Training From Scratch vs Transfer Learning

Anirudh Belaguppa Manjunath, Sriharhsa Patallipalli

Abstract

This study examines the performance of various deep learning models on a small, custom dataset of 1,800 images across six categories. We compare the efficacy of training models from scratch against using pretrained architectures, specifically ResNet-18, VGG-19, and Inception V3. Our results suggest that pretrained models significantly outperform models trained from scratch, offering better accuracy with less computational expense, particularly in data-constrained environments. The implications of these findings advocate for the strategic use of transfer learning in similar small-scale data settings.

1 Introduction

Deep Learning has revolutionized the field of machine learning, providing substantial advancements in processing large datasets with complex structures. Despite these innovations, a significant challenge persists when dealing with smaller datasets. Training deep neural networks on limited data often leads to overfitting, where a model learns the details and noise in the training data to an extent that it negatively impacts the performance of the model on new data.

The primary objective of this study is to explore and compare the performance of various deep learning architectures on a small dataset consisting of 1,800 images spread across six categories. This project specifically examines the efficacy of training models from scratch versus utilizing pretrained models. The models under consideration include ResNet-18, VGG-19, and Inception V3, each representing distinct characteristics and complexities.

This investigation is significant as it addresses the common scenario in applied machine learning where the acquisition of large-scale data is impractical. By understanding the performance of different architectures under data constraints, this study aims to provide insights that can guide the deployment of deep learning models in similar settings effectively. Furthermore, it explores the potential of transfer learning to mitigate the challenges posed by limited data, thus broadening the applicability of advanced neural network models to a wider range of problems.

2 Dataset Description

The dataset comprises 1,800 images of clothing articles, uniformly distributed across six categories: Jeans, T-Shirts, Shirts, Jackets, Sweaters, and Tops. This equal distribution helps mitigate class imbalance, which is crucial for maintaining generalization and preventing bias in model training outcomes.

2.1 Data Collection

The images were systematically collected using RSS feeds that aggregate content related to fashion and clothing. Only images explicitly discussing or depicting the designated categories were considered. This method ensures that the dataset is not only relevant but also rich in variety pertaining to the context of clothing articles.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2.2 Data Preparation

Upon collection, the images underwent a preprocessing phase where a pre-trained convolutional neural network (CNN) was employed to detect and extract regions specifically containing the clothing articles. This step was crucial to focus the model training on relevant features without the noise and background distractions typically present in raw web-scraped images.

This focused extraction not only enhances the quality of the dataset but also aligns with the objective of evaluating model performance specifically on clothing article recognition tasks. The final dataset is formatted with images resized to a standard dimension, which is compatible with the input requirements of the deep learning models used, namely ResNet-18, VGG-19, and Inception V3.

3 Model Selection

Three widely-used convolutional neural network architectures were chosen for this study to evaluate their performance on the clothing article dataset: ResNet-18, VGG-19, and Inception V3. These models were selected to cover a broad range of complexities and capacities to provide insights into their adaptability and efficiency in handling small datasets.

3.1 Training Methodology

Each model was trained under two different settings to compare their abilities to adapt and learn from a small dataset:

1. **Full Model Training:** Each model was trained end-to-end, which involves adjusting all the parameters across all the layers. This approach is parameter-intensive and explores the model's capacity to learn features specific to the dataset from scratch.
2. **Classifier Layer Training:** Only the classifier layers were trained, where all other layers were frozen. This method primarily leverages transfer learning, where the pre-trained features are adapted slightly to the new dataset. It is computationally less demanding and quicker than full model training.

These training approaches are pivotal in understanding the practicality of deploying these models in scenarios with limited data and computational resources. The outcomes not only reflect the models' efficiency but also their scalability and ease of integration into existing systems.

4 Implementation Approach

The implementation phase of this project was structured around setting up the necessary computational tools and defining the procedural workflow to efficiently train and evaluate the selected models on the clothing dataset.

4.1 Computational Resources

Due to the high computational demands of training deep neural networks, particularly when using models as large and complex as VGG-19 and Inception V3, access to robust computational resources was critical. A NVIDIA Quadro P5000 GPU, rented from PaperSpace, was utilized to handle the intensive computations involved in training and testing the models. This GPU provided the necessary power to process the models efficiently, thereby significantly reducing training time and enabling more extensive experimentation.

4.2 Software and Libraries

The project was implemented using Python, chosen for its extensive support and rich ecosystem of libraries for data science and machine learning. The following libraries were instrumental in the project:

- **PyTorch:** Used for building and training the convolutional neural networks. It provided the flexibility needed for custom layer definitions and efficient model training.

- **Scikit-Learn:** Utilized for model evaluation, particularly for calculating the accuracy metrics and performing other statistical analysis.
- **NumPy and Pandas:** These libraries were used for handling data manipulation and preprocessing, given their powerful and flexible data structures.

4.3 Preprocessing and Data Augmentation

Data preprocessing and augmentation are critical steps in ensuring that the models generalize well on new, unseen data. The following transforms were applied to the dataset before training:

- Random rotations, translations, and flips to introduce variability into the dataset, simulating real-world scenarios where clothing items can be oriented in multiple ways.
- Normalization of image pixel values to facilitate model convergence during training.

These techniques not only augment the data but also help in preventing overfitting, making the models more robust.

4.4 Training Parameters

The models were trained using the Adam optimizer with a learning rate of 0.0001. This optimizer adjusts the learning rate throughout training, which helps in settling at the best possible values of model parameters more efficiently. Each model was trained over 50 epochs with a batch size of 32. This setup provided a good balance between training speed and model performance, enabling the extensive evaluation required to compare the different architectural approaches comprehensively.

4.5 Evaluation Metrics

To comprehensively assess the performance of the selected models, several metrics were utilized:

- **Accuracy**: This metric provides a baseline evaluation of the models by indicating the overall proportion of correct predictions.
- **Precision, Recall, and F1 Score**: These metrics help to evaluate the models in terms of their capacity to classify the images correctly and the harmony between precision and recall, particularly useful given the balanced nature of the dataset across classes.
- **AUC-ROC**: While typically used in binary classification, ROC curves can be extended to multi-class classification, providing insight into the effectiveness of the models at various threshold levels.

These metrics collectively enable a robust evaluation of the models, highlighting strengths and weaknesses that are not apparent through accuracy alone.

5 Results and Evaluation

This section tabulates the performance of three convolutional neural network models: Inception V3, ResNet-18, and VGG-19, which were each trained from scratch and then fine-tuned. The models were assessed based on metrics including F1 score, accuracy, precision, and recall, both on the training and testing datasets.

5.1 Performance Metrics

The following table summarizes the results from the test datasets across different training conditions:

5.2 Comparative Analysis

5.2.1 Impact of Fine-tuning

Fine-tuning the classifier layers significantly enhances the performance metrics (F1 score, accuracy, precision, and recall) across all models. This method effectively utilizes the pre-learned features from large datasets, which is especially beneficial in scenarios with limited data availability.

Table 1: Summary of Model Performances

Model	Condition	F1 Score	Accuracy	Precision	Recall
Inception V3	Scratch	0.7506	75.44%	0.7543	0.7536
	Fine-tuned	0.8270	83.33%	0.8271	0.8343
ResNet-18	Scratch	0.7245	71.94%	0.7485	0.7216
	Fine-tuned	0.8285	83.16%	0.8298	0.8299
VGG-19	Scratch	0.7503	74.44%	0.7491	0.7508
	Fine-tuned	0.8685	86.66%	0.8823	0.8676

5.2.2 Model Efficiency

- **Inception V3** demonstrated the most considerable improvement in generalization capabilities upon fine-tuning, indicating its efficient feature utilization from pre-trained states.
- **ResNet-18** showed the highest relative increase in performance metrics, suggesting that its architecture might be particularly well-suited for adaptation through fine-tuning.
- **VGG-19**, despite its initial lower performance when trained from scratch, achieved competitive metrics post fine-tuning, underscoring the deep architecture's potential when optimized appropriately.

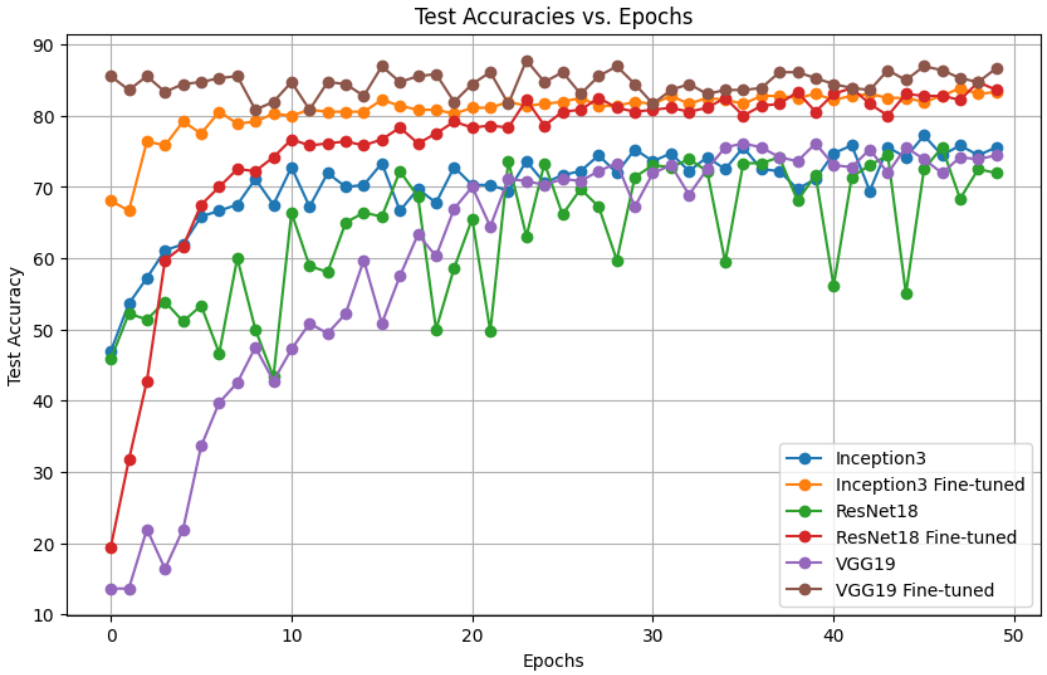

Figure 1: Comparitive Analysis of Test Accuracies with models

5.3 Discussion

These results affirm the pivotal role of choosing the right model and training strategy tailored to the specific needs of the dataset and computational constraints. The clear advantages of fine-tuning in this context highlight its utility in enhancing model performance efficiently, which could be crucial for practical deployments in resource-constrained environments.

References

[1] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[3] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.