer learning to the EuroSAT dataset.

1. Introduction

In this modern era, a revolutionary concept has arrived that uses pre-built complex algorithms to understand the variations of patterns in data called deep learning. It makes the computer understand instructions that were previously impossible. Moreover, it also uses neural networks with multiple layers to model complex patterns in data.

It should also be noted that in plenty of scenarios deep learning models become relatively insignificant to further work with. In situations where the data is finite the model only learns within the boundaries of trained data and fails to deal with data that is unseen to the model. Ultimately the model fails to deal with sudden new and unseen data. Now coming to the primary objective of the project which is to deal with the complexity, applying transfer learning to the project becomes a good initiative overall. The concept of this transfer learning method involves a certain scenario where developing a model for one or plenty of tasks can be reused whenever it is necessary for that might come in the future. As we know datasets are always finite thus it is almost impossible to deal with infinite seen data which transfers the knowledge achieved from a larger, related dataset to a smaller one. Finally, it improves performance and also reduces the need for large data collection making the deep learning model more efficient and flexible to deal with unseen data. [1]

2. Approach

There are few approaches throughout the project which have been followed according to the given requirement of the project.

2.1 First step

Organizing datasets:
First, Collect the dataset miniImageNet from the source drive link as instructed. Extract the dataset miniImageNet in the local computer. There are three folders train,val, and test under the miniImageNet folder but our objective is to use only train.zip and split it into three folders train,val, test, and name the source directory to To miniImageNet. Secondly, selecting 100 images from EuroSAT dataset, from 5 different categories, and each category includes 20 samples. Then randomly choosing 25 images from these 100 samples as training set (The 25 images are from the 5 different categories where each category includes 5 images). Finally, Upload the updated datasets to the personal Google Drive folder under the directory below. **'/content/drive/MyDrive/project/miniImageNet'.**

2.2 Second step

Mount Google Drive:
Mount Google Drive or connect Google Drive with colab environment using the following command.

```
from google.colab import drive
drive.mount('/content/drive')
```

2.3 Third Step

Importing necessary libraries:
Before working with extensive datasets, pytorch, etc importing all the libraries.

```
import os
import torch
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
from torchvision import models
import torch.nn as nn
import torch.optim as optim
```

After importing the libraries proceed with the project generation work.

3. Experiments

Preprocessing Steps:
The datasets have been well-preprocessed with standard transformations, normalization, and tensor conversion.

```
transform = transforms.Compose([
transforms.Resize((224, 224)),  # Resizing images to fit ResNet18
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]).
```

Selecting the model:
For the project selecting the model Rest18 as it is efficient in terms of the current scenario.

- Define the model.
- Pretrain the model on the dataset miniImageNet.
- Train the model.
- Select the number of Epochs.
- Finally, save the model.

```
project_folder = '/content/drive/MyDrive/project'
model_save_path = os.path.join(project_folder, 'resnet18_miniImageNet.pth')
torch.save(model.state_dict(), model_save_path)
```

The results of the saved model is given below :

```
Epoch 1/5, Loss: 1.3128431661219537
Epoch 2/5, Loss: 0.8590132582791244
Epoch 3/5, Loss: 0.610519960220856
Epoch 4/5, Loss: 0.4343314125002185
Epoch 5/5, Loss: 0.32685659160908265
Model saved to: /content/drive/MyDrive/project/resnet18_miniImageNet.pth
```

So, the accuracy rate of the saved model is around 68 percent.

Fine-tuning the saved model on the datasets of EuroSat :
At first the datasets have been properly sectioned according to the project requirement, then the custom datasets have been instantiated. After that my objective was to apply fine-tuning the saved model on the datasets of EuroSat using the function :
fine_tune_model (model, criterion, optimizer, num_epochs=5).
after fine-tuning the model, setting the model to evaluate mode. Finally, the model can adjust to new kind of tasks even though the model was previously trained on different kind of datasets.

```
model_save_path = '/content/drive/My Drive/project/resnet18_EuroSAT.pth'
torch.save(model.state_dict(), model_save_path)
print(f"Fine-tuned model saved to: {model_save_path}")
```

```
Epoch 1/5, Loss: 1.8507752418518066
Epoch 2/5, Loss: 0.757851779460907
Epoch 3/5, Loss: 0.34761515259742737
Epoch 4/5, Loss: 0.14797437191009521
Epoch 5/5, Loss: 0.061268970370292664
Accuracy of the model on the test images: 86.66666666666667%
Fine-tuned model saved to: /content/drive/My Drive/project/resnet18_EuroSAT.pth
```

Latest Accuracy of the model has increased more than before which is 86.6 percent accuracy rate which proves the effectiveness of transfer learning.

4. Conclusion

This project clarifies the significant usage of transfer learning in certain situations where the data is limited. Pretraining the miniImageNet dataset and fine-tuning it on EuroSAT is a clear application of transfer learning. As it is seen the initial accuracy rate was decent but not enough to use it for real-life purposes of remote sensing. So, for further enhancement transfer learning approach has been done. In conclusion, it is understandable that certain scenarios might occur where it is necessary to apply the concept of transfer learning to achieve the maximum possible outcome on remote sensing applications.

References:

[1] https://www.tensorflow.org/tutorials/images/transfer_learning