Transfer Learning for Land Cover Classification: A Deep Learning Approach with EuroSAT Dataset

[*]Download Project : Google Drive link

Dumindu Udawatta
University of Oulu
Oulu, Finland
dumindu.udawatta@student.oulu.fi

Abstract

This paper presents a study on enhancing classification performance of land cover imagery using EuroSAT dataset. Advanced deep learning strategies have been used to improve the classification performance of the dataset. The research employs the popular transfer learning techniques to achieve higher accuracies. Popular and successful model architecture like Resnet-18 is considered for transfer learning. Models were pre-trained on miniImageNet to achieve higher performances by optimizing its parameters through rigorous experimentation and popular hyperparameter tuning techniques. Performance of the model is considerably higher and performs really well in identifying Highway, AnnualCrop, Industrial, PermanentCrop and River SAT images. In a nutshell the study contributes to providing a robust deep learning model to classify land cover images, providing the ability to analyze the satellite images. The findings may help to advance the understanding of remote sensing applications but also underscore the importance of leveraging advanced techniques for precise land cover monitoring in environmental and agricultural contexts.

1. Introduction

Deep learning is a machine learning approach inspired by the human brain functionality. Human brain has billions of neurons communicating with each other to process information. Similarly, deep learning algorithms have artificial neurons called nodes. Human brain can easily identify complex patterns like images, sounds, text and other data. Deep learning can be used to train a computer to identify and process those complex data which is used to build digital assistants, fraud detection and facial detection applications, self driving cars, robots and in many other applications. Mainly deep learning applications are divided into four categories called, computer vision, speech recognition, natural language processing and recommendation systems.

Deep learning networks usually consist of 3 main layers. Input layer has nodes with input data. Input layer passes the information into hidden layers which consists of multiple levels. A typical deep learning network may consist of hundreds or thousands of these hidden layers and these layers adapt their behavior according to the data they receive. Large number of layers helps to analyze the data thoroughly and process it in different angles to find new information.

Deep learning networks have multiple benefits over traditional machine learning(ML) approaches. More often than not we encounter a lot of unstructured data. Deep learning (DL) models can process these types of data efficiently where traditional ML models struggle to generalize due to the infinite variations of those data. The ability to handle large amounts of data allows DL models to find hidden relationships and patterns in the data. DL models may reveal new insights on angles the developer initially did not even anticipate. Furthermore, DL models can learn and improve over time based on user behavior. Finally, the ability to handle a volatile dataset is also an advantage of DL.

However, everything comes at a cost. Deep learning algorithms are typically compute intensive and require a proper infrastructure with enough capacity to properly function. Otherwise, it may even take days, weeks or even months to produce a result that is desirable. DL algorithms usually perform much better with large quantities of high quality data to produce reliable results. In a nutshell, deep learning algorithms usually have a huge cost involved with it.

While deep learning methods have proven to be successful in many applications and scenarios in the real world there are situations that fall short. As an instance, when the dataset is limited and has fewer labeled data DL algorithms overfits the data due to its architecture. In some cases a simple traditional model will perform much faster when the association between the input and outputs are clear. In such cases, resources and time will be wasted on training a DL model which might provide subpar performance anyway.

Transfer learning is an advanced technique in deep learning to mitigate one of its weaknesses which is requiring millions of data to train a model to obtain accurate results. Transfer learning uses a pretrained model which has been already trained on millions of data beforehand and weights calculated. In transfer learning the developer tweaks the model architecture a little by changing its output layer such that it is suitable for the new application. This exploitation of the knowledge gained from previous tasks helps to generalize the new task faster. For example, in training a classifier may classify images of trees, a developer can use it to generalize it to recognize cars.

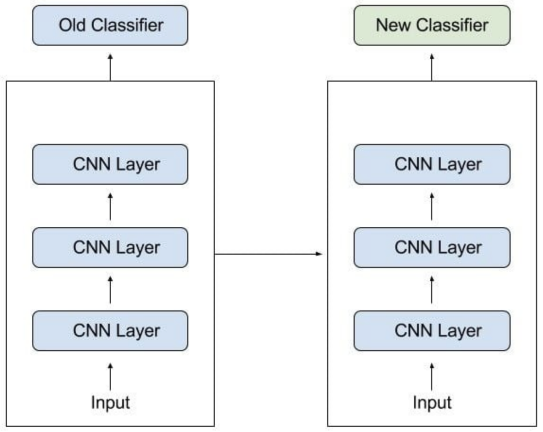


Figure 1. Transfer Learning

Transfer learning shines the most when the sample size is limited in size or the developer may not have sufficient computing power to train a huge dataset beforehand. Apart from that the time consumed to train a model is significantly lower and the rewards are high. Even with a small sample size the pretrained model only adjusts its weights according to new data. Therefore by adjusting the output layer it is possible to achieve very high accuracies with very low cost. Transfer learning prevents training models for each and every generalization.

This study is focused to enhance the classification performance on land cover images using EuroSAT dataset. Resnet-18 is a popular model included in Pytorch for image classification. This is a convolutional neural network which is 18 layers deep. More than million images from ImageNet are used to train the model. We employ this model and train it using miniImageNet dataset. Finally adjusting the output layer to classify satellite images for land cover. A data loader is created to feed the data into the model. Hyperparameter tuning has been done manually and evaluated using the accuracy on the test dataset. During the hyperparameter tuning process a different optimizer has been implemented and tested to achieve the best accuracy possible.

2. Approach

The study aims to improve the classification performance on the EuroSAT dataset by using modern hyperparameter tuning like data augmentation.

2.1. Data preparation

As the first step, required datasets have been acquired and prepared for further processing.

1 miniImageNet dataset.

miniImageNet is a Few-Shot dataset which is a subset of ImageNet dataset. This is specifically designed for academic purposes so the students can handle the dataset more conveniently. Dataset is publicly available to download. After downloading the dataset, a Dataset class is created to handle the dataset when training, validating and testing the model.SInce, this dataset has different categories in train.tar, val.tar and test.at collections, only the train.tar collection is selected for further processing. Train collection is divided further into three subsets for train, validation and testing purposes. A data loader is also introduced to provide efficient data loading. Employing the data loader, multiple benefits were added to the project implementation using a data loader such as parallel data loading, data transformation, efficient data loading, batching, shuffling and considerable memory efficiency compared to the manual loading of the data.

2 EuroSAT dataset

The EuroSAT dataset includes images for land use or land cover data. This is also publicly available to download and use. There are two variations of the dataset available to download. In this study I only focus on the RGB dataset. After downloading, EuroSAT class was created to handle the dataset for further processing. Similarly in miniImageNet, data loaders were used to train the model.

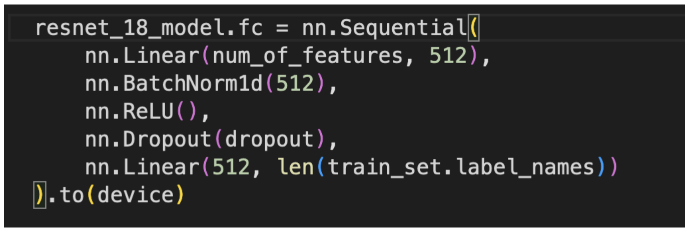
2.2. Pretrain a model

In this step, the goal is to train a model with miniImageNet data to create the pretrained model. Firstly, the Resnet-18 model is downloaded with default weights. There are two separate weights classes provided by Pytorch named ResNet18_Weights.IMAGENET1K_V1, ResNet18_Weights.IMAGENET1K_V2 (or ResNet18_Weights.DEFAULT) where according the Pytorch documentation has 76.13% and 80.858% accuracies. Output layer of the model has been updated

Parameter	Value
Cost function	`nn.CrossEntropyLoss()`
Optimizer	Stochastic gradient descent (SGD)
Learning rate scheduler	StepLR
Num of epochs	8
Dropout	0.5
Starting learning rate	0.001
Momentum	0.5
Weight decay	1×10^{-2}

Table 1. Hyperparameters for the Neural Network Model.

according to the miniImageNet dataset. Selected output layer architecture is as follows.



```
resnet_18_model.fc = nn.Sequential(
    nn.Linear(num_of_features, 512),
    nn.BatchNorm1d(512),
    nn.ReLU(),
    nn.Dropout(dropout),
    nn.Linear(512, len(train_set.label_names))
).to(device)
```

Figure 2. pretrained model output layer

Multiple strategies have been tested out to determine the best possible parameters for the model (see Experiments section). Following table includes all of the final parameters selected in the model training.

While improving the model performance Pytorch documentation suggested to include the image preprocessing beforehand. Using the data loader, these transformations have been implemented and this helped to produce higher accuracies. This preprocessing includes following data augmentation techniques.

1 Accepts Image Objects:

- The preprocessing module is designed to handle input in the form of both single image tensors (C, H, W) and batched image tensors (B, C, H, W).
- Supported input formats include `PIL.Image` and `torch.Tensor` objects.

2 Image Resizing:

- Images undergo resizing to a standardized size of 256 pixels using bilinear interpolation (`InterpolationMode.BILINEAR`).
- The purpose of resizing is to ensure uniformity and compatibility in subsequent processing steps.

3 Central Crop:

- Following resizing, a central crop is applied to the images.
- The crop size is set to 224 pixels, providing a focus on the central region of the image.

4 Value Rescaling:

- Pixel values are rescaled to the range $[0.0, 1.0]$.
- This normalization step ensures consistency in the data representation across different images.

5 Normalization:

- The final step involves normalization using predefined mean and standard deviation values.
- Mean values: $[0.485, 0.456, 0.406]$
- Standard deviation values: $[0.229, 0.224, 0.225]$
- Normalization helps standardize the pixel intensity values, contributing to improved model training and performance.

The model is trained and accuracies have been calculated. Finally the full model is saved to the disk.

2.3. Train the model for the EuroSAT dataset.

The task description stated to select only 5 image categories at once even though EuroSAT has 10 image categories. Therefore, randomly selected 5 categories used to train the model. Multiple experiments have been carried out to include all the categories into the model by five categories randomly at once. The Experiments section will explain the details on this matter. Finally, 5 randomly selected classes used the train and evaluated the model. Thise selected classes are,

1 Highway

2 AnnualCrop

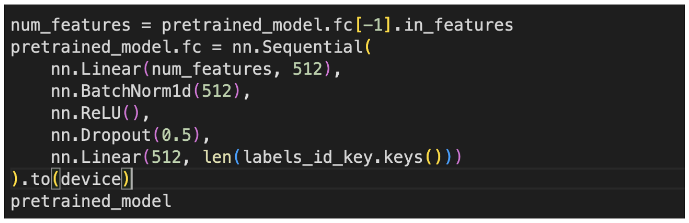
3 Industrial

4 PermanentCrop

5 River

In this step the pretrained model is loaded from the previous step. The output layer has been modified again to accompany the EuroSAT dataset. Output layer outputs only 5 categories according to the selected output layer architecture.

Following parameters have been used to train the new model. As previously, preprocessing techniques suggested by PyTorch documentation also applied here to achieve



```
num_features = pretrained_model.fc[-1].in_features
pretrained_model.fc = nn.Sequential(
    nn.Linear(num_features, 512),
    nn.BatchNorm1d(512),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(512, len(labels_id_key.keys()))
).to(device)
pretrained_model
```

Figure 3. EuroSAT model output layer

Parameter	Value
Cost function	`nn.CrossEntropyLoss()`
Optimizer	Stochastic gradient descent (SGD)
Learning rate scheduler	StepLR
Number of epochs	50
Number of rounds	30
Dropout	0.5
Starting learning rate	0.001
Momentum	0.9
Weight decay	0

Table 2. Hyperparameters for the Neural Network Model.

Parameter	Value
Total data	38400
Train data	23040
Validation data	7680
Test data	7680
Number of classes	64

Table 3. Hyperparameters and Data Split for the Neural Network Model.

higher testing accuracies. In this case the model is trained in multiple rounds. Each round consists of a random number of images selected from the dataset. Model training is carried out several times to improve the generalization performance.

3. Experiments

3.1. Datasets

miniImageNet and EuroSAT_RGB are the datasets used in this study. As explained previously, miniImageNet dataset is a few-shot dataset which means it is optimized for academic usage. However, due to the multiple categories used in train.tar, val.tar and test.tar, only the train.tar is selected in this study for simplification. Train.tar is divided into 6:2:2 ratios to obtain train, test and validation datasets.

Each class assigned a number manually before proceeding the training. Following image displays random images

Parameter	Value
Total data	100
Train data	25
Test data	75
Number of classes	5

Table 4. Hyperparameters and Data Split for the Neural Network Model.

after preprocessing techniques are applied.

Figure 4. Sample miniImageNet data after preprocessing

EuroSAT dataset consists of overall 10 categories. However, according to the task description this study only focuses on five pre-selected categories. Initially the study tried to incorporate all of the categories by training five categories randomly at a time with an adjusted output layer with 10 categories unlike in Figure 3. However, this experiment was not successful since the model tends to overfit to the initial 5 categories. First iteration produces very good accuracies around 80% and in the sub sequence iterations the performance dropped to around 20% or less. Therefore, it is decided to only stick with five categories and continue producing accurate results for selected categories.

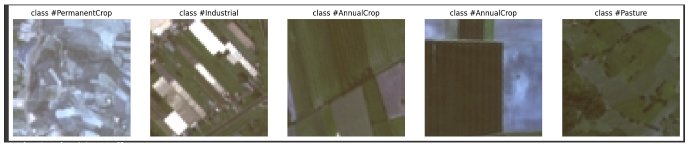


Figure 5. EuroSAT sample data after preprocessing

3.2. Model Training

3.2.1 PreTrain Model using miniImageNet data

Initially a pretrained model has been trained using the Resnet-18 model weights. Following steps taken into account when tuning the hyperparameters.

Following values have been selected as default. In each experiment we change one parameter and evaluate the model performance. Parameters that give higher performances are kept the carried into the next experiment.

1 Experiment 1: Dropout

Parameter	Value
Epochs	10
Dropout	0.5
Optimizer	SGD
Learning rate	0.001
Momentum	0.9
Weight decay	0
Data transformation	Default (ResNet-18)

Table 5. Hyperparameters and Data Split for the Neural Network Model.

Value	Accuracy (%)
0.1	80.23
0.5	79.21
0.8	78.09
0.9	72.73

Table 6. Dropout Experiments and Model Performance.

Value	Accuracy (%)
1×10^{-5}	79.69
1×10^{-2}	76.84

Table 7. Weight Decay Experiments and Model Performance.

2 Experiment 2: Learning Rate

- Learning Rate Scheduler: Increased every five steps up to 0.1 times.
- Initial Learning Rate: 0.0001

3 Experiment 3: Weight Decay

See the table 7 for details.

4 Experiment 4: Momentum
Reduced momentum to 0.5, achieved accuracy: 81.61%.

5 Experiment 5: Adam Optimizer
Adam optimizer achieved accuracy of 36.07%.

6 Experiment 6: Data Augmentation
Data Augmentation Techniques:

- RandomHorizontalFlip
- ColorJitter (brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2)
- RandomRotation (degrees=15)

Final accuracy: 63.16%.

3.2.2 EuroSAT model

In this step, the pretrained model from the previous step is loaded into and only changed the output layer and trained with a very few number of samples. Model is trained multiple times with different data selected randomly each time to improve its performance. Number of epochs has been selected as 50 and the number of rounds have been selected as 30. This allows the model to generalize the data much better. There was a concern that using this many iterations might overfit the model. However, the obtained testing accuracy is 76.04%. Therefore I can confidently declare that the model is performing well for a new dataset.

3.3. Results and analysis

After conducting the above experiments, the trained model has an final accuracy of 76.04%

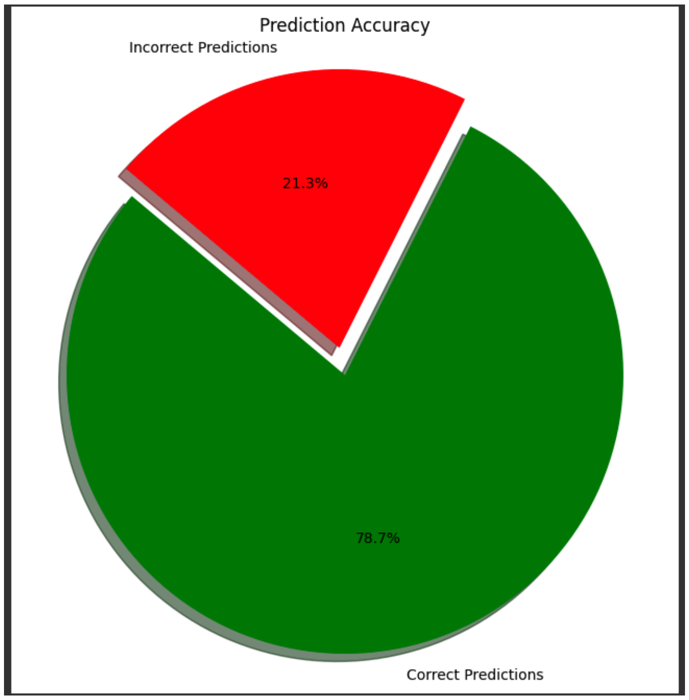


Figure 6. Prediction Accuracy

Figure 7 is a display of the percentage of correct predictions per class. it is observed that most of the classes do really well and the model is able to predict the class accurately. However the Highway class is an exception to this claim.

4. Conclusion

Deep learning is a important machine learning aspect which allows the developers to do the classifications and human like tasks that traditional machine learning failed to

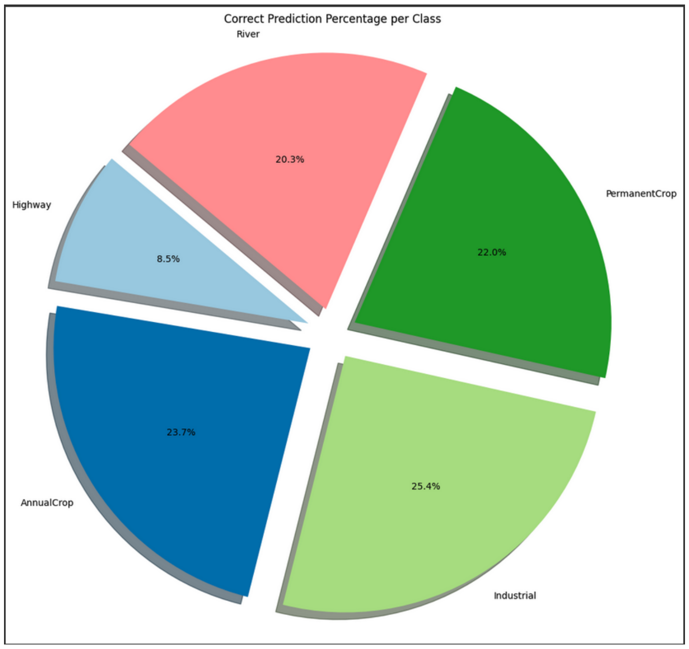


Figure 7. Class wise prediction accuracy

do. Transfer learning is advancement of machine learning which helps the developers to train high quality models with a minimum cost. In this study I try to enhance the classification accuracy of EuroSAT sattelite image dataset which includes images of land cover using transfer learning strategy. Resnet-18 model is employed to pretrain a model using few-shot miniImageNet dataset. Using the pretrained model another model is trained with a minimal set of data. I was able to obtain a 76% accuracy which is considerably good. It is evident from the visualizations that the model performs well with almost all of the classes except Highway class.

References

[1] Niklas Donges. What Is Transfer Learning? Exploring the Popular Deep Learning Approach. [Link]

[2] Amazon.com. What is Deep Learning?. [Deep Learning on AWS]

[3] PyTorch. Documentation. [PyTorch Vision Models

[4] Sajid ,Iqbal. A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks. [Original ResNet-18 Architecture on ResearchGate]