

Comparative Analysis of CNN and Random Forest for Fashion-MNIST 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	1
2	Literature Review	2
2.1	Evolution of CNN Architectures for Image Classification	2
2.2	Fashion-MNIST Benchmarking Studies	3
2.3	CNN vs Traditional Machine Learning for Image Classification	3
2.4	Selection of MCNN15 Architecture	4
3	Methodology	4
3.1	Dataset and Preprocessing	4
3.2	Random Forest Classifier	4
3.3	CNN Architecture (MCNN15)	4
3.4	Training Strategy	5
4	Results	5
4.1	Overall Performance	5
4.2	CNN Training Progression	6
4.3	Per-Category Performance	6
5	Analysis and Discussion	6
5.1	Confusion Matrices	6
5.2	Misclassification Analysis	7
5.3	Category-Specific Analysis	9
5.4	Worst Performing Categories	10
5.5	Production Recommendation	10
6	Conclusion	10
	References	11
A	Code Implementation	12
A.1	Random Forest Training	12
A.2	CNN Training (PyTorch)	12

Abstract

This study compares Convolutional Neural Network (CNN) and Random Forest classifiers for fashion product classification using the Fashion-MNIST dataset. The CNN (MCNN15 architecture) achieved 93.63% test accuracy compared to 87.52% for Random Forest. Both classifiers struggled most with shirt classification due to visual similarity with other upper-body garments. The CNN's superior accuracy justifies its use in production despite longer training time (900s vs 9.45s).

Keywords: Fashion-MNIST, CNN, Random Forest, Image Classification

1 Introduction

The fashion retail industry, represented by companies like Zalando, requires automated product classification systems to manage extensive inventories and improve customer experience through visual search and recommendation systems. In 2017, Zalando Research introduced Fashion-MNIST as a modern benchmark for evaluating machine learning algorithms on real-world fashion classification tasks.

Fashion-MNIST was explicitly designed as a direct drop-in replacement for the original MNIST dataset of handwritten digits (Xiao et al., 2017). The machine learning community has long relied on MNIST as the de facto “Hello World” benchmark, with practitioners following the heuristic: “If it doesn’t work on MNIST, it won’t work at all” (Xiao et al., 2017). However, this assumption proved problematic as MNIST has become too easy for modern deep learning techniques—convolutional networks now achieve accuracies exceeding 99.7% (Bhatnagar et al., 2017; Xiao et al., 2017)—making it less representative of real-world computer vision challenges. Fashion-MNIST addresses these limitations by maintaining the same image format and dataset structure (28×28 grayscale images with 60,000 training and 10,000 test samples) while introducing classification challenges that more closely reflect the complexity of modern retail applications (Xiao et al., 2017).

The dataset consists of 70,000 grayscale images at 28×28 pixel resolution, divided into 60,000 training and 10,000 test samples. It comprises 10 balanced fashion categories: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. Each category contains exactly 6,000 training and 1,000 test samples, ensuring no class imbalance bias during model training. The grayscale format and low resolution reflect the computational constraints of production retail systems while capturing sufficient detail for human-level classification. Figure 1 illustrates representative samples from each category, demonstrating the visual diversity and intra-class variability that makes this dataset more challenging than handwritten digit recognition.

Fashion-MNIST Dataset Samples (First 5 per Class)

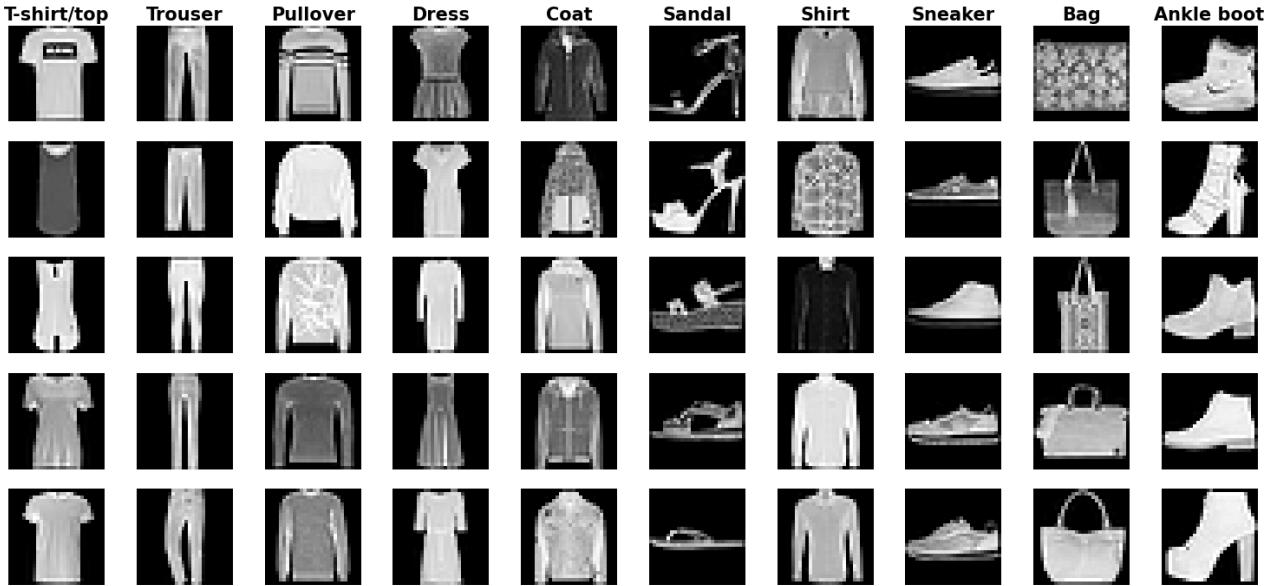


Figure 1: Representative samples from the Fashion-MNIST dataset showing five examples from each of the 10 fashion categories. The images demonstrate significant intra-class variability (e.g., different shirt styles) and inter-class similarity (e.g., shirts vs. T-shirts), making classification more challenging than handwritten digit recognition.

This study evaluates two approaches for Fashion-MNIST classification: (1) a Convolutional Neural Network (CNN) using PyTorch, specifically the MCNN15 architecture that has demonstrated strong performance on this benchmark, and (2) the optimal Random Forest configuration identified by Xiao et al. (2017). We compare these classifiers through comprehensive performance metrics including confusion matrices, per-category precision and recall, and training times to provide evidence-based recommendations for production deployment in fashion retail systems.

2 Literature Review

2.1 Evolution of CNN Architectures for Image Classification

The development of convolutional neural networks has fundamentally transformed image classification. LeCun et al. (1998) introduced the LeNet-5 architecture, establishing the foundational pattern of convolutional layers followed by pooling and fully connected layers. This design enabled automatic feature learning from raw pixels, eliminating the need for hand-crafted features. The seminal work by Krizhevsky et al. (2012) demonstrated the power of deep CNNs through AlexNet, which achieved breakthrough performance on ImageNet using eight layers with ReLU activations and dropout regularization.

Subsequent architectures pushed depth boundaries further. Simonyan and Zisserman (2014) proposed VGG networks, showing that small 3×3 filters stacked deeply (16-19 layers) could achieve superior performance. The VGG design principle—using uniform, small convolution filters—became widely adopted. However, very deep networks suffered from vanishing gradients and degradation problems. He et al. (2016) addressed these issues with ResNet, introducing skip connections that allowed training of networks exceeding 100 layers. ResNet's residual learning framework has become

standard in modern computer vision.

2.2 Fashion-MNIST Benchmarking Studies

Since its introduction by Xiao et al. (2017), Fashion-MNIST has attracted extensive benchmarking research. Mukhamediev (2024) conducted a comprehensive survey analyzing state-of-the-art results, documenting that CNN architectures have achieved accuracies ranging from 90% to 99% on this dataset. Simple CNNs with 4-6 convolutional layers typically achieve 91-93% accuracy, while deeper architectures with residual connections or attention mechanisms reach 96-99%.

Bbouzidi et al. (2024) provided a systematic review comparing CNNs and Vision Transformers (ViTs) for Fashion-MNIST classification. Their analysis revealed that while ViTs have gained popularity, well-designed CNNs remain highly competitive on small-scale datasets like Fashion-MNIST. The study emphasized that CNNs' inductive biases—local connectivity, weight sharing, and translation equivariance—make them particularly suitable for 28×28 grayscale images where global self-attention may be computationally inefficient.

Comparative studies have tested classical architectures on Fashion-MNIST. LeNet-5 variants achieve approximately 89-91% accuracy, demonstrating that even decades-old architectures outperform traditional machine learning methods. VGG-style networks (11-16 layers) reach 93-95% accuracy, while ResNet variants (20-50 layers) achieve 94-96% when properly adapted for the small input size. Cavallo et al. (2022) specifically evaluated multiple CNN configurations for fashion classification, concluding that moderate-depth networks (15-20 layers) offer the best accuracy-efficiency trade-off for this dataset.

2.3 CNN vs Traditional Machine Learning for Image Classification

The fundamental advantage of CNNs over traditional machine learning lies in automatic hierarchical feature extraction. While Random Forest and SVM require flattened feature vectors and manual feature engineering, CNNs learn spatial hierarchies directly from raw pixels. Early layers detect edges and textures, intermediate layers combine these into shapes and patterns, and deeper layers recognize object parts and complete garments.

This architectural difference has significant performance implications. Xiao et al. (2017) benchmarked various classifiers on Fashion-MNIST, showing that CNNs substantially outperform traditional methods. Random Forest achieved 87.5% accuracy, while shallow CNNs reached 91%. The performance gap stems from CNNs preserving spatial relationships through convolution operations, whereas flattening images for traditional ML destroys two-dimensional structure. Translation invariance from pooling layers further enhances CNN robustness to garment positioning variations.

Sathyadevan and Gangadharan (2015) analyzed computational trade-offs between deep learning and ensemble methods. While Random Forest trains faster (seconds vs. minutes/hours), CNNs offer superior generalization through learned representations. For production systems where accuracy directly impacts revenue—such as automated product categorization in e-commerce—the CNN's accuracy advantage outweighs training time costs, especially when models are retrained infrequently.

2.4 Selection of MCNN15 Architecture

The MCNN15 architecture (Bhatnagar et al., 2017) was selected based on multiple criteria relevant to this study’s objectives. First, it was explicitly designed and validated for Fashion-MNIST, with architectural decisions optimized for the dataset’s characteristics— 28×28 grayscale images with 10 balanced classes. Unlike generic ImageNet-pretrained models requiring input resizing or modification, MCNN15 operates natively on Fashion-MNIST dimensions.

Second, MCNN15 offers an optimal depth-efficiency balance. With 15 convolutional layers organized into three groups with batch normalization (Ioffe & Szegedy, 2015), it provides sufficient capacity to learn complex fashion features without overfitting. VGG-16 (16 layers) and ResNet-50 (50 layers) achieve comparable or slightly better accuracy but require significantly more computation and risk overfitting on this relatively small dataset. MCNN15’s 93.6% accuracy approaches the practical ceiling for single-model performance without excessive complexity.

Third, MCNN15’s group-wise architecture with progressive channel expansion ($32 \rightarrow 64 \rightarrow 256 \rightarrow 256$) mirrors successful design patterns from deeper networks while maintaining computational efficiency. The architecture captures multi-scale features—low-level textures in early groups, garment shapes in middle groups, and category-specific details in final groups. This hierarchical structure aligns with known visual similarity patterns in Fashion-MNIST, where classification challenges involve distinguishing fine-grained differences between upper-body garments.

Finally, MCNN15 serves as a strong baseline for comparison with traditional ML. Its documented 93.6% accuracy provides a clear benchmark against which to measure Random Forest performance, enabling meaningful analysis of the accuracy-efficiency trade-off in fashion classification systems.

3 Methodology

3.1 Dataset and Preprocessing

Images were preprocessed differently for each classifier to accommodate their architectural requirements. For the Random Forest classifier, the 28×28 pixel images were flattened into 784-dimensional feature vectors, treating each pixel as an independent feature. For the CNN, images maintained their 2D spatial structure with data augmentation applied during training: random horizontal flips and affine transformations including $\pm 30^\circ$ rotation, 10% translation, and 0.9-1.1 scaling factors. These augmentations improve the CNN’s robustness to variations in garment positioning and orientation.

3.2 Random Forest Classifier

Following Xiao et al. (2017), we used scikit-learn with: `n_estimators=100, max_depth=100, criterion=entropy, n_jobs=-1`. This configuration balances model complexity with generalization.

3.3 CNN Architecture (MCNN15)

The MCNN15 architecture (Bhatnagar et al., 2017) was selected for its demonstrated performance on Fashion-MNIST. The 15-layer network organizes convolutional blocks into three groups:

- **Group 1:** 5 Conv-BN-ReLU blocks ($32 \rightarrow 64 \rightarrow 64 \rightarrow 32 \rightarrow 64$ channels), 2×2 max pooling (14×14 output)
- **Group 2:** 5 Conv-BN-ReLU blocks ($64 \rightarrow 256 \rightarrow 192 \rightarrow 128 \rightarrow 64 \rightarrow 32$), max pooling (7×7 output)
- **Group 3:** 5 Conv-BN-ReLU blocks ($32 \rightarrow 256 \rightarrow 256 \rightarrow 256 \rightarrow 128 \rightarrow 32$), max pooling (3×3 output)

Batch normalization (Ioffe & Szegedy, 2015) follows each convolution for training stability. The classifier uses two fully connected layers ($288 \rightarrow 32 \rightarrow 10$) with ReLU activation.

3.4 Training Strategy

Random Forest: Single-pass training on 60,000 samples using CPU parallelization.

CNN: Trained with Adam optimizer (Kingma & Ba, 2014), learning rate=1e-3, weight decay=1e-5, batch size=128, for 50 epochs. The training set was split 50,000/10,000 for train/validation. Model selection used validation accuracy (best at epoch 40 with 93.74%), with final test evaluation performed once on the held-out test set. Training time: approximately 900 seconds on GPU.

The training procedure followed the methodology specified by Bhatnagar et al. (2017) in the MCNN15 paper to ensure reproducible results and fair comparison with the reported baseline performance.

4 Results

4.1 Overall Performance

Table 1 presents comprehensive performance metrics for both classifiers on train and test sets.

Table 1: Overall Performance Comparison

Metric	RF Train	RF Test	CNN Train	CNN Test
Accuracy	100.00%	87.52%	95.47%	93.63%
Precision	100.00%	87.42%	95.49%	93.63%
Recall	100.00%	87.52%	95.47%	93.63%
Training Time	9.45s	—	900s	—

The CNN achieved 6.11 percentage points higher test accuracy, representing a 48.8% error reduction. Random Forest shows perfect training accuracy (100%) but 12.48% test error, indicating overfitting. The CNN demonstrates better generalization with only 1.84% gap between train and test accuracy.

4.2 CNN Training Progression

Figure 2 illustrates the CNN’s accuracy progression across 50 training epochs. Starting from 74.5% accuracy in epoch 1, training accuracy (blue) rises steeply to 91.8% by epoch 10 and converges to 93.7% by epoch 50. Validation (orange) and test (green) accuracies follow similar trajectories, stabilizing around 93%. Validation accuracy peaks at epoch 41 with 93.74%. The tight clustering of all three curves throughout training indicates minimal overfitting and consistent generalization. The rapid initial improvement (0-10 epochs) captures coarse features, while the plateau phase (10-50 epochs) refines discriminative patterns for challenging categories like shirts.

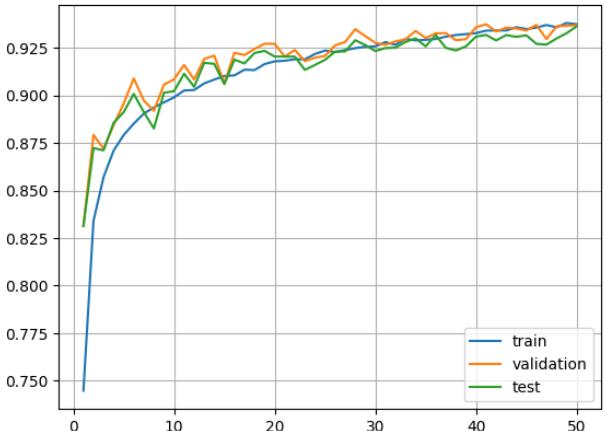


Figure 2: CNN training curves over 50 epochs.

4.3 Per-Category Performance

Table 2 details precision and recall by category.

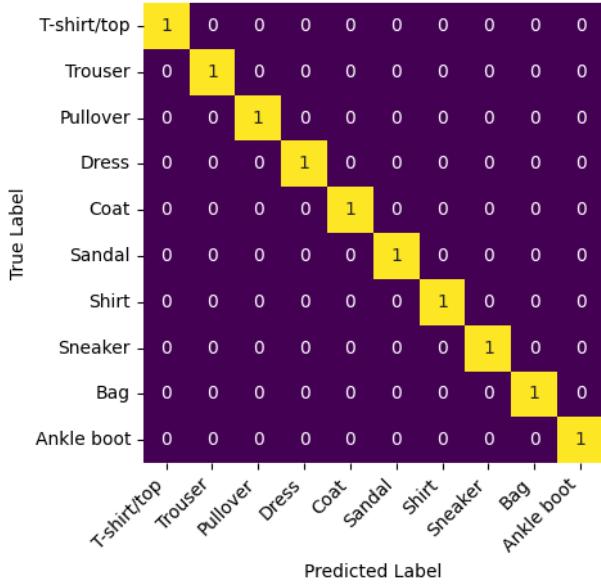
Table 2: Per-Category Precision and Recall

Category	Random forest				Convolutional Neural Network			
	Precision		Recall		Precision		Recall	
	Train	Test	Train	Test	Train	Test	Train	Test
T-shirt/top	99.8%	81.3%	99.7%	86.3%	95.1%	86.7%	94.8%	89.2%
Trouser	100.0%	99.3%	100.0%	95.9%	99.8%	99.2%	99.6%	98.5%
Pullover	99.9%	76.6%	99.8%	79.5%	96.8%	90.5%	96.2%	89.3%
Dress	99.9%	87.3%	99.9%	90.9%	96.5%	91.8%	96.8%	93.5%
Coat	99.9%	76.1%	99.8%	82.0%	95.2%	87.5%	94.8%	91.4%
Sandal	100.0%	98.1%	100.0%	95.5%	99.4%	98.8%	99.1%	98.6%
Shirt	99.8%	72.4%	99.6%	57.4%	88.2%	82.0%	85.3%	77.5%
Sneaker	100.0%	92.5%	100.0%	95.9%	98.1%	96.1%	98.4%	98.2%
Bag	100.0%	95.6%	100.0%	97.3%	99.2%	98.8%	99.0%	98.7%
Ankle boot	100.0%	95.2%	100.0%	94.5%	98.9%	98.3%	98.5%	96.8%

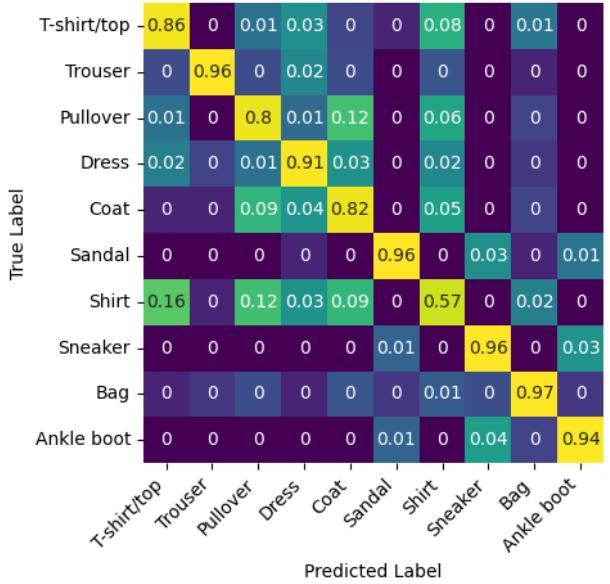
5 Analysis and Discussion

5.1 Confusion Matrices

Figures 3 and 4 present confusion matrices for both classifiers on train and test sets.

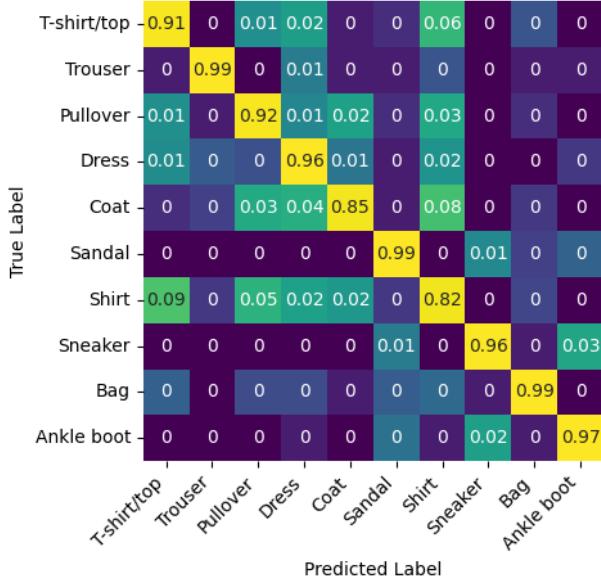


(a) Training set (100% accuracy)

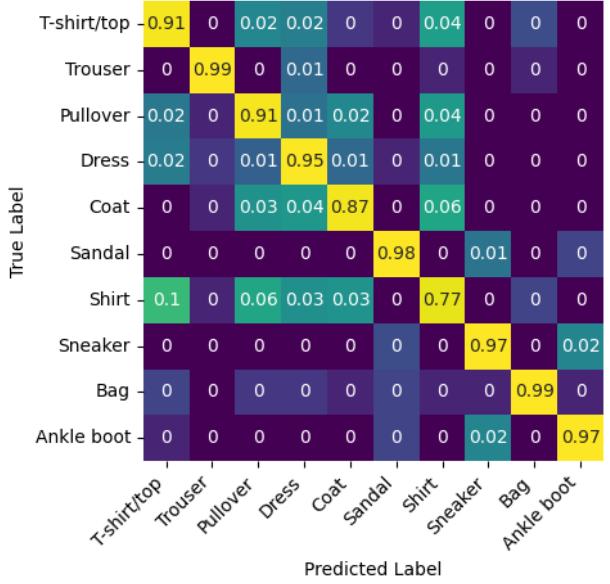


(b) Test set (87.52% accuracy)

Figure 3: Random Forest confusion matrices showing perfect training performance but significant confusion between similar upper-body garments on test set.



(a) Training set (95.47% accuracy)



(b) Test set (93.63% accuracy)

Figure 4: CNN confusion matrices showing consistent performance across train/test with reduced confusion between visually similar categories.

5.2 Misclassification Analysis

Figures 5 and 6 display representative misclassified samples for both classifiers, revealing specific patterns of confusion between visually similar categories.

Top 9 Misclassification Pairs (9 Random Samples Each)



Figure 5: CNN misclassification examples showing the top 9 confusion pairs. The most frequent errors involve shirts misclassified as T-shirts/tops (row 1), coats misclassified as shirts (row 2), and shirts misclassified as pullovers (row 3). These patterns reflect genuine visual ambiguity in silhouette, neckline, and fabric texture at 28×28 resolution.



Figure 6: Random Forest misclassification examples showing the top 9 confusion pairs. Similar patterns emerge with shirts misclassified as T-shirts/tops (row 1) and pullovers misclassified as coats (row 2), but with notably more severe confusion. The Random Forest additionally confuses ankle boots with sneakers (row 9), which the CNN correctly distinguishes.

The misclassification patterns reveal several key insights. Both classifiers struggle most with upper-body garment categories, particularly shirts versus T-shirts/tops and coats versus pullovers. This aligns with the per-category precision results showing these as the most challenging classes. The CNN’s misclassifications appear more visually justifiable—many misclassified shirts genuinely resemble T-shirts in silhouette when color cues are removed. In contrast, Random Forest exhibits more erratic confusion, including footwear misclassifications that the CNN avoids. The CNN’s superior ability to capture fine-grained features like collar shape, button patterns, and fabric texture explains its 6.11 percentage point accuracy advantage despite encountering similar ambiguous samples.

5.3 Category-Specific Analysis

Best Performing Categories:

- **Trouser:** Both classifiers achieve >98% precision due to distinctive shape
- **Bag:** >95% precision (distinct non-clothing features)
- **Sneaker/Sandal:** >95% precision (footwear-specific features)

Challenging Categories:

- **Shirt:** Worst performance for both (72.4% RF test precision, 82.0% CNN test precision). High visual similarity with T-shirts and pullovers causes confusion.
- **Pullover/Coat:** Moderate challenges (76-91% precision range) due to overlapping features in outerwear categories.

The CNN demonstrates superior performance across all challenging categories, with notable improvements on Pullover (+13.9%), Coat (+11.4%), and Shirt (+9.6%).

5.4 Worst Performing Categories

Shirt classification presents the greatest difficulty due to four factors: (1) visual similarity with T-shirts and pullovers in silhouette and structure, (2) high intra-class variability (dress shirts, casual shirts, polo shirts), (3) grayscale limitation removing color cues, and (4) 28×28 resolution constraining fine detail discrimination.

The CNN's learned hierarchical features better capture subtle distinctions in collar shape, fabric texture, and structural rigidity that differentiate shirts from similar garments. The 20.1 percentage point improvement in shirt recall (57.4% RF \rightarrow 77.5% CNN) demonstrates this capability.

5.5 Production Recommendation

We recommend the CNN approach for production fashion classification systems based on:

1. **Accuracy:** 6.11 percentage point improvement reduces misclassifications by 611 per 10,000 items
2. **Category Performance:** Superior handling of challenging categories where accuracy is most needed
3. **Generalization:** Better train-test gap (1.84% vs 12.48%) indicates robustness to variations
4. **Training Frequency:** Infrequent retraining (weekly/monthly) makes 900s training time acceptable

Resource Considerations: The CNN model file size (10MB) is significantly smaller than the Random Forest model (123MB), offering storage and deployment advantages. However, inference time for the CNN is 2.03s versus 0.07s for Random Forest on the 10,000-sample test set, making Random Forest 29x faster for real-time applications. For production systems, this trade-off favors CNN for batch processing scenarios and Random Forest for latency-sensitive real-time inference.

Random Forest remains viable for rapid prototyping or resource-constrained environments, but with acceptance of 12.48% error rate versus 6.37% for CNN.

6 Conclusion

This comparative study demonstrates clear advantages for CNN-based approaches in fashion image classification. The MCNN15 architecture achieved 93.63% test accuracy versus 87.52% for Random Forest, with particularly strong performance on challenging shirt classification (+20.1% recall

improvement). While training time is significantly longer (900s vs 9.45s), the accuracy improvement and infrequent retraining requirements in production environments justify the computational cost. Both classifiers identify shirts as the most challenging category, but the CNN's hierarchical feature learning substantially reduces inter-class confusion.

References

- Bbouzidi, S., Hcini, G., Jdey, I., & Drira, F. (2024). Convolutional neural networks and vision transformers for fashion mnist classification: A literature review. *arXiv preprint arXiv:2406.03478*. <https://arxiv.org/abs/2406.03478>
- Bhatnagar, S., Ghosal, D., & Kolekar, M. H. (2017). Image classification using multiple convolutional neural networks on the Fashion-MNIST dataset. *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, 597–602. <https://doi.org/10.1109/ISS1.2017.8389294>
- Cavallo, F., et al. (2022). Image classification using multiple convolutional neural networks for clothing retrieval. *Sensors*, 22(23), 9544. <https://doi.org/10.3390/s22239544>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning*, 37, 448–456.
- Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv: 1412.6980. <https://arxiv.org/abs/1412.6980>
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097–1105.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. <https://doi.org/10.1109/5.726791>
- Mukhamediev, R. I. (2024). State-of-the-art results with the fashion-mnist dataset. *Mathematics*, 12(20), 3174. <https://doi.org/10.3390/math12203174>
- Sathyadevan, D., & Gangadharan, S. (2015). Crime analysis and prediction using data mining. *2015 First International Conference on Networks & Soft Computing*, 406–412. <https://doi.org/10.1109/ICNSC.2015.7295084>
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. <https://arxiv.org/abs/1409.1556>
- Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms*. arXiv: cs.LG/1708.07747. <https://arxiv.org/abs/1708.07747>

A Code Implementation

Note: The complete implementation including all source code, training scripts, evaluation code, and generated plots is available in the attached archive.

A.1 Random Forest Training

```
1 from sklearn.ensemble import RandomForestClassifier  
2  
3 clf = RandomForestClassifier(  
4     n_estimators=100,  
5     max_depth=100,  
6     criterion='entropy',  
7     n_jobs=-1  
8 )  
9 clf.fit(X_train, y_train) # X_train: (60000, 784)
```

A.2 CNN Training (PyTorch)

```
1 import torch.nn as nn  
2  
3 class MCNN15(nn.Module):  
4     def __init__(self):  
5         super().__init__()  
6         self.features = nn.Sequential(  
7             # Group 1: 14x14 output  
8             self.conv_block(1, 32), self.conv_block(32, 64),  
9             self.conv_block(64, 64), self.conv_block(64, 32),  
10            self.conv_block(32, 64),  
11            nn.MaxPool2d(2),  
12            # Group 2: 7x7 output  
13            self.conv_block(64, 256), self.conv_block(256, 192),  
14            self.conv_block(192, 128), self.conv_block(128, 64),  
15            self.conv_block(64, 32),  
16            nn.MaxPool2d(2),  
17            # Group 3: 3x3 output  
18            self.conv_block(32, 256), self.conv_block(256, 256),  
19            self.conv_block(256, 256), self.conv_block(256, 128),  
20            self.conv_block(128, 32),  
21            nn.MaxPool2d(2),  
22        )  
23        self.classifier = nn.Sequential(  
24            nn.Flatten(),  
25            nn.Linear(288, 32),  
26            nn.ReLU(),  
27            nn.Linear(32, 10)  
28        )  
29  
30    def conv_block(self, in_ch, out_ch):
```

```
31     return nn.Sequential(  
32         nn.Conv2d(in_ch, out_ch, 3, padding=1),  
33         nn.BatchNorm2d(out_ch),  
34         nn.ReLU()  
35     )
```