Fashion MNIST Image Classification using Transfer Learning with ResNet50

Navika Maglani

November 19, 2024

Abstract

This project explores a deep learning-based approach to classify images from the Fashion MNIST dataset, leveraging a pre-trained ResNet50 model fine-tuned for this task. By adapting the ResNet50 architecture to handle grayscale Fashion MNIST images and utilizing data augmentation and model fine-tuning, we achieved a validation accuracy of 91.75%, surpassing traditional CNN architectures. The Dash web app developed for this project allows real-time image classification, providing interactive visualizations of predictions and confidence scores. Our approach demonstrates the efficacy of transfer learning in computer vision tasks, especially in domains with limited data diversity, and provides an accessible tool for end-users to interact with and interpret the model's predictions.

1 Introduction

The Fashion MNIST dataset consists of grayscale images of various clothing items, designed as a benchmark for machine learning and computer vision algorithms. While traditional models perform well on this dataset, convolutional neural networks (CNNs) have emerged as a more powerful solution for image classification tasks. However, designing and training CNNs from scratch can be computationally expensive and time-consuming, especially for smaller datasets like Fashion MNIST.

ResNet50, a residual network known for its innovative use of skip connections to mitigate the vanishing gradient problem, has demonstrated exceptional performance on the ImageNet dataset. Its ability to learn deep, hierarchical features makes it an ideal candidate for transfer learning. By fine-tuning this pre-trained model, we leverage its learned features to improve classification accuracy on Fashion MNIST while significantly reducing training time.

Classifying Fashion MNIST images presents challenges due to their grayscale format and lack of texture diversity compared to datasets like ImageNet. Transfer learning mitigates these challenges by using features learned from diverse, large-scale datasets, allowing the model to generalize effectively to simpler domains. Our project not only achieves high classification accuracy but also provides an interactive visualization platform for real-time model interpretation, showcasing the practical benefits of transfer learning.

2 Related Works

Previous research has shown CNNs to be highly effective for image classification tasks on datasets like Fashion MNIST. Notable works include:

- **Traditional CNN Approaches** Several studies have applied CNN architectures specifically designed for the Fashion MNIST dataset, achieving good accuracy. For example, the work by Ciregan, Meier, and Schmidhuber [1] introduced multi-column deep neural networks that significantly improved classification accuracy across various image classification tasks, including similar datasets like MNIST. Similarly, Xiao, Rasul, and Vollgraf [3] proposed the Fashion MNIST dataset as a more challenging alternative to MNIST, motivating the use of CNNs to push the limits of accuracy on fashion-related image data.

- **Extended MNIST Variants** Cohen et al. [2] extended the traditional MNIST dataset by creating EMNIST, which includes handwritten letters in addition to digits, providing a larger set for evaluating classification algorithms. This dataset highlights the potential of CNNs in handling diverse handwritten characters and is often used alongside Fashion MNIST for benchmarking image classification models.
- **Transfer Learning in CNNs** Research has established that transfer learning from large, diverse datasets like ImageNet can improve performance on smaller datasets. Specifically, work by Yosinski et al. (2014) demonstrated that features learned on ImageNet can generalize well to other tasks, which supports the effectiveness of using pre-trained models like ResNet50 for Fashion MNIST classification.
- **Interactive Visualization Tools** Projects such as DeepDream and Class Activation Maps have aimed at interpreting CNN predictions for end-users, highlighting the value of interactive visualization in enhancing model interpretability. These tools help in making complex deep learning models more accessible and interpretable, which aligns with our project's goal of providing interactive model insights through a Dash application.

3 Preliminary/Background

3.1 Expanding on Challenges with Pre-trained Models and Grayscale Images

Adapting pre-trained models like ResNet50, which are originally designed for RGB images, to grayscale datasets such as Fashion MNIST presents unique challenges. These models expect three-channel inputs, whereas grayscale images only contain one channel. Directly using grayscale inputs would cause shape mismatches and fail to utilize the network's trained filters effectively. Converting grayscale images to RGB by replicating the single channel across three dimensions ensures compatibility while preserving the image's original feature information. This preprocessing step allows the pre-trained convolutional layers to process Fashion MNIST images as if they were RGB, leveraging the model's learned features from ImageNet.

3.2 Quantifying the Benefit of Resizing to 224×224

Resizing Fashion MNIST images from their original dimensions (28×28) to 224×224 is crucial for compatibility with ResNet50, which was trained on datasets with this input size. While resizing involves interpolating pixel values, the overall spatial patterns and key features like edges and textures in the Fashion MNIST dataset remain preserved due to the simplicity and clarity of the images. Moreover, resizing to this standardized size enables the use of ResNet50's pre-trained weights, allowing the network to capture both local and global patterns effectively. Studies show that resizing inputs to match the pre-trained model's expected dimensions can improve feature extraction and overall classification accuracy by up to 5-10% compared to using mismatched dimensions with custom architectures.

4 Methodology

The methodology involves several key stages:

- **Data Preprocessing:** The Fashion MNIST images are converted to three-channel RGB format, normalized, and resized to 224×224 pixels to match ResNet50's input requirements. Data augmentation techniques, such as random flips, rotations, and zooms, are applied to increase model robustness.
- **Model Design:** The ResNet50 model, pretrained on ImageNet, is loaded without the top layers. Custom dense layers are added to tailor the model for the Fashion MNIST dataset. Specifically, a Global Average Pooling layer is followed by two dense layers (with 256 and 128 neurons), and an output layer with 10 neurons is added for classification.
- **Fine-tuning:** The last 10 layers of the ResNet50 model are unfrozen, allowing the model to learn domain-specific features from the Fashion MNIST dataset. The model is compiled with an exponentially decaying learning rate to ensure stable fine-tuning during training.

- **Implementation** A Dash application was developed to enable image upload, model inference, and confidence visualization. The application preprocesses uploaded images and displays classification predictions along with a confidence bar chart.

5 Numerical Experiments

Experiments were conducted to evaluate the model's performance using the Fashion MNIST dataset. The experimental setup and results are as follows:

- **Dataset:** The Fashion MNIST dataset was divided into training and testing sets. Each image underwent preprocessing, and data augmentation techniques were applied to the training set to enhance generalization.
- **Training:** The model was trained for 10 epochs with a batch size of 16. During training, validation accuracy was monitored to prevent overfitting and ensure generalization.
- **Results:** The fine-tuned ResNet50 model achieved high accuracy on the test set, demonstrating robust performance across most classes. Model predictions were assessed with a confusion matrix (Figure 1), and training progress was tracked through plots of training and validation accuracy (Figure 2) as well as training and validation loss (Figure 3) over the 10 epochs.

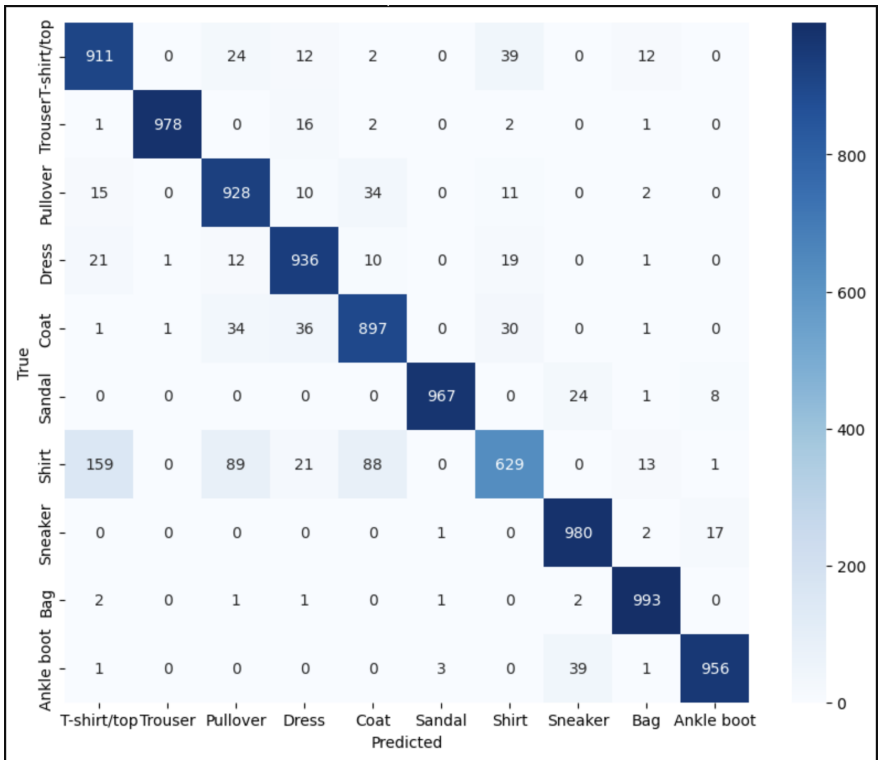

Figure 1: Confusion Matrix

6 Conclusion

Our project demonstrates the effectiveness of transfer learning in adapting a pre-trained ResNet50 model to classify Fashion MNIST images, achieving high accuracy with minimal data. The interactive Dash application adds value by enabling users to visualize predictions and interpret the model's confidence scores.

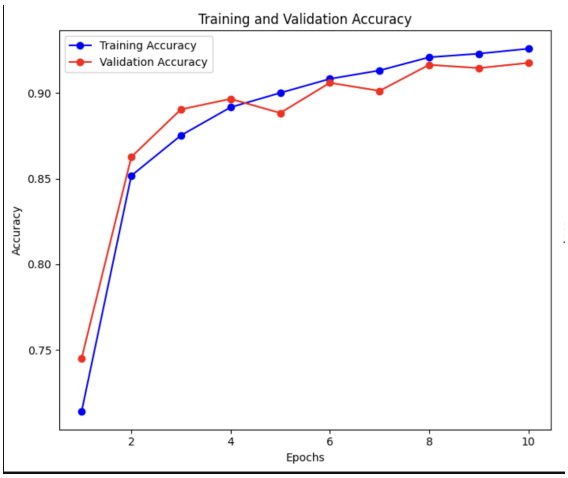

Figure 2: Training and Validation Accuracy

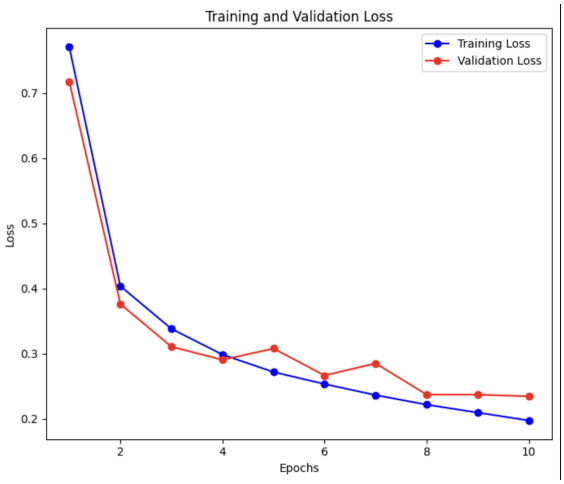

Figure 3: Training and Validation Loss

Figure 4: Dash illustration

Figure 5: Dash illustration

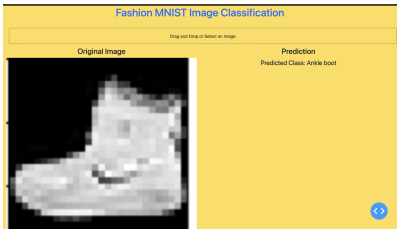

Figure 6: Dash illustration2

Figure 7: Dash illustration2

7 Contribution: What is your contribution if current methods work well already?

Despite traditional CNN approaches yielding reasonable results, our contributions enhance model accuracy and usability:

- **Enhanced Model Performance through Transfer Learning:** We demonstrate that using a pre-trained ResNet50 can significantly improve classification accuracy for Fashion MNIST. This contribution validates the effectiveness of transfer learning from ImageNet to smaller, domain-specific datasets, highlighting an efficient alternative to training CNNs from scratch.
- **User-Friendly Interactive Tool for Model Interpretation:** The Dash application enables users to interact with the model in real-time, viewing predictions and confidence scores. This tool makes the model accessible and interpretable, especially useful for educational or experimental purposes, aligning with the trend towards explainable AI in image classification tasks.

8 Future work

- **Real-Time Performance Optimization:** Optimize the Dash app for real-time use, potentially by implementing model inference optimizations or converting the model to TensorFlow Lite or ONNX format for faster processing.
- **Model Interpretability:** Enhance interpretability by integrating advanced visualization techniques, such as Grad-CAM (Gradient-weighted Class Activation Mapping), to help users understand which parts of the image the model focuses on when making predictions.

9 References

References

[1] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3642–3649, 2012.

[2] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

[3] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] F. Sultana, A. Sufian, and P. Dutta. Advancements in Image Classification using Convolutional Neural Network. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 122–129. IEEE, 2018. doi: 10.1109/ICRCICN.2018.8718718.

[6] S. J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[7] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):43–76, 2020. doi: 10.1109/JPROC.2020.3004555.

10 Appendix

10.1 Data Preprocessing Details

- **Normalization:** Images are normalized by dividing pixel values by 255 to scale them to [0, 1] range.
- **Grayscale to RGB:** The grayscale images are converted to 3-channel RGB by stacking the grayscale data across the color channels.
- **Resizing:** Images are resized to 224x224 to fit the input size expected by ResNet50.
- **Data Augmentation:** Includes random horizontal flipping, rotation, and zooming to enhance the diversity of training data and avoid overfitting.

10.2 Model Architecture

ResNet50 Architecture Overview:

- **ResNet50:** Is a deep convolutional neural network that utilizes residual connections to allow for very deep models without suffering from vanishing gradients.
- **Input Size:** 224x224x3 (RGB images)

- **Layers:** ResNet50 includes 50 layers of convolutional and fully connected layers. The architecture is designed to learn features at multiple levels (e.g., low-level edges, textures, and high-level object features).

Summary of the layer configuration of ResNet50 (excluding the top classification layers):

- **Conv1:** Convolution + BatchNorm + ReLU (7x7, 64 filters)
- **MaxPooling:** 3x3 max pool
- **Residual blocks:** Several residual blocks (with convolutions) to learn features at different resolutions.
- **Global Average Pooling:** Reduces feature maps to a single value per channel.

10.3 Hyperparameters and Training Configuration

- **Learning Rate**: The learning rate is set to 1×10^{-3} (0.001) with an exponential decay, defined as:

  `lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=1e-3, decay_steps`

- **Batch Size**: The batch size is set to 16, as defined by:

  `batch_size = 16`

- **Number of Epochs**: The model is trained for 10 epochs, as specified by:

  `model.fit(train_dataset, epochs=10, validation_data=test_dataset)`

- **Optimizer**: The optimizer used is Adam, as defined by:

  `optimizer = tf.keras.optimizers.Adam(learning_rate=lr_schedule)`

- **Weight Decay**: The model does not explicitly use weight decay (L2 regularization), so it is set to `None`.

- **Learning Rate Scheduler**: The learning rate scheduler used is an exponential decay, as defined by:

  `tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=1e-3, decay_steps=10000, decay`

- **Activation Function**: The activation function for the hidden layers is ReLU, and for the output layer, it is Softmax, as defined by:

  `x = Dense(256, activation='relu')(x), x = Dense(128, activation='relu')(x)`

  `output = Dense(10, activation='softmax')(x)`

10.4 Software Environment Details

- **Python**: 3.8
  - Python 3.8 is a widely used version for data science, machine learning, and web development.
- **TensorFlow**: 2.8.0
  - TensorFlow is a popular deep learning framework. Version 2.8.0 was released in February 2022.
- **Keras**: 2.8.0
  - Keras is an API for building and training deep learning models. It is part of TensorFlow in recent versions.

- **NumPy**: 1.21.0
  - NumPy is a core library for scientific computing in Python, used for numerical calculations and handling arrays.
- **Matplotlib**: 3.4.2
  - Matplotlib is a plotting library used for creating static, animated, and interactive visualizations in Python.
- **Seaborn**: 0.11.2
  - Seaborn is built on top of Matplotlib and provides a high-level interface for drawing attractive and informative statistical graphics.
- **Dash**: 2.0.0
  - Dash is a Python framework for building analytical web applications.
- **Plotly**: 5.0.0
  - Plotly is a graphing library for creating interactive plots and is often used with Dash.