Optimizing Transfer Learning for Efficient Image Classification in Remote Sensing Applications

Mohammad Azizul Kawser
University of Oulu
azizul.kawser@student.oulu.fi

Talha Zeeshan
University of Oulu
Talha.Zeeshan@student.oulu.fi

Abstract

This report examines the optimization of transfer learning for image classification in remote sensing applications. It focuses on the benefits of transfer learning in addressing challenges faced due to limited labeled datasets in remote sensing and highlights the use of deep learning and neural networks. The report compares the performances of ResNet-18 and VGG-16 on the miniImagenet and EuroSAT-RGB datasets, explains their architectures and differing approaches. Additionally, the report discusses the datasets, training setups, results, and analysis, emphasizing the significance of techniques such as regularization, drop-out, and fine-tuning hyper-parameters in improving the machine learning models. The findings in the report underline the potential of transfer learning using ResNet-18 and VGG-16 in enhancing image classification and highlights the performance of ResNet-18 over VGG-16 by providing empirical evidence.

The Colab notebook: Code Link

All Trained Project Models: Trained Models

1. Introduction

Deep learning is a subset of machine learning, a branch of artificial intelligence that mimics humans' learning approach to gaining knowledge [4]. It is characterized by its use of neural networks with many layers, hence the term "deep" implied here [6]. These deep neural networks enable the model to learn complex patterns from large amounts of data [18]. This capability has propelled innovations in fields like image and speech recognition, natural language processing, and autonomous driving. By leveraging neural networks with numerous layers, deep learning autonomously learns characteristics and their hierarchical representation, outperforming traditional machine learning in various fields. Deep learning models, such as convolutional neural networks (CNNs) for images and recurrent neural networks (RNNs) for sequences, automatically learn relevant features from the data, eliminating the need for manual feature extraction [24]. These advancements have transformed across diverse sectors such as healthcare, cybersecurity, IoT and Smart Cities, Agriculture, Business, financial services, consumer electronics, etc., significantly enhancing machine capabilities in tasks once deemed complex or unattainable, thus driving forward AI-driven technological innovations [19]. Although deep learning has made significant progress, basic deep learning methods may still fall short in certain scenarios due to challenges such as the need for vast amounts of unbiased training data. Additionally, these methods might not perform well with unstructured or complex data, which requires more sophisticated or specialized deep learning models for effective processing. There is a high computational cost associated with building models from scratch, and difficulties in generalizing to situations or data they haven't encountered before are also challenging. Furthermore, deep learning models with many hidden layers make it hard to interpret their decision-making processes. This "black box" nature is problematic in fields requiring transparency, such as the financial and healthcare sectors. Overfitting is another issue, where models learn to replicate the training data too closely and fail to perform well on new data [1, 8, 9].

Transfer learning is a technique that addresses some of the challenges in deep learning, where a model developed for one task is reused as the starting point for a model on a second task [5]. This approach is based on the principle that human beings can apply previously learned knowledge to solve new problems more effectively and efficiently. It is also known as knowledge transfer, a pre-training method that utilizes knowledge from previously trained data to construct new models. It is particularly useful in scenarios where there is a limited amount of data available for the second task. In traditional machine learning, models are most effective when the training and testing data share the same feature space and distribution. However, this is often not possible in many real-world scenarios, leading to the need to rebuild models from scratch with new data, which can be expensive and sometimes unfeasible. Transfer learning ad-

dresses this by transferring knowledge and applying it to a different but related problem from a domain similar to the target domain, thereby reducing the need and cost of collecting new data. Transfer learning enhances the learning process and performance by transferring information from a previously trained model, reducing both the computational cost and the need for large amounts of training data [11, 23]. The training process in transfer learning involves initially training a baseline model on a large dataset and then fine-tuning it using a smaller target dataset to achieve optimal performance. This method contrasts with directly training a model on a limited target dataset. Empirical evidence shows that transfer learning approaches effectively combat overfitting, expedite training, and significantly reduce the dataset size required for retraining [27]. By reusing features learned from related tasks, transfer learning enhances model performance without requiring extensive data typically necessary for training deep learning models from scratch [19]. This form of transfer learning is called "inductive transfer". Figure 1 displays how this form of learning in deep networks the scope of possible models is narrowed in a way that the model is fit for a different but related task [11]

The motivation behind using transfer learning for image classification in this project is to leverage the capability of pre-trained models on large, diverse datasets fine-tuned for a specific task. The pre-trained model provides a foundational understanding of image features, such as edges, textures, and patterns [22]. By fine-tuning these models on a specific task, transfer learning enables effective learning and generalization, making it a practical solution for image classification tasks with data constraints [28]. We want to enhance image classification performance in remote sensing applications when dealing with small datasets. The strategy involves initially training deep learning models, such as ResNet and VGG, on the miniImageNet dataset [26]. This preliminary training allows the models to learn a range of features and patterns that can be generalized. Once these models are pre-trained, they are ready to be fine-tuned using a much smaller subset of images from the EuroSAT (RGB) dataset [16], which is representative of the specific remote sensing tasks. By using transfer learning in this manner, the project effectively overcomes the typical data scarcity challenges in remote sensing applications, enabling the models to achieve higher accuracy and better performance.

2. Approach

Remote sensing is the science of obtaining information about objects or areas from a distance, typically using satellite or aerial imagery. It has a vast array of applications, from environmental monitoring to urban planning [13, 25]. However, one of the primary challenges in remote sensing is the limited availability of labeled datasets. Unlike domains where data can be abundantly collected, labeled

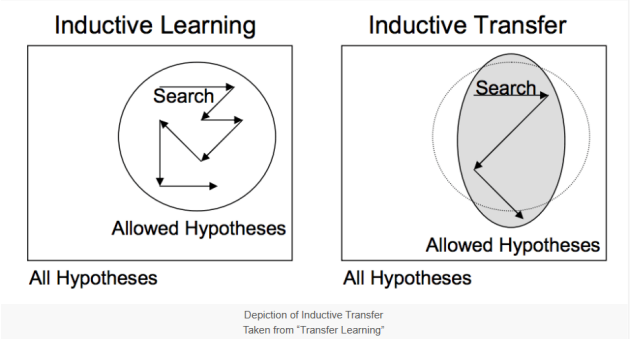

Figure 1. Inductive Learning [11]

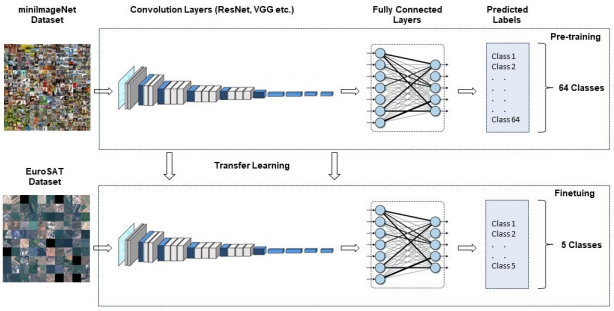

Figure 2. Visual diagram of Transfer learning from miniImageNet.

datasets in remote sensing are scarce due to the high costs and logistical complexities involved in gathering and annotating this data [7]. Deep learning has already shown its efficacy in image classification in manufacturing, healthcare, agriculture, transportation, sports, etc areas, offering models that can extract and learn complex patterns and features from images [10]. Transfer learning has emerged as an effective solution to this problem. In the context of this project, the concept is applied to train models on a large, diverse dataset (miniImageNet) and then adapt them to a specific, smaller dataset (EuroSAT) relevant to remote sensing. For this project, we are going to compare the performances of ResNet-18 and VGG-16 models on the provided data-sets. Both models utilized transfer learning where each was trained on the miniImageNet dataset and the resulting model was then used as the starting point for training on the EuroSAT-RGB dataset. The visual diagram of the approach is illustrated in Figure 2. We will also explain the architecture of each model (Section 2.6, 2.7) in detail and describe what they entail aswell as the approach utilized for preprocessing the provided datasets (Section 3.1)

The detailed steps involved in this process are outlined below:

2.1. Step 1: Dataset Preparation

We obtained the miniImageNet from the provided repository [2], which serves as the basis for pre-training the models. As instructed, we only focused on the train.tar dataset and used it as our primary dataset. Then we split the train.tar dataset into further training, validation, and test sets according to specified ratios of 70%, 15%, and 15%. The images within each class folder were shuffled and then copied into new folders for training, validation, and test sets to prevent biases that might arise if the data is ordered. Additionally, we obtained the EuroSAT (RGB) dataset from the provided repository [16], which serves as the target dataset. We randomly chose 100 images from the EuroSAT dataset, representing random 5 different categories, with each category including 20 samples. To ensure balanced representation across categories, we randomly selected 25 images from these 100 samples as the training set rest 75 images as a testing set. Detailed descriptions of these datasets are provided in the dataset section of this report.

2.2. Step 2: Model Selection and Pretraining

We chose the ResNet-18 and VGG-16 models, as both are pre-trained on large datasets like ImageNet. The version of the weights loaded corresponds to the 'ImageNet-1K' dataset. This means they have already learned a lot of features from a huge variety of images, making them very effective for training on the miniImageNet dataset. This gives the model a head start in learning useful representations, which can be more effective than starting from a randomly initialized model. Later, we train the selected models on the miniImageNet dataset. Although the miniImageNet dataset is smaller in scale compared to ImageNet, during this training phase on miniImageNet, the models undergo a process of adaptation. The pre-learned weights are fine-tuned to better suit the specific characteristics of the images in miniImageNet. This step is crucial, as it allows the models to shift their focus from the general features learned from ImageNet to more dataset-specific features relevant to miniImageNet.

2.3. Step 3: Saving the pre-trained Models

After achieving satisfactory performance on the miniImageNet dataset, we saved the model's parameters (weights and biases), which constitute the core components enabling the model to make predictions or classifications.

2.4. Step 4: Fine-Tuning with EuroSAT Dataset

For fine-tuning the model with the EuroSAT dataset, we begin by loading the saved models along with their pre-trained weights. This step is crucial as it leverages the foundational learning the models have already achieved. Following this, we adjust the training parameters as necessary. These adjustments are key in aligning the model's pre-existing knowledge with the specific characteristics and requirements of the EuroSAT dataset, ensuring the model is well-tuned for optimal performance on this particular task.

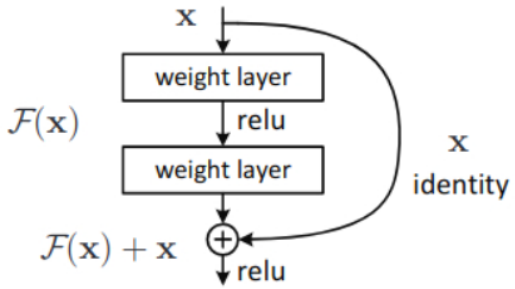

Figure 3. Residual Block: a building block [15]

2.5. Step 5: Evaluation and Testing

In this final step, we focus on evaluating the model's performance on the testing set. This step provides us with insights into how well the model has adapted to the EuroSAT dataset through the fine-tuning process. The model's accuracy on the validation set is calculated by comparing the predicted and actual labels.

2.6. ResNet18

Residual Training Network (ResNet) is a novel approach proposed by researchers at Microsoft to address the problems and limitations of deep neural networks described previously. The degradation problem along with increased complexity of ever deeper neural networks provided a significant stumbling block towards more powerful image classification models [15]. Resnet attempts to solve this problem by introducing a concept called *Residual Block*.

Figure 3 showcases a structure of the residual block that forms the basis for the ResNet architecture. The residual block is the building block which utilizes a technique called *skip connection*. Skip connection is a technique which allows us to connect activations of a layer to further layers in the network by skipping some layers in between [15]. The stacking of these residual blocks allows for the formation of a ResNet model.

Figure 4 displays the structure of the resnet18 model that has been utilized in this project. The network takes in an image of input size [224 x 224]. The first convolution layer is of size [7 x 7], kernel size 64 with a stride of 2 which is followed by a max-pooling operation. The ResNet-18 model consists of four residual blocks which allow the gradient to be directly back-propagated. Each of the four residual blocks have a constant number of channels [64, 128, 256, 512] as shown in Figure 5 of ResNet-18 architecture. In the case that the dimensions of input and output are differ-

layer name	output size	18-layer
conv1	112×112	
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$
	1×1	
FLOPs		1.8×10^9

Figure 4. ResNet18 Layers [15]

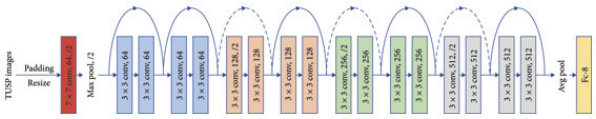

Figure 5. ResNet18 Structure [14]

ent, a [1x1] convolution with a stride of 2 is used. In the final layer of the network, there is a fully connected layer followed by a softmax function for outputting probablities. This final layer can be fine-tuned to match the desired number of output classes. For this project, the final layer was adjusted to output 64 classes for the miniImageNet dataset and 5 classes for the EuroSAT-RGB dataset.

2.6.1 Mathematics

This section of the report summarizes the mathematics behind the concept of the residual block and how it is utilized to build the ResNet model for image classification.

Assuming x denotes input, In a normal neural network, the input is mapped to the intermediate representation and then to y by the following equation:

$$y = F(x) \tag{1}$$

However, as can be seen Figure 3, the residual block has 2 layers and the input to the second layer is the output of the first layer as well as the input to the first layer. This extra input is called *identity mapping*. Hence, equation 1 changes to the following to represent the block:

$$y = F(x) + x \tag{2}$$

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Figure 6. VGG-16 Layers [3]

Throughout the course, we have studied that the function F(x) signifies the multiplication of weights (W) with input x, hence we can substitute the function F(x) to signify the multiplication of weight W with our input:

$$y = Wx + x \tag{3}$$

The residual block has 2 layers and a ReLU non-linearity mapping applied before the second layer aswell as after the identity mapping. This makes it so that we have two sets of weight vectors W(1) and W(2) for the two layers. Hence our equation becomes (σ denotes the ReLU non-linearity):

$$y = W_2\sigma(W_1x) + x \tag{4}$$

2.7. VGG-16

VGG-16, developed by the Visual Graphics Group at Oxford, is a deep convolutional neural network known for its simplicity and depth. It's characterized by its use of small (3x3) convolution filters throughout the network, allowing it to learn complex features at various scales. Figure 6 displays the structure of the VGG-16 model that has been utilized in this project. It takes an input image of size 224x224 pixels. The network consists of 13 convolutional layers, using rectified linear unit (ReLU) activations. The convolution layers are organized into five blocks and each followed by 5 max-pooling layers to reduce dimensions. The convolutional layers increase in depth from 64 filters up to 512 as the network progresses. Following the convolutional layers, there are three fully connected layers, the first two with 4096 channels each, and the last one with 1000 channels which corresponds to the number of output classes in the ImageNet challenge [21]. The last fully connected layer of the model is replaced to match the number of classes in our datasets. For this project, the final layer was adjusted to output 64 classes for the pretraining miniImageNet dataset and 5 classes for the finetuning EuroSAT-RGB dataset. Figure 7 displays the complete, connected

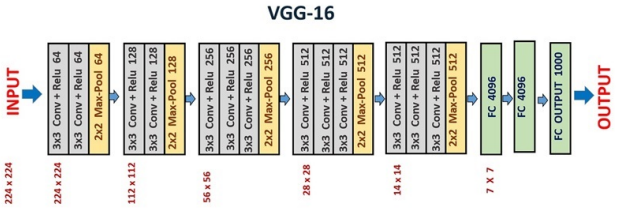

Figure 7. VGG-16 Architecture [3]

VGG-16 architecture utilized in this project. The output is a softmax activation that yields the probabilities of each class.

2.7.1 Mathematics

The mathematics behind VGG-16 is less complex than ResNet's because it doesn't involve skip connections or residual mappings. It can be described by the repeated sequential application of convolution operations to the input image, followed by a ReLU non-linearity. Given input x, VGG-16 applies a series of convolutional layers with filters $W^{(i)}$, each followed by a non-linear ReLU function σ. The process can be summarized as:

$$y^{(i)} = \sigma(W^{(i)} * x^{(i-1)} + b^{(i)}) \tag{5}$$

Here, $y^{(i)}$ is the output after the i^{th} convolutional layer, $W^{(i)}$ represents the convolutional filters at the i^{th} layer, $b^{(i)}$ is the bias term, and $*$ denotes the convolution operation. The input $x^{(i-1)}$ for the first layer is the raw image, and for subsequent layers, it is the output of the previous layer $y^{(i-1)}$. After the final convolutional layer, VGG-16 applies three fully connected layers to produce the final output, which can be represented as:

$$y = W^{(FC)} \cdot \sigma(\ldots \sigma(W^{(3)} * y^{(2)} + b^{(3)}) \ldots) + b^{(FC)} \tag{6}$$

$W^{(FC)}$ and $b^{(FC)}$ are the weights and bias of the fully connected layers, respectively. The output y is then passed through a softmax function to obtain class probabilities for classification tasks.

3. Experiments

In this section, we discuss in detail about datasets, training setups, the results, and analysis.

3.1. Datasets

In this project, we utilize two datasets, miniImageNet and EuroSAT, each serving a specific purpose within the framework of transfer learning for image classification in remote sensing applications. The miniImageNet dataset, a

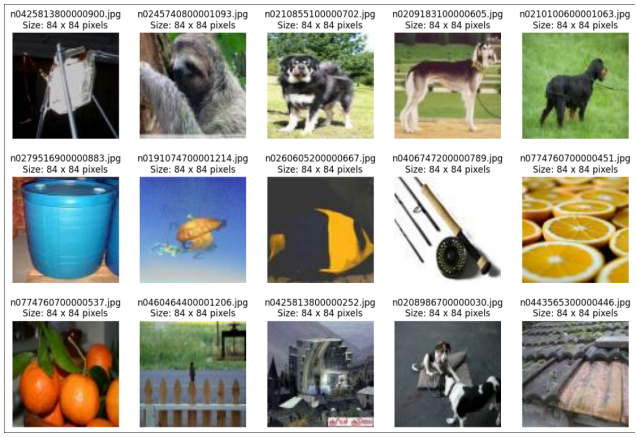

Figure 8. Samples from miniImageNet Dataset

subset of the larger ImageNet database designed for few-shot learning experiments, comprises 60,000 images across 100 classes, with each class containing 600 images [12]. We focus on the 'train.tar' file, splitting it into training, validation, and test sets. The training dataset includes 38,400 images from 64 classes, with 600 images per class, each image is of size [84x84]. Figure 8 showcases some samples from the miniImageNet dataset. We divided the 'train.tar' dataset into training, validation, and test sets in ratios of 70%, 15%, and 15%, respectively. To prevent biases from ordered data, we shuffled the data within each class folder and redistributed it into new folders for training, validation, and testing. The dataset undergoes augmentations, including random rotations of up to 10 degrees and horizontal flips, to increase variability and assist the model in learning invariant features. The images are normalized and resized to 224x224 pixels to enhance detail and ensure compatibility and consistency for the model to learn complex classification tasks [17,20]. These transformations are standard and recommended practices when using Resnet models [15]. Later the images are transformed into a tensor which is then loaded using a DataLoader.

Additionally, we obtained the EuroSAT (RGB) dataset from the provided repository. Captured by the Sentinel-2 satellite, the EuroSAT dataset is focused on Earth observation and is specifically designed for land use and land cover classification tasks [16]. We randomly chose 100 images from the EuroSAT dataset, representing five different categories, with each category including 20 samples. To ensure balanced representation across categories, we selected 25 images from these 100 samples for the training set and the remaining 75 images for the testing set. All images undergo similar augmentation steps like miniImageNet. Figure 9 showcases some samples from the EuroSAT (RGB) dataset. Each image is of size [64x64] and 10 categories corresponding to these images.

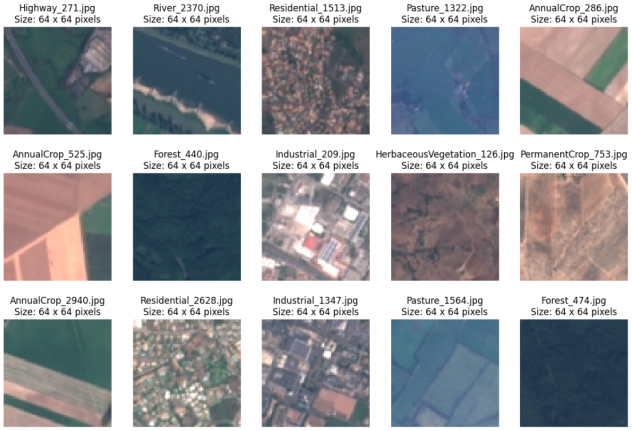

Figure 9. Samples from EuroSAT Dataset

3.2. Training setups

The experiments carried out in the project are divided in to two subsections for each of the two models. To have a fair comparison and assessments between the two models, the data pre-processing utilized was identical for both models. The steps carried out for data pre-processing are detailed in Section 3.1.

3.2.1 PreTraining

As detailed in Section 2.2, the models (ResNet-18 and VGG-16) are initially pre-trained on the miniImageNet dataset, and the top-performing weights are saved for subsequent tasks. The dataset undergoes preprocessing with specific transformations to enhance model performance, as outlined in the section 3.1.

We found key hyperparameters through experimentation to govern the final learning process:

- Number of Epochs = 8
- Learning Rate = 0.001
- Momentum = 0.9
- Criterion = CrossEntropyLoss()
- Optimizer = SGD

Three distinct pre-training methods were tested on ResNet-18. Initially, the model trained without PyTorch's pre-trained weights showed limited performance, achieving only 35-45% validation accuracy, as depicted in Figure 10. Attempts to improve outcomes by increasing epochs or adding weight decay were unsuccessful, barely surpassing 50% validation accuracy depicted in Figure 11.

This experiment was quickly abandoned due to the poor performance results obtained. Subsequent experiments utilized the 'ImageNet 1K' pre-trained weights. The second

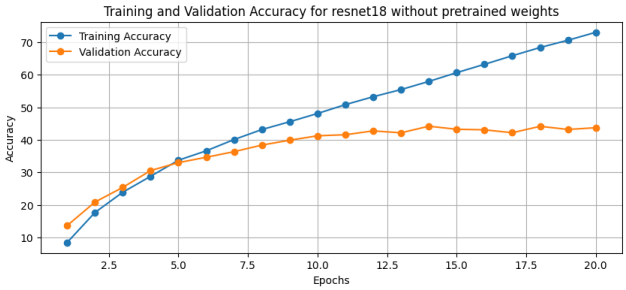

Figure 10. ResNet-18 performance with no pre-trained weights.

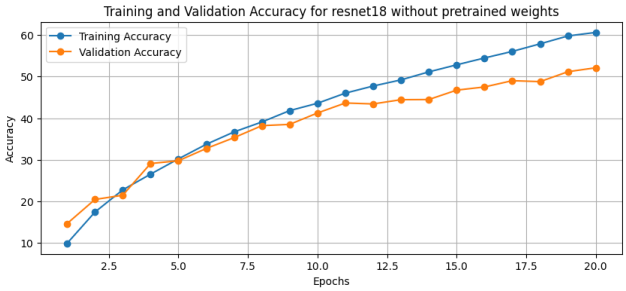

Figure 11. ResNet-18 performance with adjusted hyperparameters and no pre-trained weights.

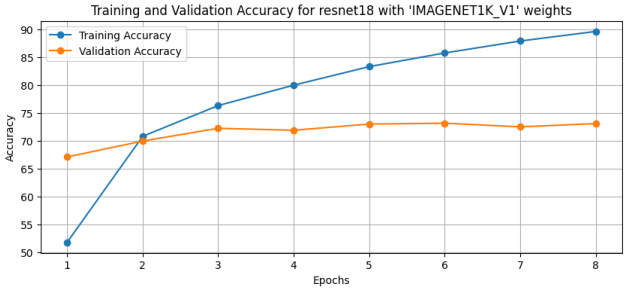

Figure 12. ResNet-18 performance with pre-trained weights and without image resized.

experiment investigated the impact of image size on model performance. The original miniImageNet images [84x84] were compared to the recommended ResNet size [224x224] [15]. Keeping the original size yielded a validation and testing accuracy of approximately 73% (Figure 12). Resizing to [224x224] significantly improved accuracy to 85% (Figure 13), indicating a substantial gain from this transformation. This optimized model was finally applied to fine-tune the EuroSAT (RGB) dataset.

Finally, applying the same configuration to VGG-16 model, we observed a validation and testing accuracy of about 80%, as shown in Figure 14.

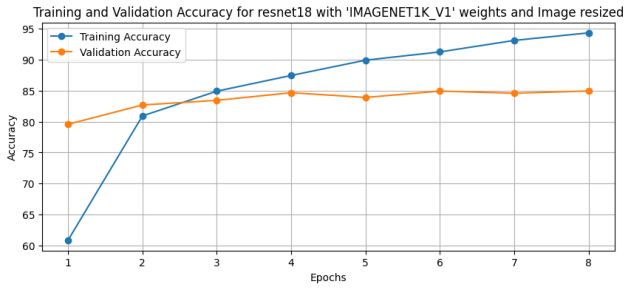

Figure 13. ResNet-18 performance with pre-trained weights and image resized.

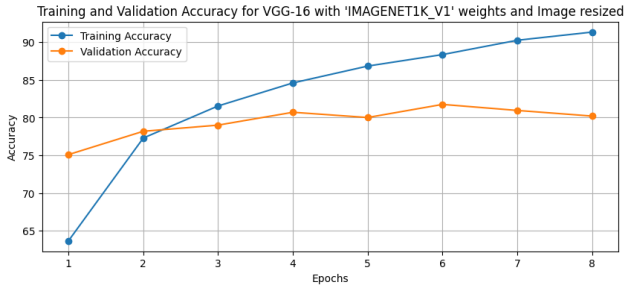

Figure 14. VGG-16 performance with pre-trained weights and image resized.

3.2.2 Fine Tuning

The ResNet-18 model is pre-trained on the miniImageNet dataset and the best-performing model is utilized as the starting point for training on the EuroSAT dataset. Similar data preparation and pre-processing are done in fine-tuning as described in Section 2.1 and Section 3.1. The hyperparameters listed in Section 3.2.1 are again re-used here to maintain some constants when analysing the model performance.

Since the dataset size is limited to 100 images (25 training and 75 testing), we run the model several times and take the average of the results to get a good understanding of the model performance since a single run on an extremely limited dataset can give extremely volatile results. Figure 15 summarizes the results obtained when running the models (pre-trained ResNet-18 and VGG-16) five times over the EuroSAT dataset.

3.3. Results

The EuroSAT dataset, which consists of satellite images, represents a distinct domain when compared to the miniImageNet dataset. This difference challenges the models' ability to transfer learned features to new contexts. The fine-tuning process of ResNet-18 has demonstrated consistent and promising results, as shown in Figure 15. Across five runs, the model achieved an average training loss of 0.440 and an average test loss of 0.500, suggesting a balanced adaptation to the EuroSAT dataset. Notably, the training accuracy averaged 87.20%, while the test accuracy was marginally lower at 82.40%. The highest training accuracy occurred in Run 3 at 96.00%, corresponding to a moderately low test loss of 0.539. However, there was a discernible trend of overfitting in runs 2 and 3, as indicated by higher training accuracies compared to the lower test accuracy. The best test accuracy was achieved in Run 5 at 86.67%, along with the lowest test loss of 0.360, indicating effective generalization in this particular instance. The close correspondence between training and test accuracies in several runs suggests that ResNet-18 was generally able to generalize well without significant overfitting. In Figure 16 illustrates the predicted label of satellite image against true values by ResNet-18 finetuend model.

In contrast, VGG-16 showed greater variability and lower overall performance metrics, as illustrated in Figure 15. The model had an average training loss of 1.114 and an average test loss of 1.157, with training and test accuracies averaging 52.80% and 51.73%, respectively. The performance varied significantly across the runs, with Run 2 exhibiting notably poor results, achieving only 28.00%

Fine-tune performance on EuroSAT Images					
Pre-trained Model on miniImageNet	**Run Number**	**Training Loss**	**Test Loss**	**Training Accuracy (%)**	**Test Accuracy (%)**
ResNet-18	1	0.561	0.431	80.00%	85.33%
	2	0.435	0.695	92.00%	76.00%
	3	0.280	0.539	96.00%	81.33%
	4	0.408	0.474	84.00%	82.67%
	5	0.515	0.360	84.00%	86.67%
ResNet-18 Average		**0.440**	**0.500**	**87.20%**	**82.40%**
VGG-16	1	0.786	1.053	64.00%	56.00%
	2	1.463	1.458	28.00%	32.00%
	3	0.862	0.950	64.00%	62.67%
	4	1.160	1.259	60.00%	52.00%
	5	1.301	1.062	48.00%	56.00%
VGG-16 Average		**1.114**	**1.157**	**52.80%**	**51.73%**

Figure 15. Pre-trained models performance on EuroSAT over 5 runs.

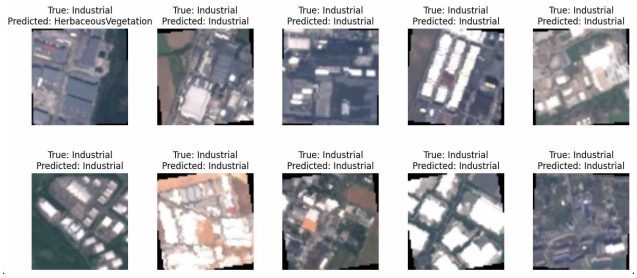

Figure 16. Prediction by ResNet-18 finetuend model

training accuracy and 32.00% test accuracy. Such inconsistency indicates a less stable adaptation to the EuroSAT dataset.

The overall analysis suggests that ResNet-18 surpasses VGG-16 in fine-tuning for the EuroSAT dataset, with higher accuracy and lower loss. These findings point to ResNet-18 being a more appropriate choice for tasks that involve satellite image classification, especially when pre-trained on image data from diverse domains.

However, there is still room for improvement. Techniques such as regularization, dropout, and the adjustment of fine-tuning hyperparameters could be considered. These methods can help mitigate overfitting and enhance both the validation accuracy and the stability of the model, potentially leading to better overall performance.

3.4. Analysis

This section discusses the potential reasons behind the observed results, exploring the architectural differences, the impact of EuroSAT image selection for fine-tuning, and the suitability of each model to the task at hand. ResNet-18 and VGG-16 differ significantly in their architecture. ResNet-18 employs residual connections to facilitate the training of deeper networks by addressing the vanishing gradient problem. This feature allows information to bypass one or more layers, which can be particularly beneficial when adapting to a new domain where certain features from the pre-trained dataset may still be relevant. The residual blocks might have enabled ResNet-18 to retain and refine useful features from miniImageNet effectively during fine-tuning on EuroSAT. In contrast, VGG-16 has a more traditional architecture with sequential convolutional layers followed by fully connected layers. Without mechanisms to mitigate the vanishing gradient problem, VGG-16 may have had difficulty optimizing and adapting its deeper layers during fine-tuning. Consequently, this might have led to the poorer and more inconsistent performance observed across the various runs.

The variation in performance across individual runs of the same model suggests that the fine-tuning process is sensitive to the specific selection of image classes from the EuroSAT dataset. Each experiment involves a dataset randomly chosen from the entire EuroSAT collection, but limited to only 5 out of the 10 available classes. As the pre-trained models are fine-tuned, the particular subset of image classes chosen can significantly impact their learning trajectory. This is evidenced by the range of training and test accuracies observed in the VGG-16 runs, suggesting that the selection of classes might considerably affect the model's ability to adapt to the EuroSAT dataset. ResNet-18, however, appears less affected by the choice of image classes, potentially due to its architectural features that may provide a degree of robustness against the variance in class selection.

ResNet-18 showed a healthy generalization from the training data to the test data, with test accuracies closely trailing the training accuracies. The relatively low test losses indicate that the features learned during fine-tuning were relevant and predictive for unseen data from the same domain. This is not the case for VGG-16. While there is not a significant discrepancy between its training and test accuracies, the overall lower values of these accuracies indicate challenges in learning effectively from the training data and extending that learning to new data. The consistency of performance across the training and test datasets suggests that the model is neither significantly overfitting nor underfitting, but it may be struggling to capture the complexity and variety inherent in the EuroSAT dataset. This could be attributed to the model's architecture or to the specific nature of the dataset, which might not align well with the features learned by VGG-16 during its pre-training on miniImageNet.

4. Conclusion

This report has provided an in-depth examination of optimizing transfer learning for image classification in the context of remote sensing, with a focus on the applications and effectiveness of deep learning and neural networks. The study primarily concentrated on the ResNet-18 and VGG-16 models, comparing their performance on the miniImageNet and EuroSAT-RGB datasets. This comparison was instrumental in demonstrating the capacity of transfer learning to overcome challenges associated with limited labeled data in remote sensing. Additionally, the report underscored the importance of implementing strategies such as regularization, dropout, and fine-tuning of hyperparameters to improve the performance of machine learning models. The empirical results presented in this study clearly highlight the efficacy of transfer learning, particularly through the use of ResNet-18 and VGG-16 models. Notably, ResNet-18 outperformed VGG-16, consistently achieving higher levels of performance across various runs. On average, ResNet-18 attained a test accuracy of 82.4%, significantly surpassing the 51.73% test accuracy of VGG-16 models. The thorough investigation into the architectures and methodologies of these models has provided critical insights into their individual strengths and limitations. Based on the findings of this report, ResNet-18 stands out as the more favorable option for tasks in remote sensing image classification, due to its superior adaptability and effectiveness in transfer learning scenarios.

References

[1] Challenges in Deep Learning | HackerNoon | https://shorturl.at/fuLMY. 1

[2] mini-imagenet | Google Drive | https://shorturl.at/nosxT. 3

[3] VGGNet-16 Architecture: A Complete Guide. 4, 5

[4] What is Deep Learning? |IBM | https://www.ibm.com/topics/deep-learning. 1

[5] Transfer learning | DeepAI | https://shorturl.at/bJPSY, May 2019. 1

[6] Deep learning | Wikipedia | https://shorturl.at/pKQSV, Jan. 2024. Page Version ID: 1193512259. 1

[7] Mahmoud Ahmed, Naser El-Sheimy, Henry Leung, and Adel Moussa. Enhancing Object Detection in Remote Sensing: A Hybrid YOLOv7 and Transformer Approach with Automatic Model Selection. *Remote Sensing*, 16(1):51, Jan. 2024. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute. 2

[8] Laith Alzubaidi, Jinshuai Bai, Aiman Al-Sabaawi, Jose Santamaría, A. S. Albahri, Bashar Sami Nayyef Al-dabbagh, Mohammed A. Fadhel, Mohamed Manoufali, Jinglan Zhang, Ali H. Al-Timemy, Ye Duan, Amjed Abdullah, Laith Farhan, Yi Lu, Ashish Gupta, Felix Albu, Amin Abbosh, and Yuantong Gu. A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. *Journal of Big Data*, 10(1):46, Apr. 2023. 1

[9] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1):53, Mar. 2021. 1

[10] Gaudenz Boesch. The 100 Most Popular Computer Vision Applications in 2024, Dec. 2023. 2

[11] Jason Brownlee. A Gentle Introduction to Transfer Learning for Deep Learning, Dec. 2017. 2

[12] Da Chen, Yuefeng Chen, Yuhong Li, Feng Mao, Yuan He, and Hui Xue. Self-Supervised Learning For Few-Shot Image Classification, Feb. 2021. arXiv:1911.06045 [cs]. 5

[13] Karim Ennouri, Slim Smaoui, and Mohamed Ali Triki. Detection of Urban and Environmental Changes via Remote Sensing. *Circular Economy and Sustainability*, 1(4):1423–1437, Dec. 2021. 2

[14] Minghui Guo, Kangjian Wang, Shunlan Liu, Yongzhao Du, Peizhong Liu, Qichen Su, and Guorong Lv. Recognition of thyroid ultrasound standard plane images based on residual network. *Computational Intelligence and Neuroscience*, 2021:1–11, 06 2021. 4

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. https://arxiv.org/pdf/1512.03385v1.pdf. 3, 4, 5, 6

[16] Patrick Helber. phelber/EuroSAT, Jan. 2024. original-date: 2018-12-21T23:39:26Z. 2, 3, 5

[17] Wanli Liu, Chen Li, Md Mamunur Rahamana, Tao Jiang, Hongzan Sun, Xiangchen Wu, Weiming Hu, Haoyuan Chen, Changhao Sun, Yudong Yao, and Marcin Grzegorzek. Is the aspect ratio of cells important in deep learning? A robust comparison of deep learning methods for multi-scale cytopathology cell image classification: from convolutional neural networks to visual transformers, Nov. 2021. arXiv:2105.07402 [cs]. 5

[18] Maryam M. Najafabadi, Flavio Villanustre, Taghi M. Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1, Feb. 2015. 1

[19] Iqbal H. Sarker. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, 2(6):420, Aug. 2021. 1, 2

[20] Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60, July 2019. 5

[21] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, Apr. 2015. arXiv:1409.1556 [cs]. 4

[22] Yashbir Singh, Colleen Farrelly, Quincy A. Hathaway, Ashok Choudhary, Gunnar Carlsson, Bradley Erickson, and Tim Leiner. The Role of Geometry in Convolutional Neural Networks for Medical Imaging. *Mayo Clinic Proceedings: Digital Health*, 1(4):519–526, Dec. 2023. 2

[23] Zhiyuan Tang, Dong Wang, Yiqiao Pan, and Zhiyong Zhang. Knowledge Transfer Pre-training, June 2015. arXiv:1506.02256 [cs, stat]. 2

[24] Mohammad Mustafa Taye. Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. *Computers*, 12(5):91, May 2023. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute. 1

[25] National Oceanic and Atmospheric Administration US Department of Commerce. What is remote sensing? 2

[26] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning, Dec. 2017. arXiv:1606.04080 [cs, stat]. 2

[27] Zehui Zhao, Laith Alzubaidi, Jinglan Zhang, Ye Duan, and Yuantong Gu. A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations. *Expert Systems with Applications*, 242:122807, May 2024. 2

[28] Wenbo Zhu, Birgit Braun, Leo H. Chiang, and Jose A. Romagnoli. Investigation of transfer learning for image classification and impact on training sample size. *Chemometrics and Intelligent Laboratory Systems*, 211:104269, Apr. 2021. 2