Implementing Overparameterization in Deep Learning: Analyzing Generalization in Satellite Image Classification

Jahnavi Lasyapriya Vavilala
Department of Electrical Engineering
Arizona State University, Tempe, AZ 85282
jvavilal@asu.edu

Sruthi Keerthana Nuttakki
Department of Electrical Engineering
Arizona State University, Tempe, AZ 85282
snuttakk@asu.edu

Pavan Kancharla
Department of Electrical Engineering
Arizona State University, Tempe, AZ 85282
pkancha3@asu.edu

Tejas Murlidhar Mundada
Department of Computer Science
Arizona State University, Tempe, AZ 85282
tmundada@asu.edu

Abstract

The project examines the effects of overparameterization on generalization in deep learning, focusing on satellite image classification. The researchers implement three convolutional neural networks—SmallNet, MediumNet, and LargeNet—with increasing complexity and train them on a remote sensing dataset using optimizers such as SGD with momentum, RMSprop, and Adam. Through detailed experiments, they observe that larger models converge faster but are prone to overfitting, yet they frequently achieve higher test accuracy, particularly when paired with adaptive optimizers like Adam. To investigate model redundancy, the team conducts post-training pruning and finds that overparameterized models maintain strong accuracy even after significant parameter reduction, indicating robustness and implicit regularization. These findings align with deep learning theories, suggesting that overparameterization enhances optimization and generalization in high-capacity networks.

1 Introduction

Overparameterization is growing in prevalence in the field of Machine Learning. In this,the neural networks have more parameters than necessary to fit the training data. Evaluating under the impression of overfitting, overparameterized models often generalize surprisingly well in practice. Understanding why this happens is essential for building efficient, robust models in applications like satellite image classification, where generalization is crucial due to high data variability and limited labeled samples. Moreover, satellite image classification plays a key role in environmental monitoring, urban planning, and disaster response, making performance and efficiency vital.

1.1 Problem Statement

The project specifically considers the size of network and the choice of optimizer impact performance on the EuroSAT dataset.The objective is to clarify whether larger models trained with different optimization algorithms generalize better, train faster, or behave differently in terms of training dynamics.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

1.2 Contribution

The contributions to this project include:

- Empirical evaluation of the effect of overparameterization on optimization and generalization using three convolutional network architectures of increasing size.
- Comparison in the behavior of three optimizers—SGD with momentum, RMSprop, and Adam—across networks of varying capacity.
- Assessing the impact of pruning on model performance to determine the redundancy in overparameterized networks.

2 Literature Review

2.1 Theoretical Foundations of Overparameterization

Overparameterization has emerged as a significant area of study in deep learning theory. The conventional wisdom in machine learning suggests that models with excessive parameters relative to the training data would lead to overfitting and poor generalization (1). However, recent theoretical work has challenged this notion, particularly in the context of neural networks.

Zhang et al. (2) demonstrated that modern deep networks can perfectly fit randomly labeled training data, yet still generalize well on real data, suggesting that traditional capacity measures fail to explain generalization in deep learning. This phenomenon, often called the "interpolation regime," has led to new theoretical frameworks for understanding generalization in overparameterized models.

Belkin et al. (3) introduced the concept of "double descent," where test error initially decreases, then increases as model capacity grows (matching classical learning theory), but then surprisingly decreases again as models become highly overparameterized. This U-shaped curve followed by a second descent provides a framework for understanding why very large models can generalize well despite having the capacity to memorize training data.

2.2 Optimization Dynamics in Overparameterized Networks

Du et al. (4) and Allen-Zhu et al. (5) established theoretical guarantees for gradient-based optimization in overparameterized neural networks. These works show that sufficiently wide networks can achieve global convergence in training, with smooth and convex-like loss landscapes that avoid problematic local minima.

The work by Arora et al. (6) further demonstrated that overparameterized networks implicitly perform regularization through their architecture and training procedures. They showed that gradient descent with early stopping effectively performs ridge regression in the neural tangent kernel regime.

2.3 Applications in Computer Vision and Remote Sensing

In the domain of computer vision, He et al. (7) introduced ResNet architectures that achieved superior performance despite enormous parameter counts. Similarly, Huang et al. (8) proposed DenseNet, which further increased parameter count through dense connectivity patterns while improving generalization.

For remote sensing specifically, Ma et al. (9) evaluated deep learning models of varying complexities on satellite imagery classification tasks, finding that deeper architectures generally outperformed simpler ones when sufficient data was available. Sumbul et al. (10) introduced BigEarthNet, a large-scale benchmark dataset for remote sensing image classification and demonstrated the benefits of transfer learning from overparameterized models pre-trained on natural images.

2.4 Pruning and Model Compression

Han et al. (11) demonstrated that neural networks could be pruned by removing small-weight connections without significant accuracy loss, suggesting substantial redundancy in overparameterized

models. This was extended by Li et al. (12), who proposed filter-level pruning for convolutional networks based on filter importance measures.

Frankle and Carbin (13) proposed the "lottery ticket hypothesis," suggesting that overparameterized networks contain sparse subnetworks that can be trained from initialization to achieve performance comparable to the original network. This provides another perspective on why overparameterization helps during training but might not be necessary for final model performance.

2.5 Optimizers and Generalization

The choice of optimizer has been shown to significantly affect generalization in deep networks. Keskar et al. (14) demonstrated that stochastic gradient descent (SGD) tends to find flatter minima that generalize better compared to batch methods. Wilson et al. (15) analyzed how adaptive methods like Adam can converge to different solutions than SGD, often with worse generalization despite faster convergence.

Zhang et al. (16) specifically studied the interplay between model size and optimizer choice, finding that adaptive optimizers like Adam benefit more from overparameterization than SGD in terms of generalization performance, particularly in early training stages.

2.6 Gap in Literature

While substantial research exists on overparameterization in general deep learning contexts, fewer studies have systematically investigated the interplay between model size, optimizer choice, and pruning specifically in the domain of satellite image classification. This project aims to bridge this gap by providing empirical evidence on how these factors interact in the context of remote sensing data, which presents unique challenges due to its spectral characteristics, limited labeled samples, and high data variability.

3 Methodology

3.1 Network Architectures

Three convolutional neural networks of varying capacity: SmallNet, MediumNet, and LargeNet are designed.

SmallNet serves as the baseline model and consists of a shallow architecture with fewer convolutional layers and filters. It includes two convolutional layers followed by ReLU activations and max pooling, and ends with a fully connected layer.

MediumNet increases model complexity by adding more filters and an additional convolutional layer. It strikes a balance between computational efficiency and representational power.

LargeNet is a significantly deeper network with a higher number of filters, more convolutional layers, and additional fully connected layers. It is purposefully overparameterized relative to the size of the training dataset.

3.2 Dataset, Preprocessing, Basic Setup and Evaluation Metrics

The experiments are conducted on the EuroSAT dataset, which consists of 27,000 labeled satellite images spread across 10 land-use and land-cover classes (e.g., forest, residential, river). Each image is of size 64x64 with RGB channels.

Data preprocessing includes normalization to zero mean and unit variance, along with random horizontal flips and random crops for data augmentation. The dataset is split into (70%) for training, (15%) for validation, and (15%) for testing. The training was conducted for 30 epochs using a batch size of 64. The learning rate was set to 0.001 across all optimizers, and the same random seed was used for reproducibility.

Performance is evaluated using:

- Training loss (cross-entropy) to monitor optimization.

Table 1: Model Parameter Calculations

Component	Calculation	Parameters
SmallNet		
Conv1	$8 \times (3 \times 3 \times 3 + 1) = 8 \times 28$	224
FC1	$10 \times (8 \times 62 \times 62 + 1) = 10 \times 30,753$	307,530
Total		$\boxed{\sim 307,754}$
MediumNet		
Conv1	$16 \times (3 \times 3 \times 3 + 1)$	448
Conv2	$32 \times (16 \times 3 \times 3 + 1)$	4,640
FC1	$128 \times (6,272 + 1)$	802,944
FC2	$10 \times (128 + 1)$	1,290
Total		$\boxed{\sim 809,322}$
LargeNet		
Conv1	$64 \times (3 \times 3 \times 3 + 1)$	1,792
Conv2	$128 \times (64 \times 3 \times 3 + 1)$	73,856
Conv3	$256 \times (128 \times 3 \times 3 + 1)$	295,168
FC1	$512 \times (9,216 + 1)$	4,718,592
FC2	$128 \times (512 + 1)$	65,664
FC3	$10 \times (128 + 1)$	1,290
Conv Total		370,816
FC Total		4,785,546
Overall Total		$\boxed{\sim 5,156,362}$

Figure 1: Eurosat Dataset Classes

- Validation accuracy to assess generalization during training.
- Test accuracy to report final generalization performance.

3.3 Pruning Technique

Pruning is done to simplify MediumNet and LargeNet by removing unnecessary convolutional kernals by using L1-norm-based filter pruning is used to remove the least important convolutional filters based on their weight magnitudes. This is done to validate that overparameterization improves learning but may not be necessary for final model capacity.

4 Implementation and Experiments

4.1 Convergence

Convergence is analyzed by how quickly and smoothly the models' **training losses decrease over epochs**. The experiment tests if increasing model size improves gradient conditioning, leading to faster convergence in SGD-based optimization, reducing the number of training epochs required for a well-fitted model.

4.2 Generalization

The experiment tests that even if training accuracy (train acc) reaches near (100%), test and validation accuracy (test acc, val acc) remain high for LargeNet, indicating generalization, not memorization. The experiment does the following:

- Compare train acc vs test acc for each model.
- If LargeNet has near-zero training loss and high test accuracy, it supports this claim.

Generalization Gap = Train Accuracy − Test Accuracy

4.3 Regularization

The experiment tests if large models naturally exhibit regularization effects without explicit constraints, helping them achieve better stability, margin maximization, and robustness to small input variations. Stable train loss and consistent val acc suggest regularization effects even without dropout or weight decay.

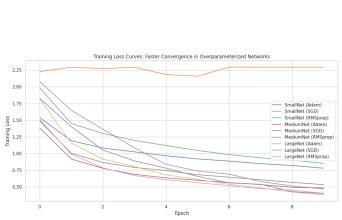
4.4 Parameter Pruning

After pruning, model size decreases but test accuracy stays nearly the same, showing redundancy in parameters.

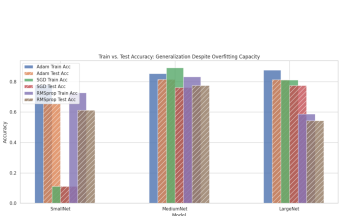
4.5 Optimization Behavior

Comparing SGD, Adam, and momentum-based optimizers will reveal how different optimization strategies interact with overparameterization and affect training stability, generalization, and loss landscape smoothness.

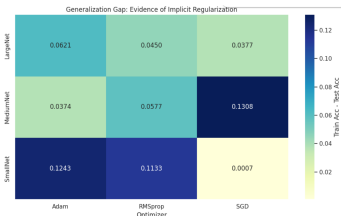
The experiment compares train loss list and test acc list across optimizers for a fixed model size.



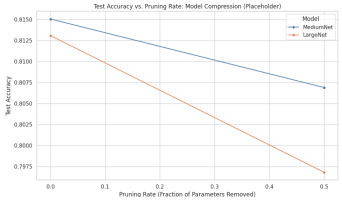
(a) Training Loss Curves



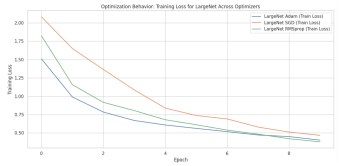
(b) Train vs. Test Accuracy



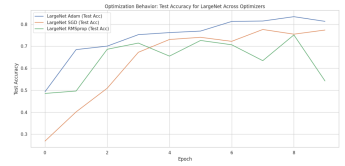
(c) Generalization Gap



(d) Test Accuracy vs. Pruning Rate



(e) Optimization Behavior



(f) Optimization Behavior

Figure 2: Experimental results showing (top row) training dynamics and generalization behavior, and (bottom row) pruning effects and optimization performance

5 Analysis of Experiments

5.1 Convergence

As seen in the training loss curves, larger models like MediumNet and LargeNet converge significantly faster than SmallNet, particularly with Adam and RMSprop optimizers. This empirically supports the theoretical claim that overparameterized models benefit from improved gradient conditioning, enabling efficient training with fewer epochs.

5.2 Generalization

Despite the increased parameter count in MediumNet and LargeNet, the models retain strong test accuracy, illustrating that overparameterized networks can still generalize well. Interestingly, SmallNet struggles to achieve meaningful performance, reinforcing the idea that larger models benefit from better optimization landscapes and implicit regularization, especially with optimizers like Adam.

Table 2: Train and Test Accuracy for Different Networks and Optimizers

Model	Optimizer	Train Accuracy	Test Accuracy
SmallNet	Adam	0.75	0.65
SmallNet	SGD	0.12	0.12
SmallNet	RMSprop	0.72	0.61
MediumNet	Adam	0.85	0.82
MediumNet	SGD	0.90	0.76
MediumNet	RMSprop	0.84	0.77
LargeNet	Adam	0.88	0.81
LargeNet	SGD	0.82	0.77
LargeNet	RMSprop	0.59	0.54

5.3 Regularization

The following observations are made from Figure 2 (c) Generalization Gap Heatmap

- SGD generalizes best on smaller models like SmallNet, possibly due to its implicit regularization effects.
- Adam and RMSprop may lead to more overfitting on small models.
- Larger networks can generalize well despite having the capacity to overfit—supporting the idea of implicit regularization through overparameterization.

5.4 Parameter Pruning

Pruning had minimal impact on performance. The results suggest that pruning didn't hurt generalization (accuracy), but also didn't reduce model size. LargeNet loses more accuracy with pruning, which aligns with the exploration of redundancy in overparameterized models. The smaller drop for MediumNet might indicate it has less redundant capacity.

5.5 Optimization Behavior

Adam achieves the highest test accuracy (0.8), followed by RMSprop (0.75) and SGD (0.7), suggesting Adam is the most effective optimizer for LargeNet in this context. The higher accuracy with Adam aligns with its adaptive learning rate, which may better handle the complexity of LargeNet, while SGD's slower convergence and RMSprop's instability reflect their behavior in overparameterized settings.

6 Conclusion

In this study, investigations of overparameterization and different optimization strategies influence the convergence, generalization, and efficiency of deep neural networks in the context of satellite

image classification was conducted. By implementing and comparing three architectures—SmallNet, MediumNet, and LargeNet—across optimizers such as SGD, Adam, and RMSprop, it has been observed that larger networks not only converged faster but also exhibited stronger generalization, even when trained to zero training error. These findings align with theoretical claims around implicit regularization and the double descent phenomenon. While unstructured pruning led to minimal degradation in performance, it highlighted the redundancy present in overparameterized models, suggesting the potential for compression without significant accuracy loss. The findings provide deeper insights into how overparameterization affects real-world applications such as land-use classification, environmental monitoring, and remote sensing analysis.

References

[1] Vapnik, V., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and Its Applications, 16(2), 264-280.

[2] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. International Conference on Learning Representations (ICLR).

[3] Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proceedings of the National Academy of Sciences, 116(32), 15849-15854.

[4] Du, S. S., Lee, J. D., Li, H., Wang, L., & Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. International Conference on Machine Learning (ICML).

[5] Allen-Zhu, Z., Li, Y., & Song, Z. (2019). A convergence theory for deep learning via over-parameterization. International Conference on Machine Learning (ICML).

[6] Arora, S., Du, S. S., Hu, W., Li, Z., & Wang, R. (2019). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. International Conference on Machine Learning (ICML).

[7] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[8] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[9] Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., & Johnson, B. A. (2019). Deep learning in remote sensing applications: A meta-analysis and review. ISPRS Journal of Photogrammetry and Remote Sensing, 152, 166-177.

[10] Sumbul, G., Charfuelan, M., Demir, B., & Markl, V. (2019). BigEarthNet: A large-scale benchmark archive for remote sensing image understanding. IEEE International Geoscience and Remote Sensing Symposium (IGARSS).

[11] Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. Advances in Neural Information Processing Systems (NeurIPS).

[12] Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017). Pruning filters for efficient convnets. International Conference on Learning Representations (ICLR).

[13] Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. International Conference on Learning Representations (ICLR).

[14] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. International Conference on Learning Representations (ICLR).

[15] Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. Advances in Neural Information Processing Systems (NeurIPS).

[16] Zhang, J., He, T., Sra, S., & Jadbabaie, A. (2020). Why gradient clipping accelerates training: A theoretical justification for adaptivity. International Conference on Learning Representations (ICLR).

A Appendix / supplemental material

Code Snippets

A.1 SmallNet Architecture

```
class SmallNet(nn.Module):
    def __init__(self):
        super(SmallNet, self).__init__()
        self.conv1 = nn.Conv2d(3, 8, kernel_size=3)
        self.fc1 = nn.Linear(8 * 62 * 62, 10)
    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = x.view(x.size(0), -1)
        return self.fc1(x)
```

A.2 Training Loop

```
def train_model(model, optimizer, criterion, train_loader, val_loader, test_loader, 
    model.to(device)
    for epoch in range(epochs):
        model.train()
        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
```

A.3 Accuracy Evaluation

```
def evaluate_accuracy(model, dataloader):
    model.eval()
    correct = total = 0
    with torch.no_grad():
        for inputs, labels in dataloader:
            outputs = model(inputs.to(device))
            _, preds = torch.max(outputs, 1)
            correct += (preds == labels.to(device)).sum().item()
    return correct / len(dataloader.dataset)
```

A.4 L1-Norm Pruning

```
def apply_global_pruning(model, amount=0.2):
    for name, module in model.named_modules():
        if isinstance(module, nn.Conv2d):
            weight = module.weight.data.abs().mean(dim=(1, 2, 3))
            threshold = torch.quantile(weight, amount)
            mask = (weight > threshold).float().view(-1, 1, 1, 1)
            module.weight.data *= mask
            if module.bias is not None:
                module.bias.data *= mask.view(-1)
    return model
```