ENS491-492 - Graduation Project Final Report

Project Title: Domain Adaptation for Remote Sensing in Scene Classification: An Investigation into Adaptive Dropout and Adaptive Batch Normalization Techniques

Group Members: Emirhan Böge, Ekin Beyazıt

Supervisor: Erchan Aptoula

Date: 4.06.2023

Contents

1 Executive Summary

Domain adaptation in remote sensing scene classification is an essential area of study due to the scarcity of labeled data. In this study, we combine one of the well-established domain adaptation methods, domain-adversarial Training, with adaptive dropout and adaptive batch normalization to enhance performance in this area. To the best of our knowledge, this is the first application of adaptive dropout and adaptive batch normalization in the context of domain adaptation in remote sensing scene classification research.

We have employed a pre-trained ResNet-50 model and augmented it with an adaptive batch normalization layer and adaptive dropout layer. This model was fine-tuned on the source data with the aim of better feature extraction and label prediction. Furthermore, we have compared two notable methods for domain adaptation using the ResNet-50 model, which are DANN and DeepCoral, and we have leveraged the accuracy evaluation metric for evaluation.

We evaluated our models on two datasets, AID and UC Merced, under various configurations: with and without adaptive dropout and adaptive batch Normalization.

With our approach we have concluded that these methods have varied effects in different scenarios. While the adaptive dropout caused performance increase in most cases, we observed that its benefits may be hindered while used along with adaptive batch normalization. Additionally, the adaptive batch normalization

layer, possibly due to its lack of domain-related mechanisms, seems to decrease the model performance in most scenarios.

Our findings point to a benefit in further studies examining the interaction between the standout and adaptive batch normalization layers, and suggest that standout could also improve the performance of the remaining state-of-the-art architectures.

2 Introduction

2.1 Background

2.1.1 Scene Classification and Remote Sensing

Scene classification is the task of assigning labels to different scenes based on their content. It covers a wide array of potential scenes, including urban environments, natural landscapes, residential areas, and industrial areas. This research plays a vital role in various applications like autonomous navigation, surveillance, and disaster management, among others.

The task of scene classification becomes much more critical and complex in the context of remote sensing. Remote sensing data is typically gathered through satellite or aircraft-based sensor technologies and is about acquiring information about an object or a phenomenon without making physical contact with the object. Remote sensing made it possible to gain a deep understanding of our environment and contributed to multiple fields, including geography, forestry, meteorology, and urban planning.

Despite the valuable role of scene classification and remote sensing, it has significant challenges mainly because of the high variability in image representations [9]. There are wide range of possibilities of visual appearances for any given scene type, and with the varying perspective, scales, and occlusion, creating a generalizable scene classifier can be challenging. Moreover, the increased availability of images and the diversity of scenes due to the development of digi-

tal cameras and social media platforms create an additional challenge. There are databases like Places [33] with millions of images from numerous unique places. The complexity and diversity of these datasets make it challenging to design and train a successful classifier that can perform well across a wide variety of scenes.

Deep learning techniques have brought considerable improvements in the task of scene classification [34]. These techniques have the capacity to learn hierarchical features from raw pixel data. However, deep learning requires vast amounts of labeled data [22], and obtaining such vast data in remote sensing is often infeasible because of the high acquisition costs and time constraints. This is the point where techniques like domain adaptation become essential, which aims to transfer knowledge from one domain where labeled data is high to another domain where labeled data is low.

2.1.2 The Role of Domain Adaptation

The research topic of domain adaptation has emerged as a practical solution to the limited availability of labeled data problems in many fields, including remote sensing classification. Domain adaptation techniques aim to transfer knowledge from a source domain to a target domain where labeled data is not available as available in the source domain [4]. This approach is crucial as it helps models to generalize and perform well in another, different domain.

Domain adaptation methods can handle situations where the data distributions of the source and the target domains are different, and this situation is often encountered in remote sensing problems. These methods try to learn domain-

invariant features, which are features that are helpful in both the source domain and the target domain [6].

There exist several approaches for domain adaptation. For example, instance-based methods that reweight the source samples, feature-based methods that align the source and target data distributions in a common feature space, deep learning approaches that learn to extract domain-invariant features using adversarial methods [4, 6, 16].

We aim to improve the classification performance in the target domain for the scene classification process by integrating domain adaptation techniques to leverage the information present in the source domain effectively.

2.1.3 Adaptive Dropout and Batch Normalization

For enhancing the performance of domain adaptation, other techniques can be used for remote sensing classification. In this study, we investigate the effectiveness of adaptive dropout [2] and adaptive batch normalization [14].

Dropout is a simple yet effective regularization technique to prevent overfitting in neural networks [20]. Dropout randomly sets a fraction of neurons to 0, which helps prevent overfitting by ensuring that no single neuron gets too much trained. Adaptive dropout advances the original dropout concept by allowing the dropout probability to be different for different units, which makes the trained model more robust to overfitting and increases its generalization performance.

Batch normalization is another technique that has been used to improve the training of deep neural networks [11]. It normalizes the activations of each layer of the network. As a result, it helps reduce the covariate shift problem and accelerates the training process. On the other hand, adaptive batch normalization adjusts the normalization parameters based on the specific characteristics of the source and target domains [13]. Adaptive batch normalization makes the model more robust to domain shifts and improves its performance on the target domain.

The combination of adaptive dropout and adaptive batch normalization with domain adaptation techniques can provide a powerful tool for handling the challenges of domain adaptation in remote sensing scene classification. By integrating these techniques, we can build models that are robust to overfitting and adapt to domain shifts.

2.2 Literature Review

Remote sensing scene classification and domain adaptation have been explored previously. Several innovative approaches have been proposed, each adding unique contributions to the field.

ADA-BDC, an unsupervised adversarial domain adaptation method, was proposed by [15], utilizing generative adversarial nets [8] to align the distributions of different domains.

Further advancements included methods like the classifier-contained deep ad-

versarial domain adaptation [23] for cross-domain semi-supervised classification in remote sensing images and the introduction of subspace alignment layers to CNN models for domain adaptation in remote sensing scene image classification [19].

The correlation subspace dynamic distribution alignment was proposed [29] to balance the source and target domains by maximizing subspace correlation and aligning dynamic statistical distributions. Additionally, unsupervised domain adaptation research using error-correcting boundaries and a feature adaptation metric for remote-sensing scene classification was proposed [27].

The domain feature enhancement network [30] was introduced to adaptively enhance the discriminative ability of learned features for domain variances in scene classification. In addition, efforts were made to improve multi-source semi-supervised domain adaptation, such as using an EfficientNet-B3 convolutional neural network model [12].

Partial domain adaptation, where the source domain's label space covers the target domain's label space, was researched [31]. DATSNET framework [32] was proposed to help with aligning the feature distributions of the source and the target domains using task-specific decision boundaries.

The universal domain adaptation method UniDA was proposed for cases when the source data is unavailable [26]. Additionally, methods like the easy-to-hard adaptation structure [17] have been developed to tackle multitarget domain adap-

tation problems by encoding semantic information between target domains.

2.3 Problem Statement

Scene classification with remote sensing techniques is essential for various applications such as land-use monitoring, disaster management, and urban planning. However, due to the lack of labeled data, developing models across different domains is not easy.

The practical implementation of remote sensing scene classification models often encounters numerous problems. First, remote sensing data makes it diverse and heterogeneous, leading to a broad spectrum of representations for any scene. Second, the amount of labeled data required to train deep learning models effectively is not always available or feasible to obtain.

These issues show the need for development in domain adaptation techniques, which can address the challenge posed by the scarcity of labeled data in the target domain.

In this study, we focus on enhancing domain adaptation strategies for remote sensing scene classification by utilizing adaptive dropout and adaptive batch normalization techniques. We aim to build a model that is robust against overfitting, more stable during training, and, most importantly, more accurate in classifying scenes from various domains.

Our most significant commitment lies in the exploration and application of new techniques, adaptive dropout and adaptive batch normalization. We aim to contribute by discovering novel insights and developing improved methods for domain adaptation in remote sensing scene classification data.

2.4 Objectives/Tasks

- Conduct a literature review in order to learn about optimal approaches to the problem of domain adaptation for remote sensing scene classification, and to see how much room for improvement there is.

- Replicate the experiments and results of selected research papers, understand the proposed neural network structures in order to learn how to improve upon them.

- Establish baselines by examining model performances without any domain adaptation techniques. These baselines are supposed to be the boundaries of our model's performance.

- Implement our proposed architecture on a preferred platform (we chose Google Colab).

- Replicate the experimental setup of the selected research papers and exper-

iment with our proposed architecture. This way, we have the ability to see how well our model performs compared to the existing literature.

- Perform hyperparameter tuning and compare the results with the selected research papers.

2.5 Realistic Constraints

Economic constraints: Normally, non-theoretical machine learning research is carried out using high performance GPUs. Unfortunately, we are not in possession of such expensive equipment and this led our experiments to be very time-consuming and inefficient.

Manufacturability: Replicating the experimental setups and architectures of state-of-the-art literature was hard to achieve because the papers in question did not publish any code. Another possible problem of manufacturability was the lack of publications in the specific topic we are working on. But, this also implied that there was a lot of room for trying novel approaches. Our most significant problem would stem from the fact that the architecture of our design requires in-depth PyTorch (and several other libraries for using methods such as pruning) knowledge and experience. Thankfully, we have been able to gain such experience during our studies in the previous term. Still, the implementation of our design did prove to be quite laborious.

3 Methodology

3.1 Dataset

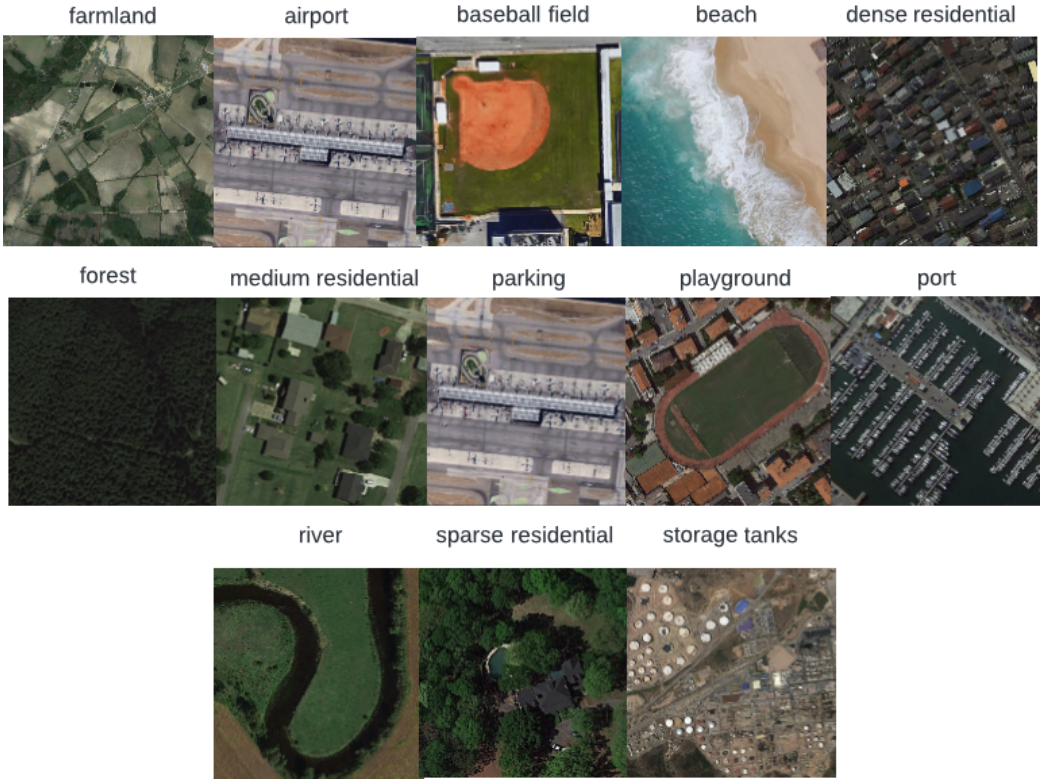

Figure 1: Samples of the 13 common classes from the AID dataset.

To train and test an effective domain adaptation strategy for remote sensing scene classification, we have leveraged two pre-existing datasets: the UC Merced (UCM) dataset [28], and the AID dataset [25].

The UCM dataset is composed of 21 unique scenes, classes, each having 100 images of size 256 x 256 pixels. Each image has a high spatial resolution of 1 foot, which provides detailed information for classification tasks. This dataset covers

various scenes, including agricultural landscapes and residential areas.

On the other hand, the AID dataset provides a more extensive collection of 30 unique classes, where each class contains between 220 and 420 images of size 600 x 600 pixels. This dataset offers spatial resolutions from 0.5 to 0.8 meters.

In this study, we have utilized 13 common classes shared between these two datasets. These include scenes of 'Farmland', 'Airport', 'BaseballField', 'Beach', 'DenseResidential', 'Forest', 'MediumResidential', 'Parking', 'Playground', 'Port', 'River', 'SparseResidential', and 'StorageTanks'.

To make sure that our models process the images from these datasets effectively, we have employed a series of data transformations. The transformations include resizing the images to a standard size of 256 x 256 pixels to maintain consistency across datasets, converting the images to tensors for compatibility with the PyTorch deep learning library [18], and normalizing the pixel values of images using the mean and standard deviation of the ImageNet [5] dataset.

Figure 1 is a sample of the images in the datasets, depicting the common 13 classes.m

3.2 Techniques Used

3.2.1 ResNet Architecture

The Residual Network (ResNet) [10], developed by Kaiming He et al., is a landmark innovation in the history of deep neural networks. Its primary contribution

is the concept of skip connections which allow gradients from early layers to flow through the later layers of the network directly. This idea allows layers to learn ”residual mappings” that add some detail to the existing knowledge represented in the network. ResNet architecture have been used in various domains, including remote sensing scene classification [24].

3.2.2 DANN and DeepCORAL Methods

Domain-adversarial training of neural networks (DANN) [7] is a method proposed for tackling the domain adaptation problem. In the DANN approach, a neural network is trained to be domain invariant. It learns a feature representation that is useful for the classification task and simultaneously confuses a domain classifier network. This way, it ensures that the learned features are domain-invariant, and thus, the network can generalize well to the target domain.

DeepCORAL, deep correlation alignment for unsupervised domain adaptation, [21] is another domain adaptation method, which is focused on aligning the second-order statistics of source and target distributions. It minimises the correlation discrepancy between source and target layers in a deep neural network.

3.3 Experimental Design

3.3.1 Architecture

While designing our architecture, we chose the ResNet-50 architecture as the base model for our domain adaptation problem. This architecture is a 50-layer deep variation of residual networks. We took advantage of the ResNet-50's pre-trained

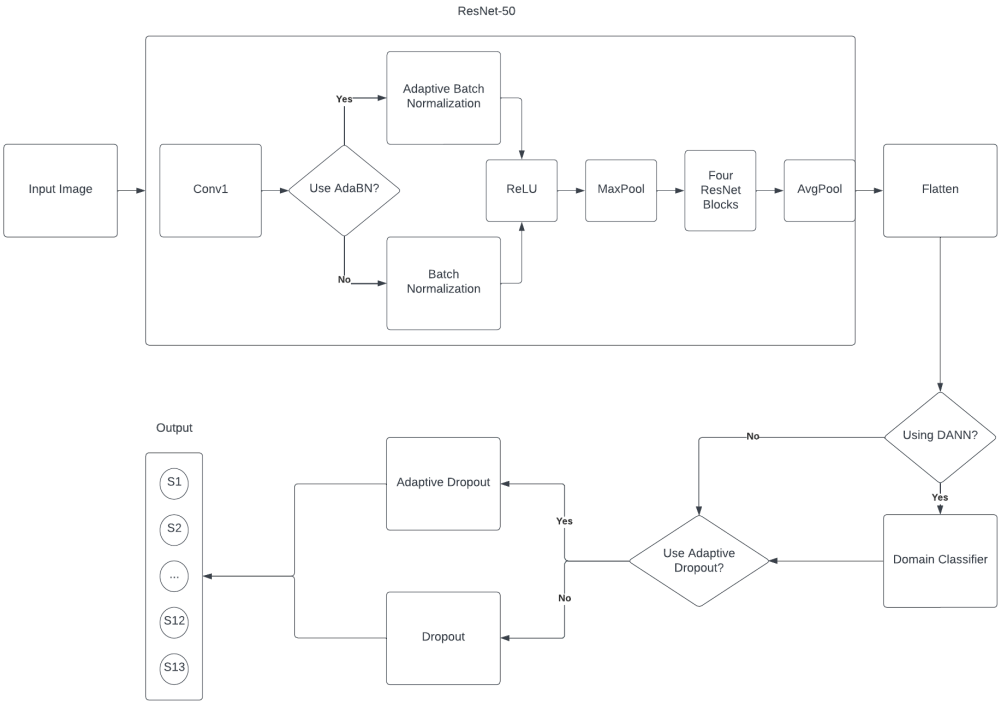

Figure 2: Overview of the leveraged architecture.

model, which allowed us to start with a network that has already learned useful image features and can be further fine-tuned on our specific task of remote sensing scene classification.

Furthermore, for our task we have extended the ResNet-50 model by introducing the concept of adaptive batch normalization and adaptive dropout. We aimed to give the network the ability to adjust its normalization and regularization techniques while taking characteristics of the target domain data into account.

When the adaptive batch normalization was used, the parameters of batch normalization layers were adjusted, which allowed the model to quickly adapt to the

target domain. This technique uses a different set of batch normalization parameters for each domain, helping to handle the domain shift problem. Moreover, when the adaptive dropout was utilized, it controls the learning capacity of the model during the adaptation phase.

Finally, depending on the chosen domain adaptation method, if using DANN, we utilized a domain classifier, which consisted of fully connected layers with ReLU activation [1] and dropout for regularization. The domain classifier was used to confuse the network, which enables that the model learns features that were domain-invariant, allowing for better performance on the target domain.

Overall, the selection of ResNet-50 as the base architecture and the introduction of adaptive batch normalization and adaptive dropout techniques, combined with the chosen domain adaptation methods, were instrumental in achieving successful domain adaptation for remote sensing scene classification.

You can see the flowchart of the implemented architecture in the Figure 2

3.3.2 Implementation of Adaptive Dropout and Adaptive Batch Normalization

Our adaptive dropout, or standout [2] implementation is utilizing one simple function, sample mask aside from its main code. This sample mask operation, given a tensor of retention probabilities (the probability a node will be retained) returns a tensor consisting of 1s and 0s determining which nodes are dropped out. The layer's parameters are also trained along with the rest of the network while apply-

ing these retention masks. So, as training progresses, the standout layer prioritizes keeping the nodes impacting the decision-making while dropping the ones that do not. This is why the layer takes both the input and output of a layer as parameters.

The adaptive batch normalization [3] implementation is simply a 2D batch normalization layer with trainable parameters. These weights are trained so that they can adapt the operation to the training set's distribution and also adapt to the operation itself, possibly making it more efficient. This also allows the strength of the normalization to become somewhat adjustable or dynamic. This layer simply returns a the sum of one weight, and the multiplication of the normalized batch with a trainable scalar.

3.3.3 Comparative Study of DANN and DeepCORAL Methods

As part of our work, we have conducted a comparative study between DANN and DeepCORAL. Both methods were implemented with the ResNet-50 as backbone architecture to ensure a fair comparison.

We introduced additional modifications such as adaptive batch normalization and adaptive dropout to these architectures, and by doing so we have compared these methods too.

We compared these two methods in different settings: using only adaptive dropout, using only adaptive batch normalization, using both adaptive dropout and adaptive batch normalization, and using none of the adaptive methods. This approach enabled us to thoroughly understand the strengths and weaknesses of

each method, and how they interact with our additional modifications using the adaptive methods.

4 Results and Discussion

4.1 Presentation of Results

In this research, we performed a comprehensive experiment to analyze the performance of our proposed domain adaptation method in remote sensing scene classification domain. We have used the accuracy metric to conduct our evaluations. The results of our experiment are summarized in the Tables 1 and 2.

In Table 1, we observe that the highest accuracy for the AID to UCM transfer is achieved when DANN is employed with adaptive dropout and without adaptive batch normalization, which results with an accuracy of 71.52%. For the DeepCORAL method, the highest accuracy is 69.53% with the same configuration.

In Table 2, the highest accuracy for the UCM to AID transfer is achieved when DANN is used with adaptive dropout and without adaptive batch normalization, which results with an accuracy of 64.61%. However, for the DeepCORAL method, the highest accuracy of 65.11% is achieved when both adaptive dropout and adaptive batch normalization are employed.

4.2 Discussion

The results from both Tables 1 and 2 show that the standout layer successfully improves performance when used only by itself.

AD	AdaBN	DANN	DeepCORAL
0	0	64.92%	64.45%
0	1	63.88%	66.87%
1	0	71.52%	69.53%
1	1	68.66%	62.23%

Table 1: Results of our models for AID to UCM transfer with various configurations of Adaptive Dropout (AD) and Adaptive Batch Normalization (AdaBN). 1: True, 0: False.

AD	AdaBN	DANN	DeepCORAL
0	0	60.64%	61.03%
0	1	55.71%	56.86%
1	0	64.61%	62.37%
1	1	49.85%	65.11%

Table 2: Results of our models for UCM to AID transfer with various configurations of Adaptive Dropout (AD) and Adaptive Batch Normalization (AdaBN). 1: True, 0: False.

As can be observed from the Tables 1 and 2, there seems to be a performance drop caused by using adaptive dropout and adaptive batch normalization together. We hypothesize that the interaction between these two modules may be causing them to train themselves towards conflicting objectives. While the Standout layer affects which nodes are dropped according to the inputs and outputs of a fully connected layer, batch normalization adjusts the data itself. This may be causing the activity to shift to other neurons, possibly sabotaging the Standout layer. A similar interaction may have happened between DANN's domain classifier and the adaptive batch normalization layer.

Additionally, our adaptive batch normalization implementation does not differentiate between source and target data. This, along with the fact that the layer is trainable, may have negatively affected the mechanism. We hypothesize that

when fed with both source and target data uniformly, the layer might have ended up taking the middle of the domain shift as a reference point, instead of aligning the distributions.

5 Impact

5.1 Practical Implications

There are no immediate practical impacts of this research project aside from pointing out the benefits and downsides of our studied methods for the field we are working in. It seems that our implementation of adaptive batch normalization is not fit for domain application, and may hinder the performance of the model when used carelessly. adaptive dropout, on the other hand may increase performance as its sole training purpose is increasing the performance or lowering the error with no regard to the distribution of the input.

5.2 Theoretical Implications

We discovered that any application of adaptive batch normalization in the field of domain adaptation should possibly include some sort of mechanism towards discriminating between the source and target distributions or aligning them. When done without these mechanisms, we observed that the batch normalizer may have possibly trained itself in such a way that will place itself between the two distributions, resulting in incorrect decisions for both source and target scenarios.

6 Ethical Issues

There are no ethical issues regarding our project.

7 Project Management

Our project has changed objectives and plans multiple times during its various stages, due to the abundance of possibilities to work on and varying obstacles we encountered. We learned a lot about how keeping our findings in an organized manner, along with our project files has a significant impact on our productivity and the overall quality of our work. Additionally, we learned that attempting such projects are not only a matter of working hard, but are also about choosing the right objectives considering the given time and resources at hand.

8 Conclusion

Observing adaptive dropout's potential for improving domain adaptation performance, studies utilizing other state-of-the-art architectures should be conducted to further verify the effects of this mechanism. Keeping in mind that our backbone, ResNet50 does not originally include dropout layers, scenarios where adaptive dropout can be used instead of regular dropout layers may have more drastic performance improvements. Possibly, adding a domain related mechanism, perhaps training the weights to prioritize retaining the nodes that are effective for inputs of both domains, to the Standout layer may also further improve performance in the task.

Studies investigating the interactions between the adaptive batch normalization and standout layers may shed some light onto the issue with the performance drop. Because there are scenarios where both mechanisms individually improve performance, developing a synergy between the two can prove to be beneficial.

Altering the implementation to the adaptive batch normalization layer in order to account for the domain shift inside the batches, or adding a mechanism to somehow align the two distributions could also prove to be beneficial. Normalizing batches such that domain-invariant samples are more apparent could create a DANN-like effect, maybe even supplementing the domain classifiers if used together.

9 Appendix

9.1 ResNet50_DA Class

```
class ResNet50_DA(nn.Module):
    def __init__(self, num_classes, adaptive_dropout=False, adaptive_bn=False):
        super(ResNet50_DA, self).__init__()
        self.base_model = resnet50(pretrained=True)
        self.adaptive_dropout = adaptive_dropout
        self.adaptive_bn = adaptive_bn

        if self.adaptive_bn:
            self.base_model.bn1 = AdaptiveBatchNorm2d(64)
```

```
        self.fc1 = nn.Linear(2048, 1024)
        if self.adaptive_dropout:
            self.std1 = Standout(self.fc1.weight, 0.5, 0.5, nn.Sigmoid())
        self.out = nn.Linear(1024, num_classes)

        self.domain_classifier = nn.Sequential(
            nn.Linear(2048, 1024),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(1024, 1024),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(1024, 2),
        )

    def forward(self, x, alpha):
        features = self.base_model.conv1(x)
        features = self.base_model.bn1(features)
        features = self.base_model.relu(features)
        features = self.base_model.maxpool(features)
        features = self.base_model.layer1(features)
        features = self.base_model.layer2(features)
        features = self.base_model.layer3(features)
        features = self.base_model.layer4(features)
```

```
        features = self.base_model.avgpool(features)
        features = torch.flatten(features, 1)

        reverse_features = ReverseLayerF.apply(features, alpha)
        inputs1 = features.clone().detach()
        features = F.relu(self.fc1(features))
        if self.adaptive_dropout:
            features = self.std1(inputs1, features)

        class_output = self.out(features)
        domain_output = self.domain_classifier(reverse_features)
        return class_output, domain_output
```

9.2 AdaptiveBatchNorm2d Class

```
class AdaptiveBatchNorm2d(nn.Module):
    def __init__(self, num_features, eps=1e-5, momentum=0.1, affine=True):
        super(AdaptiveBatchNorm2d, self).__init__()
        self.bn = nn.BatchNorm2d(num_features, eps, momentum, affine)
        self.a = nn.Parameter(torch.FloatTensor(1, 1, 1, 1))
        self.b = nn.Parameter(torch.FloatTensor(1, 1, 1, 1))

    def forward(self, x):
        return self.a * x + self.b * self.bn(x)
```

9.3 Standout Class

```
class Standout(nn.Module):
    def __init__(self, previousWeights, alpha, beta, nonlinearity):
        super(Standout, self).__init__()
        self.pi = previousWeights
        self.alpha = alpha
        self.beta = beta
        self.nonlinearity = nonlinearity

    def forward(self, inputs, outputs):
        self.p = self.nonlinearity(self.alpha*inputs.matmul(self.pi.t())
                            + self.beta)
        if(self.training):
            self.mask = sample_mask(self.p)
            return self.mask*outputs
        else:
            return self.p*outputs
```

References

[1] A. F. Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] J. Ba and B. Frey. Adaptive dropout for training deep neural networks. *Advances in neural information processing systems*, 26, 2013.

[3] Q. Chen, J. Xu, and V. Koltun. Fast image processing with fully-convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2497–2506, 2017.

[4] G. Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[9] Y. Gu, Y. Wang, and Y. Li. A survey on deep learning-driven remote sensing image scene understanding: Scene classification, scene retrieval and scene-guided object detection. *Applied Sciences*, 9(10):2110, 2019.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[12] T. Lasloum, H. Alhichri, Y. Bazi, and N. Alajlan. Ssdan: Multi-source semi-supervised domain adaptation network for remote sensing scene classification. *Remote Sensing*, 13(19):3861, 2021.

[13] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

[14] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[15] W. Liu and F. Su. A novel unsupervised adversarial domain adaptation network for remotely sensed scene classification. *International Journal of Remote Sensing*, 41(16):6099–6116, 2020.

[16] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[17] B. H. Ngo, Y. J. Chae, J. H. Park, J. H. Kim, and S. I. Cho. Easy-to-hard structure for remote sensing scene classification in multi-target domain adaptation. *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[19] S. Song, H. Yu, Z. Miao, Q. Zhang, Y. Lin, and S. Wang. Domain adaptation for convolutional neural networks-based remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters*, 16(8):1324–1328, 2019.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[21] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 443–450. Springer, 2016.

[22] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[23] W. Teng, N. Wang, H. Shi, Y. Liu, and J. Wang. Classifier-constrained deep adversarial domain adaptation for cross-domain semisupervised classification in remote sensing images. *IEEE Geoscience and Remote Sensing Letters*, 17(5):789–793, 2019.

[24] M. Wang, X. Zhang, X. Niu, F. Wang, and X. Zhang. Scene classification of high-resolution remotely sensed image based on resnet. *Journal of Geovisualization and Spatial Analysis*, 3:1–9, 2019.

[25] G.-S. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, L. Zhang, and X. Lu. Aid: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7):3965–3981, 2017.

[26] Q. Xu, Y. Shi, X. Yuan, and X. X. Zhu. Universal domain adaptation for remote sensing image scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–15, 2023.

[27] J. Yang, H. Chen, Y. Xu, Z. Shi, R. Luo, L. Xie, and R. Su. Domain adaptation for degraded remote scene classification. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 111–117. IEEE, 2020.

[28] Y. Yang and S. Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 270–279, 2010.

[29] J. Zhang, J. Liu, B. Pan, and Z. Shi. Domain adaptation based on correlation subspace dynamic distribution alignment for remote sensing image scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 58(11):7920–7930, 2020.

[30] X. Zhang, X. Yao, X. Feng, G. Cheng, and J. Han. Dfenet for domain adaptation-based remote sensing scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–11, 2021.

[31] J. Zheng, Y. Zhao, W. Wu, M. Chen, W. Li, and H. Fu. Partial domain adaptation for scene classification from remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 2022.

[32] Z. Zheng, Y. Zhong, Y. Su, and A. Ma. Domain adaptation via a task-specific classifier framework for remote sensing cross-scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13, 2022.

[33] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[34] Q. Zou, L. Ni, T. Zhang, and Q. Wang. Deep learning based feature selection for remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters*, 12(11):2321–2325, 2015.