Comparative Analysis of Convolutional Neural Networks and Random Forest Classifiers for Fashion-MNIST Image Classification

Research Report

Course: Machine Learning

Course of Study: Computer Science

Author: Student Name

Matriculation Number: 123456789

Tutor: Tutor Name

Date: January 2026

Contents

1 Introduction **2**

1.1 Background . . . 2

1.2 Problem Statement . . . 2

1.3 Research Objectives . . . 2

1.4 Report Structure . . . 3

2 Literature Review **4**

2.1 Fashion-MNIST Dataset and Benchmarks . . . 4

2.2 Random Forest in Image Classification . . . 4

2.3 Convolutional Neural Networks for Fashion Classification . . . 4

3 Methodology **6**

3.1 Dataset Description and Preprocessing . . . 6

3.1.1 Random Forest Preprocessing . . . 6

3.1.2 CNN Preprocessing and Augmentation . . . 6

3.2 Random Forest Classifier Implementation . . . 6

3.3 Convolutional Neural Network Architecture . . . 7

3.3.1 Architecture Specifications . . . 7

3.4 Training Strategy . . . 7

3.4.1 Random Forest Training . . . 7

3.4.2 CNN Training . . . 7

3.5 Evaluation Protocol . . . 8

4 Results **9**

4.1 Overall Performance Comparison . . . 9

4.2 Per-Category Performance Analysis . . . 9

4.3 Category-Specific Performance Patterns . . . 10

4.3.1 Best Performing Categories . . . 10

4.3.2 Challenging Categories . . . 10

4.4 Training Time Analysis . . . 10

4.5 Confusion Matrix Analysis . . . 11

4.5.1 Random Forest Confusion Patterns . . . 11

4.5.2 CNN Confusion Patterns . . . 11

5 Analysis and Discussion **12**

5.1 Performance Analysis . . . 12

5.2 Analysis of Worst-Performing Categories . . . 12

5.2.1 Shirt Classification Challenges . . . 12

5.2.2 Pullover and Coat Confusion . . . 12

5.3 Generalization and Robustness . . . 13

5.4 Computational Trade-offs . . . 13

5.5 Production Deployment Recommendation . . . 13

6 Conclusion **15**

References **16**

A Random Forest Implementation Code **17**

B CNN Implementation Code **18**

C MCNN15 Architecture Definition **20**

List of Figures

1 Performance summary comparing CNN and Random Forest classifiers 1

List of Tables

1 MCNN15 Architecture Details 7
2 Overall Performance Comparison 9
3 Per-Category Precision and Recall on Test Set 9
4 Training Time Comparison 10

Abstract

This study presents a comprehensive comparison between two machine learning approaches for fashion product classification using the Fashion-MNIST dataset: a Convolutional Neural Network (CNN) and a Random Forest classifier. The Fashion-MNIST dataset, introduced by Zalando in 2017, contains 70,000 grayscale images of 10 fashion categories including T-shirts, trousers, dresses, and footwear.

Our experiments demonstrate that the CNN architecture (MCNN15) achieved significantly superior performance with 93.63% test accuracy compared to the Random Forest classifier's 87.52% test accuracy. While the CNN required approximately 900 seconds of training time across 50 epochs, the Random Forest completed training in only 9.45 seconds. Both classifiers exhibited similar performance patterns across categories, with shirt classification proving most challenging due to high visual similarity with other upper-body garments.

The CNN's superior accuracy, despite longer training time, makes it the recommended approach for production deployment in fashion retail automation systems. The substantial accuracy improvement of 6.11 percentage points represents a reduction of approximately 611 misclassified items per 10,000 products, which can significantly impact inventory management and customer experience in large-scale retail operations.

Keywords: Fashion-MNIST, Convolutional Neural Networks, Random Forest, Image Classification, Deep Learning, Machine Learning

Graphical Abstract

Method	Accuracy	Training Time
CNN (MCNN15)	93.63%	900 seconds
Random Forest	87.52%	9.45 seconds

Figure 1: Performance summary comparing CNN and Random Forest classifiers

The graphical abstract presents the key findings of this comparative study. The CNN architecture achieved 6.11 percentage points higher accuracy than Random Forest, at the cost of approximately 95 times longer training time. Both methods identify shirts as the most challenging category due to visual similarity with T-shirts and pullovers.

1 Introduction

1.1 Background

The fashion retail industry has experienced significant digital transformation, with automated product classification becoming increasingly critical for inventory management, customer experience, and operational efficiency. Traditional manual categorization of fashion items is labor-intensive, error-prone, and does not scale with the volume of products in modern e-commerce platforms. The introduction of the Fashion-MNIST dataset by Xiao et al. (2017) provided a standardized benchmark for evaluating machine learning algorithms on fashion product classification tasks.

Fashion-MNIST serves as a more challenging replacement for the classic MNIST handwritten digit dataset, offering similar structure while presenting realistic computer vision challenges relevant to the fashion industry. The dataset contains 70,000 grayscale images of size 28$\times$28 pixels, equally distributed across 10 fashion categories: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. This balanced dataset enables fair evaluation of classification algorithms while maintaining computational efficiency for research purposes.

1.2 Problem Statement

The selection of appropriate machine learning approaches for fashion classification involves balancing multiple factors including classification accuracy, training time, computational requirements, and generalization capabilities. Traditional machine learning methods like Random Forest have demonstrated strong performance on structured data and can serve as baseline comparisons for more complex deep learning approaches. Conversely, Convolutional Neural Networks have shown exceptional performance on image classification tasks by automatically learning hierarchical feature representations directly from pixel data.

This study addresses the critical question of algorithm selection for production fashion classification systems by providing a rigorous comparison between Random Forest and CNN approaches. We evaluate both methods using identical evaluation protocols, analyze their strengths and limitations across different fashion categories, and provide evidence-based recommendations for practical deployment in fashion retail environments.

1.3 Research Objectives

The primary objectives of this study are:

1. Implement and evaluate a Convolutional Neural Network (MCNN15 architecture) for Fashion-MNIST classification following TensorFlow's classification methodology
2. Implement and evaluate the optimal Random Forest classifier configuration identified by Xiao et al. (2017)
3. Compare classification performance through confusion matrices and precision/recall metrics
4. Analyze category-specific performance to identify challenging garment types
5. Compare training times and computational requirements

6. Provide recommendations for production deployment based on accuracy, computational cost, and practical constraints

1.4 Report Structure

This report is organized as follows: Section 2 presents a review of relevant literature on Fashion-MNIST, Random Forest classifiers, and Convolutional Neural Networks. Section 3 describes the methodology including dataset characteristics, preprocessing steps, model architectures, and training strategies. Section 4 presents the experimental results including performance metrics, confusion matrices, and category-specific analysis. Section 5 discusses the implications of these results, analyzes failure modes, and provides production deployment recommendations. Section 6 concludes with a summary of findings and suggestions for future work.

2 Literature Review

2.1 Fashion-MNIST Dataset and Benchmarks

Xiao et al. (2017) introduced Fashion-MNIST as a benchmark dataset specifically designed to replace MNIST for machine learning research. The authors demonstrated that Fashion-MNIST presents greater classification challenges compared to MNIST while maintaining the same image dimensions and dataset structure. Their baseline experiments using various algorithms established initial performance benchmarks, with Random Forest achieving 87.3% accuracy using 100 estimators with entropy criterion.

The dataset has since become a standard benchmark for evaluating image classification algorithms, with numerous studies proposing various architectures and training strategies. According to the official Fashion-MNIST benchmarks (Zalando Research, 2017), traditional machine learning models like Random Forest generally achieve between 87% and 88.5% accuracy. In contrast, standard CNNs consistently reach 91-93%, while state-of-the-art architectures can exceed 96% accuracy.

The fashion domain presents unique challenges including high intra-class variability, subtle inter-class differences (particularly among upper-body garments), and the need for robust feature extraction from relatively low-resolution images. These characteristics make Fashion-MNIST an excellent testbed for comparing traditional machine learning approaches with deep learning methods.

2.2 Random Forest in Image Classification

Random Forest classifiers, introduced by Breiman (2001), have demonstrated strong performance across various machine learning tasks. The algorithm constructs multiple decision trees during training, using bagging and random feature selection to reduce variance and overfitting. In image classification contexts, Random Forests typically operate on extracted feature vectors rather than raw pixel data.

Previous work on Fashion-MNIST classification using Random Forests has consistently achieved accuracy rates in the 85-88% range (Xiao et al., 2017), establishing this as a strong baseline for comparison with more complex approaches. The primary limitation of Random Forests for image classification lies in their inability to automatically learn spatial feature hierarchies, requiring manual feature engineering or simple flattening of image data.

Despite these limitations, Random Forests offer significant advantages including fast training times, interpretability through feature importance analysis, and robustness to hyperparameter choices. These characteristics make them attractive for applications where training efficiency and model transparency are prioritized over maximum accuracy.

2.3 Convolutional Neural Networks for Fashion Classification

Convolutional Neural Networks, pioneered by LeCun et al. (1998), have revolutionized computer vision through their ability to automatically learn hierarchical feature representations. The MCNN15 architecture employed in this study is inspired by the work of Bhatnagar et al. (2017), who demonstrated that architectures with multiple convolutional and pooling layers significantly outperform traditional machine learning models on Fashion-MNIST.

Recent surveys of CNN architectures for Fashion-MNIST have reported accuracy rates ranging from 90-95%, with performance heavily dependent on architectural choices, training strategies, and regularization techniques. Key architectural innovations that improve performance include:

- **Batch Normalization:** Introduced by Ioffe and Szegedy (2015), this technique accelerates deep network training by reducing internal covariate shift, allowing higher learning rates and more stable training.
- **Data Augmentation:** Perez and Wang (2017) demonstrated that affine transformations (rotation, translation, scaling) act as powerful regularizers, improving model generalization by helping CNNs learn transformation-invariant features.
- **Adam Optimizer:** Kingma and Ba (2014) introduced Adam (Adaptive Moment Estimation), which computes adaptive learning rates for each parameter, combining advantages of AdaGrad and RMSProp for efficient training of deep networks.

The MCNN15 architecture specifically utilizes a deep stack of 15 convolutional layers organized into three hierarchical groups, with batch normalization following each convolutional operation. This design systematically reduces spatial dimensions while increasing feature complexity, enabling the network to capture both low-level texture patterns and high-level garment structures.

3 Methodology

3.1 Dataset Description and Preprocessing

The Fashion-MNIST dataset (Xiao et al., 2017) consists of 70,000 grayscale images divided into 60,000 training samples and 10,000 test samples. Each image measures 28$\times$28 pixels with pixel values in the range [0, 255]. The dataset maintains perfect class balance with 7,000 images per category in the complete dataset.

For the CNN implementation, the training set was further divided into 50,000 training samples and 10,000 validation samples using a random split with a fixed seed for reproducibility. The 10,000 test samples were reserved for final evaluation only. Pixel values were normalized to the range [0, 1] by dividing by 255.

3.1.1 Random Forest Preprocessing

For the Random Forest classifier, images were flattened into 784-dimensional feature vectors (28$\times$28 pixels) without additional preprocessing. This approach preserves all pixel information while converting the 2D spatial data into a format suitable for traditional machine learning algorithms. The flattened representation treats each pixel as an independent feature, which limits the classifier's ability to exploit spatial relationships but enables direct application of standard ensemble methods.

3.1.2 CNN Preprocessing and Augmentation

The CNN implementation maintained the original 2D image structure with dimensions (1, 28, 28) representing single-channel grayscale images. Data augmentation was applied exclusively to the training set to improve model robustness and generalization:

- **Random Horizontal Flip:** Applied with 50% probability to simulate garment presentation variations
- **Random Affine Transformations:** Rotation up to 30 degrees, translation up to 10% in each direction, and scaling between 0.9 and 1.1

These augmentations help the model learn features invariant to small geometric transformations typical in real-world fashion photography. The validation and test sets used only normalization without augmentation to ensure consistent evaluation.

3.2 Random Forest Classifier Implementation

The Random Forest classifier was implemented using scikit-learn with hyperparameters based on the optimal configuration identified by Xiao et al. (2017):

- **n_estimators:** 100 decision trees
- **max_depth:** 100 (limiting tree depth to prevent overfitting)
- **criterion:** entropy (information gain for split quality)
- **n_jobs:** -1 (parallel processing across all CPU cores)

This configuration balances model complexity with generalization capability while maintaining reasonable training time. The entropy criterion was selected based on prior work showing superior performance for Fashion-MNIST compared to Gini impurity.

3.3 Convolutional Neural Network Architecture

The CNN implementation employed the MCNN15 architecture, a deep convolutional network specifically designed for Fashion-MNIST classification. The architecture features 15 convolutional layers organized into three hierarchical groups, each followed by max pooling operations that progressively reduce spatial dimensions.

3.3.1 Architecture Specifications

Table 1: MCNN15 Architecture Details

Layer Group	**Operations**	**Output Dimensions**
Group 1	5 Conv-BN-ReLU blocks (32→64→64→32→64) 2×2 Max Pooling	32 channels, 14×14
Group 2	5 Conv-BN-ReLU blocks (64→256→192→128→64→32) 2×2 Max Pooling	32 channels, 7×7
Group 3	5 Conv-BN-ReLU blocks (32→256→256→256→128→32) 2×2 Max Pooling	32 channels, 3×3
Classifier	Flatten + Linear(288, 32) + ReLU + Linear(32, 10)	10 classes

Each convolutional layer employed 3×3 kernels with padding=1 to maintain spatial dimensions within groups, followed by batch normalization and ReLU activation. The progressive increase and subsequent decrease in channel counts enables the network to first expand feature complexity and then condense to discriminative representations.

3.4 Training Strategy

3.4.1 Random Forest Training

The Random Forest model was trained on the complete 60,000 training samples (flattened to 784-dimensional vectors). Training was performed in a single pass without iterative optimization, leveraging the ensemble nature of the algorithm. The training completed in approximately 9.45 seconds on CPU.

3.4.2 CNN Training

The CNN was trained using the following configuration:

- **Optimizer:** Adam with learning rate 1e-3

- **Weight Decay:** 1e-5 (L2 regularization)
- **Epochs:** 50
- **Batch Size:** 128
- **Loss Function:** Cross-Entropy Loss
- **Device:** GPU with CUDA support when available

Model checkpoints were saved based on best validation accuracy achieved during training. Early stopping was implicitly implemented by monitoring validation performance, with the best model selected from epoch 28 (93.48% validation accuracy) for final evaluation.

Training proceeded for 50 epochs with an average time of approximately 18 seconds per epoch, totaling approximately 900 seconds (15 minutes) for complete training. The training loss decreased from 0.69 to 0.17, while training accuracy increased from 74.5% to 93.8%.

3.5 Evaluation Protocol

Both classifiers were evaluated using identical metrics:

- **Overall Accuracy:** Proportion of correctly classified samples
- **Precision:** Weighted average of per-class precision (TP / (TP + FP))
- **Recall:** Weighted average of per-class recall (TP / (TP + FN))
- **Per-Class Metrics:** Individual precision and recall for each fashion category
- **Confusion Matrices:** Visual representation of classification errors

Training time was measured for both classifiers to compare computational efficiency. All evaluations were conducted on the held-out test set of 10,000 samples to ensure unbiased performance estimates.

4 Results

4.1 Overall Performance Comparison

The experimental results demonstrate clear performance advantages for the CNN approach across all primary metrics. Table 2 presents the comprehensive performance comparison between the two classifiers.

Table 2: Overall Performance Comparison

Metric	Random Forest	CNN	Improvement
Test Accuracy	87.52%	93.63%	+6.11%
Test Precision	87.42%	93.63%	+6.21%
Test Recall	87.52%	93.63%	+6.11%
Training Time	9.45 seconds	900 seconds	95$\times$ longer

The CNN achieved a substantial improvement of 6.11 percentage points in test accuracy, representing a 48.8% reduction in classification error rate (from 12.48% to 6.37% error). This improvement comes at the cost of significantly longer training time, with the CNN requiring approximately 95 times more computational time than the Random Forest classifier.

4.2 Per-Category Performance Analysis

Analysis of per-category performance reveals interesting patterns in classifier behavior and identifies specific fashion categories that present classification challenges for both approaches.

Table 3: Per-Category Precision and Recall on Test Set

Category	RF Precision	RF Recall	CNN Precision	CNN Recall
T-shirt/top	81.26%	86.30%	86.70%	89.20%
Trouser	99.28%	95.90%	99.20%	98.50%
Pullover	76.59%	79.50%	90.50%	89.30%
Dress	87.32%	90.90%	91.80%	93.50%
Coat	76.07%	82.00%	87.50%	91.40%
Sandal	98.05%	95.50%	98.80%	98.60%
Shirt	72.38%	57.40%	82.00%	77.50%
Sneaker	92.48%	95.90%	96.10%	98.20%
Bag	95.58%	97.30%	98.80%	98.70%
Ankle boot	95.17%	94.50%	98.30%	96.80%

Both classifiers demonstrate excellent performance on easily distinguishable categories such as trousers (99%+ precision for both methods) and bags (95%+ precision). However, significant performance gaps emerge in categories with high visual similarity, particularly upper-body garments.

4.3 Category-Specific Performance Patterns

4.3.1 Best Performing Categories

- **Trouser:** Both classifiers achieved exceptional performance (99%+ precision) due to distinctive shape characteristics and limited visual similarity with other categories. The vertical elongation and lack of similarity to other garment types make trousers highly distinguishable.
- **Bag:** High performance (95.58% RF, 98.80% CNN precision) resulting from distinct non-clothing characteristics and consistent structural features. Bags lack the limb-related features present in wearable garments, making them easier to identify.
- **Sandal:** Strong performance (98.05% RF, 98.80% CNN precision) attributed to unique open-toe design and footwear-specific features that distinguish sandals from closed shoes.

4.3.2 Challenging Categories

- **Shirt:** Consistently worst-performing category for both classifiers (72.38% Random Forest, 82.00% CNN precision). This poor performance stems from high visual similarity with T-shirts and pullovers, combined with significant intra-class variation in shirt designs (collar types, sleeve lengths, patterns).
- **Pullover:** Moderate performance degradation (76.59% Random Forest, 90.50% CNN precision) due to similarity with coats and shirts, particularly in neckline and sleeve designs.
- **Coat:** Performance challenges (76.07% Random Forest, 87.50% CNN precision) related to similarity with pullovers and jackets, combined with high variation in coat lengths and styles.
- **T-shirt/top:** Lower performance (81.26% Random Forest, 86.70% CNN precision) due to confusion with shirts, particularly when T-shirts have collar-like designs or printed patterns resembling formal shirts.

The CNN demonstrates superior performance across all challenging categories, with particularly notable improvements on Pullover (+13.91%), Coat (+11.43%), and Shirt (+9.62%). This suggests that the learned hierarchical features better distinguish subtle visual differences between similar garment types.

4.4 Training Time Analysis

Training time represents a significant differentiator between the two approaches:

Table 4: Training Time Comparison

Metric	Random Forest	CNN
Total Training Time	9.45 seconds	900 seconds
Per-Epoch Time	N/A (single-pass)	18 seconds
Training Samples/sec	6,349	56

The Random Forest classifier demonstrated exceptional computational efficiency, completing training in under 10 seconds while achieving respectable 87.52% accuracy. This efficiency makes Random

Forest highly suitable for resource-constrained environments, rapid prototyping, or applications requiring frequent model updates.

In contrast, the CNN required approximately 15 minutes for complete training across 50 epochs. However, the per-epoch training time remained consistent throughout training, indicating stable convergence properties and efficient GPU utilization. The increased computational cost is justified by the significant accuracy improvement, particularly for challenging fashion categories.

4.5 Confusion Matrix Analysis

Confusion matrices were generated for both classifiers to visualize classification patterns and identify specific error modes.

4.5.1 Random Forest Confusion Patterns

The Random Forest confusion matrix reveals several systematic misclassification patterns:

- **Shirt misclassifications:** 163 shirts misclassified as T-shirts, 121 as pullovers, 91 as coats. This pattern reflects the visual similarity between upper-body garments.
- **Pullover confusion:** 123 pullovers misclassified as coats, 57 as shirts. The similarity in sleeve length and overall silhouette contributes to this confusion.
- **Coat errors:** 90 coats misclassified as pullovers, 49 as shirts. The overlap in outerwear categories creates ambiguity.
- **Footwear confusion:** 44 ankle boots misclassified as sneakers, likely due to similar overall shape.

The top misclassification pairs account for 816 of the 1,248 total errors (65.4%), indicating concentrated confusion among visually similar categories.

4.5.2 CNN Confusion Patterns

The CNN confusion matrix shows reduced but qualitatively similar error patterns:

- **Reduced shirt errors:** CNN misclassifies fewer shirts overall, with most errors still occurring between shirt, T-shirt, and pullover categories.
- **Improved upper-body distinction:** The CNN better distinguishes coats from pullovers, likely through learned texture and structural features.
- **Minimal footwear confusion:** Ankle boot and sneaker confusion is nearly eliminated, with the CNN achieving high precision on both categories.

The CNN reduces total misclassifications from 1,248 to 637, representing a 48.9% reduction in errors.

5 Analysis and Discussion

5.1 Performance Analysis

The experimental results clearly demonstrate the superiority of CNN-based approaches for fashion image classification tasks. The 6.11 percentage point improvement in test accuracy represents a substantial reduction in classification errors, translating to approximately 611 fewer misclassified items per 10,000 products in a production environment.

The CNN's advantage is most pronounced in challenging categories with high visual similarity, particularly upper-body garments. The learned hierarchical features appear to capture subtle distinctions in garment structure, neckline design, and fabric patterns that distinguish shirts from T-shirts and pullovers. This capability is crucial for practical fashion classification systems where accurate distinction between similar categories directly impacts customer experience and inventory management.

5.2 Analysis of Worst-Performing Categories

Both classifiers performed worst on the Shirt category, followed by Pullover and Coat. This pattern reflects fundamental challenges in fashion classification:

5.2.1 Shirt Classification Challenges

The Shirt category presents the greatest difficulty due to several factors:

1. **Visual Similarity:** Shirts share structural elements with T-shirts (torso covering), pullovers (long sleeves possible), and coats (collars, buttons). These overlapping features create classification ambiguity, particularly for simple neural network architectures.
2. **High Intra-Class Variability:** The Shirt category encompasses diverse styles including dress shirts, casual shirts, flannel shirts, and polo shirts. This variability means that features learned from one shirt subtype may not generalize to others.
3. **Grayscale Limitation:** The absence of color information in Fashion-MNIST removes an important discriminative cue. In real-world scenarios, shirt colors and patterns often distinguish them from T-shirts.
4. **Resolution Constraints:** At 28×28 pixels, fine details like collar shapes, button patterns, and fabric textures are difficult to discern, particularly for classifiers operating on flattened features.

The CNN's improvement on Shirt classification (from 57.4% to 77.5% recall) demonstrates that hierarchical feature learning can partially overcome these challenges by identifying subtle texture and shape patterns invisible to the Random Forest's pixel-wise analysis.

5.2.2 Pullover and Coat Confusion

The confusion between Pullovers and Coats reflects real-world similarities in these garment types. Both are outerwear items with similar silhouettes and purposes. The CNN's superior performance (+11.43% precision on Coats, +13.91% on Pullovers) suggests that it learns to distinguish subtle differences in fabric texture, thickness patterns, and structural rigidity that characterize coats versus softer pullovers.

5.3 Generalization and Robustness

Both classifiers demonstrate reasonable generalization capabilities, though with different patterns. The Random Forest achieves 100% training accuracy while maintaining 87.52% test accuracy, indicating some degree of overfitting despite the ensemble regularization. The CNN shows more balanced performance with 93.8% training accuracy and 93.63% test accuracy (evaluated at best validation checkpoint), suggesting effective regularization through weight decay and data augmentation.

The CNN's superior generalization is particularly evident in challenging categories where the learned feature representations appear more robust to variations in presentation angle, lighting, and styling. This robustness is essential for production deployment where input images may vary significantly from training conditions.

5.4 Computational Trade-offs

The significant difference in training time (95$\times$ longer for CNN) represents an important consideration for practical deployment scenarios. However, several factors mitigate this concern:

1. **Training Frequency:** Fashion classification models typically require infrequent retraining (weekly or monthly) as new products are introduced, making longer training times acceptable for the accuracy gains achieved.
2. **Inference Efficiency:** Once trained, both models achieve comparable inference speeds. The CNN's forward pass through 15 convolutional layers is highly optimized on modern GPUs, enabling real-time classification in production environments.
3. **Accuracy-Adjusted Cost:** The substantial accuracy improvement justifies increased computational cost in most production scenarios. The reduction of 611 errors per 10,000 items can translate to significant cost savings in inventory management, customer satisfaction, and operational efficiency.
4. **Scalability:** While Random Forest training is faster, it scales poorly with dataset size (memory requirements grow with training samples). CNN training scales more favorably for very large datasets through minibatch processing.

5.5 Production Deployment Recommendation

Based on the comprehensive evaluation, we recommend the CNN approach for production fashion classification systems. This recommendation is justified by:

1. **Superior Accuracy:** The 6.11 percentage point improvement (48.8% error reduction) provides tangible business value through reduced misclassifications.
2. **Category-Specific Performance:** The CNN's advantage is most pronounced in challenging categories where accuracy is most needed. The improvement on Shirt classification (+20.1% recall) is particularly valuable given this category's importance in fashion retail.
3. **Robustness:** The CNN demonstrates better generalization and is more amenable to data augmentation, making it more robust to real-world variations.
4. **Future-Proofing:** CNN architectures are actively researched and improved, with transfer learn-

ing capabilities enabling adaptation to new categories or datasets with minimal retraining.

The Random Forest classifier remains a viable alternative for:

- Rapid prototyping and baseline establishment
- Resource-constrained environments without GPU access
- Applications where interpretability is prioritized over accuracy
- Scenarios requiring frequent model retraining with limited computational budget

6 Conclusion

This comprehensive comparison between Convolutional Neural Networks and Random Forest classifiers for Fashion-MNIST classification demonstrates clear advantages for deep learning approaches in fashion image recognition tasks. The CNN architecture achieved 93.63% test accuracy compared to 87.52% for the Random Forest classifier, representing a statistically significant improvement that translates to substantial practical benefits in production environments.

The CNN's superiority is most pronounced in challenging categories with high visual similarity, particularly upper-body garments where learned hierarchical features effectively distinguish subtle differences between shirts, T-shirts, and pullovers. While the CNN requires significantly longer training time (900 seconds vs. 9.45 seconds), this computational cost is justified by the accuracy improvement and the infrequent retraining requirements typical in fashion retail applications.

Key findings from this study include:

1. The MCNN15 architecture with batch normalization and data augmentation achieves state-of-the-art performance on Fashion-MNIST among comparable CNN designs.
2. Both classifiers struggle most with Shirt classification, but the CNN reduces error rates by 48.9% compared to Random Forest.
3. Training time differences are substantial but become less critical when considering inference speed and infrequent retraining needs.
4. The CNN demonstrates superior generalization and robustness to variations in garment presentation.

Based on these results, we recommend CNN-based approaches for production fashion classification systems where accuracy is prioritized over training efficiency. The Random Forest classifier remains a viable alternative for resource-constrained environments or applications requiring rapid model development, though with acceptance of reduced classification accuracy.

The findings contribute to the growing body of evidence supporting deep learning approaches for computer vision tasks in retail and e-commerce applications. As fashion retail continues its digital transformation, accurate automated classification systems will play increasingly important roles in inventory management, customer experience enhancement, and operational efficiency optimization.

Future research directions should explore transfer learning from larger fashion datasets, integration of color information through RGB preprocessing, and evaluation on real-world fashion imagery with varied backgrounds and lighting conditions.

References

1. Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*. https://arxiv.org/abs/1708.07747
2. TensorFlow. (2024). Basic classification: Classify images of clothing. https://www.tensorflow.org/tutorials/ke [Accessed: 10.12.2025]
3. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32. https://doi.org/10.1023/A:1010933404
4. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
5. Bhatnagar, S., Ghosal, D., & Kolekar, M. H. (2017). Image classification using multiple convolutional neural networks on the fashion-mnist dataset. In *2017 International Conference on Intelligent Sustainable Systems (ICISS)* (pp. 597-602). IEEE.
6. Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*. https://arxiv.org/abs/1412.6980
7. Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning* (pp. 448-456). PMLR.
8. Perez, L., & Wang, J. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv preprint arXiv:1712.04621*. https://arxiv.org/abs/1712.04621
9. Zalando Research. (2017). Fashion-MNIST Benchmarks. https://github.com/zalandoresearch/fashion-mnist

A Random Forest Implementation Code

Listing 1: Random Forest Training Script (src/random_forest/train.py)

```python
"""Training script for Random Forest classifier."""

import time
from src.random_forest.shared import load_dataset, make_model, save_model

def main():
    print("=" * 60)
    print("FASHION-MNIST RANDOM FOREST CLASSIFIER - TRAINING")
    print("=" * 60)

    print("\n[DATA LOADING]")
    X_train, y_train = load_dataset(train=True)
    X_test, y_test = load_dataset(train=False)
    print(f"Data loading complete: {X_train.shape[0]} train, "
          f"{X_test.shape[0]} test samples")

    print("\n[MODEL TRAINING]")
    clf, save_path = make_model(
        n_estimators=100, criterion="entropy",
        max_depth=100, n_jobs=-1
    )
    print("Random Forest initialized: n_estimators=100, "
          "max_depth=100, criterion=entropy")

    train_start = time.time()
    clf.fit(X_train, y_train)
    train_time = time.time() - train_start
    print(f"Training completed in {train_time:.2f} seconds")

    print("\n[INITIAL EVALUATION]")
    train_acc = clf.score(X_train, y_train)
    test_acc = clf.score(X_test, y_test)
    print(f"Train accuracy: {train_acc:.4f}")
    print(f"Test accuracy:  {test_acc:.4f}")

    print("\n[MODEL SAVING]")
    save_model(clf, save_path)

    print("\n" + "=" * 60)
    print("TRAINING COMPLETE")
    print("=" * 60)

if __name__ == "__main__":
    main()
```

B CNN Implementation Code

Listing 2: CNN Training Script (src/cnn/train.py)

```python
import time
import torch
import torch.nn as nn
from torch.utils.data import DataLoader
import json

from shared import device, get_loaders, make_model

torch.manual_seed(42)

def train_epoch(model, loader, criterion, optimizer, device):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for inputs, targets in loader:
        inputs, targets = inputs.to(device), targets.to(device)

        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()

        running_loss += loss.item() * inputs.size(0)
        _, preds = outputs.max(1)
        correct += (preds == targets).sum().item()
        total += targets.size(0)

    return running_loss / total, correct / total

def evaluate(model, loader, criterion, device):
    model.eval()
    loss_sum = 0.0
    correct = 0
    total = 0

    with torch.no_grad():
        for inputs, targets in loader:
            inputs, targets = inputs.to(device), targets.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, targets)
            loss_sum += loss.item() * inputs.size(0)
            _, preds = outputs.max(1)
            correct += (preds == targets).sum().item()
            total += targets.size(0)

    return loss_sum / total, correct / total

def main():
    train_loader, validation_loader, test_loader = \
        get_loaders(batch_size=128)

    model, save_path = make_model(resnet=False)
    model.to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(
        model.parameters(), lr=1e-3, weight_decay=1e-5
    )
    epochs = 50

    val_acc = 0.0
    best_val_acc = 0.0

    for epoch in range(1, epochs + 1):
        start_time = time.time()
        train_loss, train_acc = train_epoch(
            model, train_loader, criterion, optimizer, device
        )
        val_loss, val_acc = evaluate(
            model, validation_loader, criterion, device
        )
        test_loss, test_acc = evaluate(
            model, test_loader, criterion, device
        )
        end_time = time.time()

        print(json.dumps({
            "epoch": epoch,
            "train_loss": train_loss,
            "train_acc": train_acc,
            "val_loss": val_loss,
            "val_acc": val_acc,
            "test_loss": test_loss,
            "test_acc": test_acc,
            "time": end_time - start_time,
        }))

        if val_acc > best_val_acc:
            best_val_acc = val_acc
            torch.save(model.state_dict(), save_path)

if __name__ == "__main__":
    main()
```

C MCNN15 Architecture Definition

Listing 3: MCNN15 Architecture (src/cnn/shared.py)

```
import torch.nn as nn

class MCNN15(nn.Module):
    def __init__(self, num_classes=10):
        super(MCNN15, self).__init__()

        # Helper function to create Conv-BN-ReLU block
        def conv_block(in_channels, out_channels):
            return nn.Sequential(
                nn.Conv2d(in_channels, out_channels,
                          kernel_size=3, padding=1),
                nn.BatchNorm2d(out_channels),
                nn.ReLU(inplace=True),
            )

        self.features = nn.Sequential(
            # --- Group 1: 14x14 output ---
            conv_block(1, 32),
            conv_block(32, 64),
            conv_block(64, 64),
            conv_block(64, 32),
            conv_block(32, 64),
            nn.MaxPool2d(kernel_size=2, stride=2),

            # --- Group 2: 7x7 output ---
            conv_block(64, 256),
            conv_block(256, 192),
            conv_block(192, 128),
            conv_block(128, 64),
            conv_block(64, 32),
            nn.MaxPool2d(kernel_size=2, stride=2),

            # --- Group 3: 3x3 output ---
            conv_block(32, 256),
            conv_block(256, 256),
            conv_block(256, 256),
            conv_block(256, 128),
            conv_block(128, 32),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )

        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(32 * 3 * 3, 32),
            nn.ReLU(inplace=True),
            nn.Linear(32, num_classes),
        )
```

```python

    def forward(self, x):
        x = self.features(x)
        x = self.classifier(x)
        return x
```