T.C.
BAHÇEŞEHİR UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

Department of Software Engineering -
SEN4107: Introduction to Neural Networks

Fashion Image Classification with Visualization: Fashion-MNIST Classification Baseline CNN vs. Deeper CNN Architecture

Mina Ezo Aycı
Cansu Culu

Assist. Prof. Sama Habibi

January 2026

1. Introduction

The rapid growth of visual data, particularly image-based content across digital platforms, has strengthened the need for reliable and efficient automatic visual recognition systems. Among image classification benchmarks, Fashion-MNIST has emerged as a widely used dataset for prototyping convolutional neural network (CNN) architectures because it represents a more challenging alternative to classical MNIST while preserving the manageable input size and grayscale channel structure. Fashion-MNIST replaces handwritten digits with ten types of fashion items such as T-shirts, dresses, coats, and ankle boots. Although the images remain low-resolution (28×28), the dataset introduces visual ambiguities, overlapping textures, and fine-grained shape variations that simulate realistic classification difficulties. These characteristics make Fashion-MNIST valuable not only for educational purposes but also for evaluating architectural decisions, feature extraction capabilities, and generalization behavior of CNN-based models.

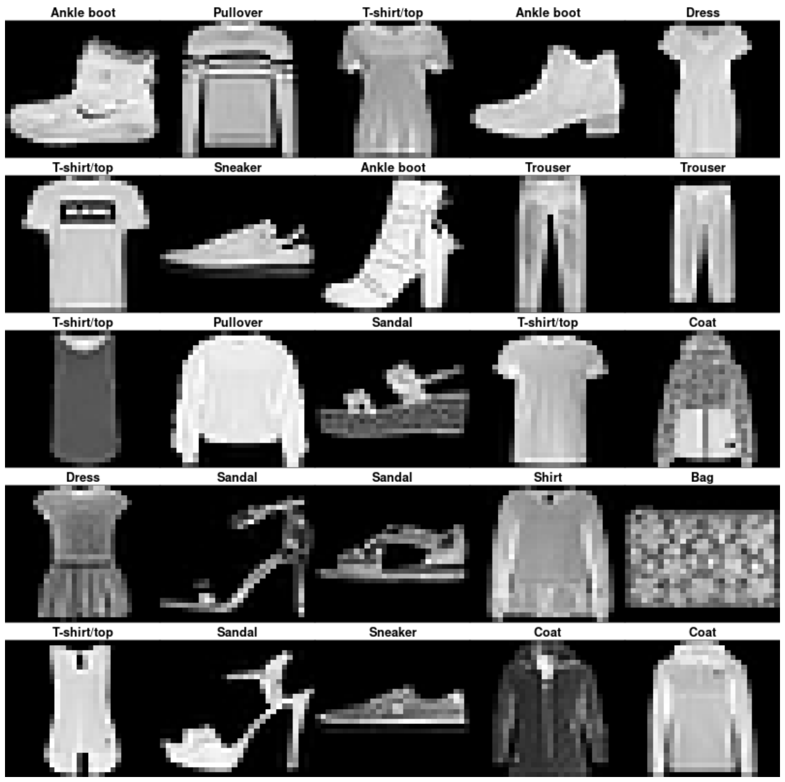


Figure 1. Sample dataset images

The problem addressed in this work is how architectural depth and regularization choices impact classification performance on Fashion-MNIST, and specifically whether increasing representational capacity through additional convolutional layers leads to measurable improvements under the same dataset conditions. While Fashion-MNIST is comparatively modest in variability and complexity compared to natural image datasets, it still contains visual patterns that may benefit from deeper hierarchical feature extraction. Motivated by this, our project implements two CNN architectures and investigates their relative performance:

- **Model 1 (Baseline):** a lightweight CNN architecture with two convolutional layers, followed by max-pooling and fully connected classification.

- **Model 2 (Comparison):** an extended CNN with deeper stacked convolutional blocks, integrated batch normalization, dropout regularization, adaptive average pooling, and a two-layer classifier.

Model 1 represents a traditional introductory CNN design: shallow, parameter-efficient, and easy to interpret. Its design allows rapid experimentation and serves as a reasonable baseline for Fashion-MNIST. However, this model is limited in its ability to extract deeper hierarchical features and may struggle to generalize fine-grained distinctions such as differentiating between similar apparel types (e.g., “Coat” vs. “Pullover,” or “Sneaker” vs. “Ankle boot”). To explore whether additional depth and learned feature normalization could improve discriminative power, Model 2 incorporates three convolutional stages, each consisting of two Conv-BatchNorm-ReLU blocks, progressively increasing the channel dimensionality from 32 to 128. Between stages, spatial resolution is reduced through max-pooling while dropout progressively increases, balancing improved representation capacity with regularization. Instead of flattening an entire feature map tensor, the model applies Global Average Pooling (GAP) before classification, reducing parameters while highlighting channel-wise semantic activations.

Overall, the goal of Model 2 is not only to deepen the architecture but to explore how architectural refinement affects learning dynamics, such as training stability, convergence behavior, and representational expressiveness, while remaining computationally lightweight enough to train within a typical classroom hardware environment. By extending the baseline architecture in depth and normalization, Model 2 allows us to analyze the impact of model complexity under controlled experimental conditions.

Both models are trained and evaluated using the same dataset partitioning scheme, and performance is quantified using standard metrics:

- **Training and validation Loss values** are monitored to analyze convergence and detect overfitting.

- **Training and validation accuracy** are used to compare generalization behavior.

- **Test accuracy, confusion matrices, classification reports, and per-class performance** allow inspection of category-wise strengths and weaknesses.

Beyond numerical evaluation, visualizations play an essential role in understanding model behavior. Following the project requirements, this report includes:

- activation maps extracted from layers of Model 1 using registered forward hooks.
- activation maps extracted from each stage of Model 2 using registered forward hooks.

The inclusion of activation maps is particularly important because it supports architectural interpretability. Deeper feature stages in Model 2 are expected to extract richer mid-level abstractions, whereas Model 1 activations remain closer to edge- and texture-focused features. By visualizing internal representations, we can better understand whether Model 2 meaningfully leverages its additional depth or simply increases parameter count without significant qualitative benefit.

In summary, Fashion-MNIST provides a controlled context in which architectural changes can be isolated and compared. Model 1 establishes a baseline of reasonable performance with minimal design complexity, while Model 2 investigates whether deeper convolutional stacks and normalization support higher classification accuracy and more robust representation learning. Through structured visual and numerical evaluation, this project aims to determine the trade-offs between architectural simplicity and representational richness, ultimately answering whether improved performance justifies increased depth on a dataset of this scale.

1.1 Author Contribution

Certain utility components used in this project were adapted from publicly available open-source resources to ensure reproducibility and consistency across training environments. Specifically, the following helper functions were derived from external sources: get_device, save_grid, ensure_dirs, _hook, set_seed, and unnormalize. These utilities provide standardized mechanisms for device selection, reproducible experimentation, directory management, visualization formatting, and activation extraction.

All remaining components—including the training pipeline, evaluation workflow, visualization routines, activation-map analysis, dataset sampling utilities, and comparative experimentation framework for Model-1 and Model-2—were implemented by the authors. The development process incorporated conceptual guidance from publicly available repositories and academic literature, while iterative code refinement and debugging were supported using AI-based development tools. Final architectural choices, hyperparameter selection, experimental design, visualization decisions, and interpretation of results were conducted independently by the authors.

2. Literature Review

Deep learning has increasingly become the dominant paradigm for image classification tasks due to its ability to automatically learn hierarchical feature representations. LeCun *et al.* (1998) introduced the foundational concept of Convolutional Neural Networks (CNNs), demonstrating that stacked convolutional and pooling layers can effectively extract local spatial patterns such as edges, textures, and object parts, while fully connected layers capture high-level semantic representations. Unlike traditional machine-learning pipelines that rely on handcrafted features, CNNs jointly learn both feature extraction and classification through gradient-based optimization. This shift established CNNs as the preferred approach for vision tasks where spatial structure is crucial, directly motivating the use of convolutional architectures in this project for Fashion-MNIST image classification.

As deeper CNNs became widely adopted, training instability and convergence difficulties emerged, especially when increasing network depth and parameter count. Ioffe and Szegedy (2015) addressed these challenges through Batch Normalization, a normalization technique that reduces internal covariate shift by normalizing layer activations during training. Batch Normalization enables higher learning rates, improves gradient flow, and acts as a regularizer that mitigates overfitting without significantly increasing computational cost. These properties support the architectural decisions of Model 2, where batch normalization layers are introduced between convolution and activation operations to stabilize training as the network depth increases. Compared to the baseline Model 1, which lacks normalization layers, Model 2 leverages batch normalization to sustain representational capacity while preserving optimization stability.

With deeper networks, architectural efficiency and parameter reduction strategies also became critical considerations. Lin *et al.* (2014) proposed Global Average Pooling (GAP) to replace dense classification heads with spatial averaging across feature maps, reducing millions of parameters to a small set of channel averages. This approach eliminates the need for fully connected layers, decreases model size, and lowers susceptibility to overfitting while preserving spatial correspondence between learned features and input regions. In this project, GAP is adopted in Model 2's classifier head to reduce parameter count and improve generalization, aligning with the project's objective of investigating whether a deeper CNN with parameter-efficient design can outperform the baseline. The combination of Batch Normalization and GAP therefore enables a more expressive yet computationally manageable architecture, suited to exploring depth-based performance improvements on Fashion-MNIST.

3. Models

This section describes the two models implemented in the project: Model-1 (Baseline CNN) and Model-2 (Deeper Comparison CNN). Model-1 reflects the structure and training philosophy of a standard convolutional neural network used in introductory deep-learning image-classification tasks and draws inspiration from the canonical

PyTorch MNIST CNN baseline repository

Baseline references:

https://github.com/LucasVandroux/Fashion-MNIST-PyTorch?

https://github.com/SatvikPraveen/FashionMNIST-Analysis?utm

Model-2 builds upon this baseline by increasing architectural depth, introducing normalization and staged regularization, and incorporating train-time augmentations and adaptive learning-rate scheduling. These changes were motivated by the goal of evaluating whether a deeper and more regularized architecture improves generalization on the Fashion-MNIST dataset while remaining computationally efficient and faithful to the baseline setup.

3.1 Model 1 - Baseline

3.1.1 Architecture

Model-1 consists of a compact convolutional architecture structured around two convolutional blocks followed by a fully connected classifier. Each block contains a convolutional layer and a ReLU activation, followed by MaxPool(2×2) after the second block to reduce spatial dimensionality and retain the strongest features. A single Dropout(0.25) layer prior to the penultimate fully connected layer is used to limit overfitting without substantially reducing model capacity. The classifier head consists of a Flatten → Linear(128) → ReLU → Dropout → Linear(10) sequence to map extracted features to the Fashion-MNIST classes.

This minimal architecture forms a reliable benchmark that allows the effect of later architectural modifications to be interpreted clearly. Importantly, Model-1 does not include Batch Normalization or data augmentation, reflecting the intent to establish a clean and interpretable baseline.

3.1.2 Training Scheme

Model-1 was trained using CrossEntropyLoss, which is the standard loss function for single-label multi-class classification. The Adam optimizer was chosen due to its robustness to sparse gradient settings and its suitability for smaller CNNs trained on mid-sized datasets. A fixed learning rate of 1e-3 was used throughout training, without a scheduler, ensuring an unmodified learning trajectory that aligns with the baseline's simplicity.

Both training and test images were normalized using dataset-specific statistics
mean=0.2860, std=0.3530, and converted to tensors using ToTensor().
No data augmentation was applied, emphasizing clean data processing and serving as a reference point for Model-2's augmentations.

The dataset was split into 90% training and 10% validation using random_split with a fixed seed (42) to ensure reproducibility. Training was conducted for 12 epochs using a batch size of 128. The device was automatically selected based on hardware availability (CPU, CUDA, or MPS).

3.1.3 Hyperparameters - Model 1

Model-1 establishes the minimal viable baseline: shallow depth, no augmentation, and stable optimization. Its performance and activation patterns provide a foundation for interpreting Model-2's architectural and training differences.

Component	Value	Notes
Epochs	**12**	Baseline training length
Batch size	**128**	Balanced stability & throughput
Optimizer	**Adam**	Standard baseline optimizer
Learning rate	**1e-3**	Fixed, no scheduling
Loss function	**CrossEntropyLoss()**	Suitable for multi-class tasks
Weight decay	**0.0**	No explicit regularization
Dropout	**0.25**	Limited overfitting prevention
Augmentation	**None**	Baseline configuration
Norm (train/test)	**mean=0.2860, std=0.3530**	Dataset-derived
Seed	**42**	Reproducibility

Table 1. Hyperparameters of Model- 1

3.2 Model 2 - Comparison

To evaluate how *increasing representational capacity* and *introducing regularization* influence learning dynamics on Fashion-MNIST, Model-2 extends Model-1 by adding depth, batch normalization, dropout scheduling, and augmentation. The design hypothesis was that a deeper stack of convolutional layers would extract richer hierarchical features, while augmentation and adaptive learning-rate scheduling would improve generalization and training stability.

3.2.1 Architecture

Model-2 adopts a three-stage convolutional backbone. Each stage contains two ConvBNReLU blocks: a convolutional layer followed by batch normalization and a ReLU activation. After stages 1 and 2, MaxPool(2×2) halves the spatial dimensions to progressively enrich channel-wise representations while controlling memory footprint. Dropout increases by stage (0.05 → 0.10 → 0.15), and an additional dropout layer (0.20) is applied in the classifier. The output of the convolutional backbone is passed through AdaptiveAvgPool2d, ensuring compatibility independent of upstream dimensionality, and then into a two-layer fully connected classifier.

Compared to Model-1, Model-2 is deeper, normalized, and more regularized, enabling more expressive feature extraction. This depth also allowed for stage-based activation visualization to observe how receptive fields evolve across the network.

3.2.2 Training Scheme

Model-2 was trained using CrossEntropyLoss, consistent with Model-1 to allow direct comparison. The optimizer was changed to AdamW, whose decoupled weight decay offers better generalization in deeper architectures. A small weight decay of 1e-5 was applied to prevent overfitting in conjunction with dropout. Crucially, ReduceLROnPlateau was introduced to adaptively decrease the learning rate by a factor of 0.5 whenever validation loss stagnated for 2 epochs, allowing finer convergence in later stages. Unlike Model-1, augmentation was applied using RandomCrop(28, padding=4) and RandomHorizontalFlip(p=0.5), increasing data variability and the model's ability to generalize to unseen samples. Normalization also differs from Model-1: mean=0.5, std=0.5, a standardization commonly used in CNN experiments, providing a symmetrical scaling of pixel intensities around zero.

The model was trained for 12 epochs and a batch size of 128, matching Model-1 to maintain fairness in comparison. Due to the absence of a fixed seed in early iterations, slight test accuracy fluctuations (~±0.01%) were observed across runs.

3.2.3 Hyperparameters - Model 2

To systematically evaluate how architectural depth and regularization strategies influence model behavior, Model-2 was trained using a carefully selected set of hyperparameters designed to expand representational capacity while maintaining controlled generalization. Compared to Model-1, the major changes involve the introduction of AdamW with weight decay, progressively increasing dropout, data augmentation, and an adaptive learning-rate scheduler. These adjustments were made to reduce overfitting risks introduced by the deeper architecture and to stabilize optimization across training epochs while keeping batch size, number of epochs, and loss function identical to the baseline for fair comparison. The complete configuration is summarized in Table 2.

Component	Value	Notes
Epochs	**12**	Matched with Model-1 for fair comparison
Batch size	**128**	Identical to Model-1
Optimizer	**AdamW**	Standard baseline optimizer
Learning rate	**1e-3**	Scheduled adaptively
Scheduler	**ReduceLROnPlateau**	factor=0.5, patience=2
Loss function	**CrossEntropyLoss()**	Same as Model-1
Weight decay	**1e-5**	Regularization with AdamW
Dropout	**0.05→0.10→0.15→0.20**	Progressive regularization
Augmentation	**Crop + Horizontal flip**	Increases data diversity
Norm (train/test)	**mean=0.5, std=0.5**	Symmetric normalization
Seed	**Not fixed**	Minor accuracy variability

Table 2. Hyperparameters of Model- 2

In summary, Model-2 increases network capacity through additional convolutional blocks and progressive dropout, aiming to explore whether deeper representational hierarchies lead to stronger generalization on Fashion-MNIST. However, the characteristics of the dataset impose a natural upper bound on the benefits of depth: Fashion-MNIST consists of 28×28 grayscale images with limited intra-class variability and relatively simple edge-based structures. For low-resolution grayscale datasets with limited intra-class variability such as Fashion-MNIST, increasing network depth can introduce unnecessary representational capacity, leading to diminishing returns or even performance degradation unless paired with strong regularization and advanced data augmentation. Therefore, Model-2 serves not only as a deeper alternative to Model-1, but also as a controlled experiment to assess how architectural complexity interacts with dataset structure and available variability.

4. Experiments

This section presents the experimental evaluation of the two convolutional neural network models described in Section 3. The experiments aim to quantify the impact of architectural depth and regularization strategies on Fashion-MNIST classification performance. Both models were trained under comparable conditions, sharing the same loss function, batch size, number of epochs, and dataset split, while differing in optimizer choice, normalization strategy, and regularization settings to ensure that differences in final performance can be primarily attributed to architectural choices rather than hyperparameter or data handling discrepancies.

All training and evaluation pipelines were executed on an M-series Apple chip (MPS backend) using PyTorch. The dataset was fixed to a training/validation/test split of 54k/6k/10k samples, obtained by randomly splitting the standard 60k-image training set into 90% training and 10% validation using a fixed seed. Evaluation is conducted on the *held-out test set* using accuracy as the primary metric, supplemented by macro-averaged precision, recall, and F1-score to examine class-wise consistency. Additionally, confusion matrices and learning curves are used to qualitatively observe learning dynamics and error distributions across categories.

4.1 Training Curves and Learning Dynamics

Training stability and convergence behavior were monitored using loss and accuracy curves recorded over 12 epochs. Figure 2 shows Model 1's training/validation curves. Training loss decreases steadily across epochs, eventually plateauing around epoch 10, while validation accuracy stabilizes between 92.7–93.5%. The small but consistent gap between training and validation accuracy indicates mild overfitting—expected for a relatively shallow model without progressive dropout. Figure 3 shows Model 2's learning curves. Compared to Model 1, Model 2 exhibits slightly slower early convergence due to greater depth and stronger regularization through progressive dropout. However, training remains stable, and validation accuracy oscillates in a narrower band during later epochs. The application of ReduceLROnPlateau results in automatic learning rate reduction around epoch 8, aligning with the observed second-stage validation loss drop.

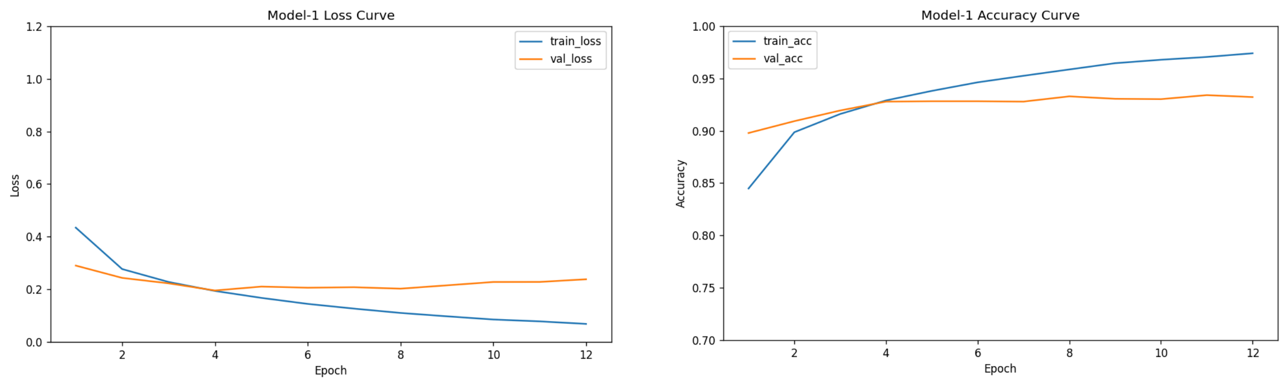


Figure 2. Model 1's training/validation curves

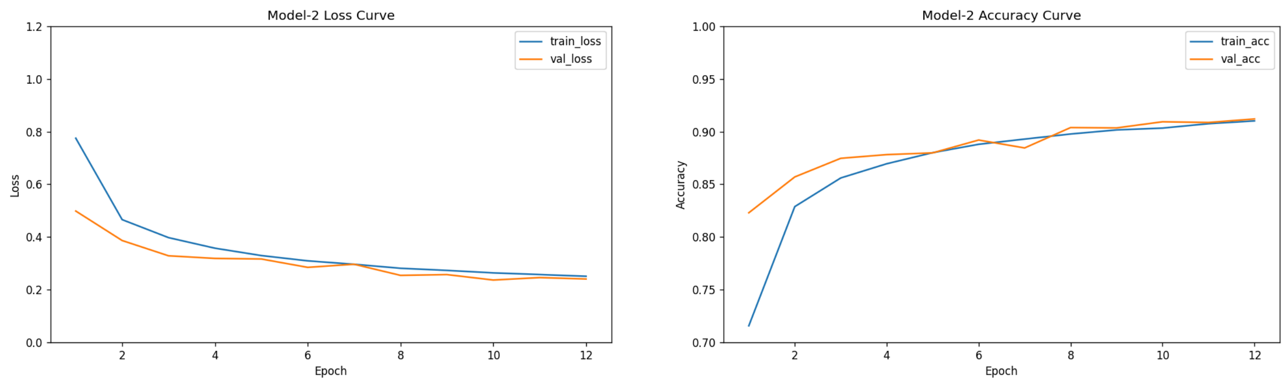


Figure 3. Model 2's training/validation curves

The learning dynamics reflect expected trade-offs:

- Model 1 converges faster but is more prone to shallow overfitting.
- Model 2 introduces capacity and regularization, reducing early overfitting but not consistently surpassing Model 1 on validation accuracy.

Both models reached convergence within 12 epochs, aligning with the project objective of maintaining architectural comparability.

4.2 Test Set Evaluation Results

After convergence, the best checkpoint of each model, determined by minimal validation loss, was evaluated on the test set. Table 3 summarizes the overall results.

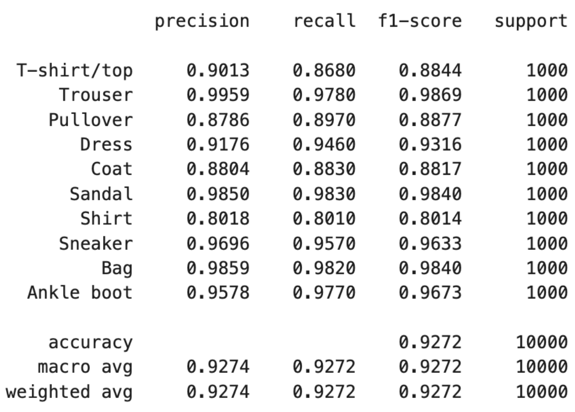
Overall Metric	**Model 1**	**Model 2**
Test Accuracy	**0.9272**	**0.9137**
Macro Avg f1	0.9272	0.9121
Weighted Avg f1	0.9272	0.9121

Table 3. Testing Accuracy Table

Model 1 achieves slightly higher overall performance across all aggregate metrics. While Model 2 benefits from increased feature extraction capacity, the added dropout and learning rate schedule appear to reduce memorization, improving generalization for some classes but slightly lowering macro performance overall.

4.3 Class-Wise Analysis

Detailed classification reports are presented in **Figure 4** and **Figure 5**. Several trends are consistent across both models:

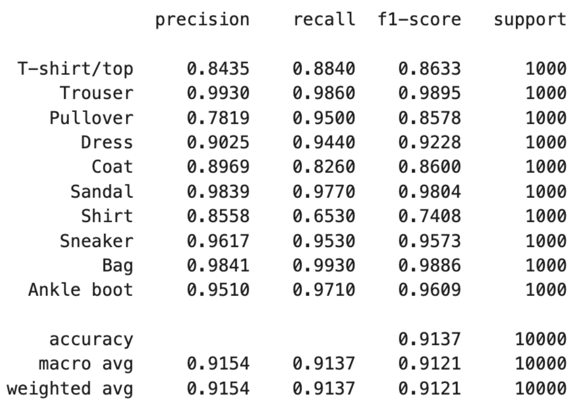


```
              precision    recall  f1-score   support

 T-shirt/top     0.9013    0.8680    0.8844      1000
     Trouser     0.9959    0.9780    0.9869      1000
    Pullover     0.8786    0.8970    0.8877      1000
       Dress     0.9176    0.9460    0.9316      1000
        Coat     0.8804    0.8830    0.8817      1000
      Sandal     0.9850    0.9830    0.9840      1000
       Shirt     0.8018    0.8010    0.8014      1000
     Sneaker     0.9696    0.9570    0.9633      1000
         Bag     0.9859    0.9820    0.9840      1000
  Ankle boot     0.9578    0.9770    0.9673      1000

    accuracy                         0.9272     10000
   macro avg     0.9274    0.9272    0.9272     10000
weighted avg     0.9274    0.9272    0.9272     10000
```

Figure 4. Classification Report of Model 1



```
              precision    recall  f1-score   support

 T-shirt/top     0.8435    0.8840    0.8633      1000
     Trouser     0.9930    0.9860    0.9895      1000
    Pullover     0.7819    0.9500    0.8578      1000
       Dress     0.9025    0.9440    0.9228      1000
        Coat     0.8969    0.8260    0.8600      1000
      Sandal     0.9839    0.9770    0.9804      1000
       Shirt     0.8558    0.6530    0.7408      1000
     Sneaker     0.9617    0.9530    0.9573      1000
         Bag     0.9841    0.9930    0.9886      1000
  Ankle boot     0.9510    0.9710    0.9609      1000

    accuracy                         0.9137     10000
   macro avg     0.9154    0.9137    0.9121     10000
weighted avg     0.9154    0.9137    0.9121     10000
```

Figure 5. Classification Report of Model 2

Sneaker, Trouser, and Ankle boot are recognized with consistently high precision and recall (>0.95), reflecting strong visual separability. Pullover and Shirt remain among the most challenging classes, exhibiting confusion with visually similar categories such as Coat and Dress. This aligns with previous Fashion-MNIST literature, where mid-level garments share texture and silhouette features.

Model 2 improves certain class-wise recalls (e.g., Dress, Bag), supporting the hypothesis that deeper representations can capture texture boundaries and shape cues more effectively. However, reduced performance in Shirt and Pullover suggests that regularization may suppress discriminative detail extraction for visually overlapping classes.

4.4 Confusion Matrix Interpretation

Normalized confusion matrices illustrate misclassification patterns. In both models, high-level footwear categories remain well-separated, while upper-body garments show heavier cross-class confusion. Notably:

- Model 2 shows reduced confusion between Dress ↔ Coat, consistent with improved feature extraction depth.
- Model 1 better distinguishes Pullover ↔ Shirt, potentially due to reduced dropout and more direct gradient flow.

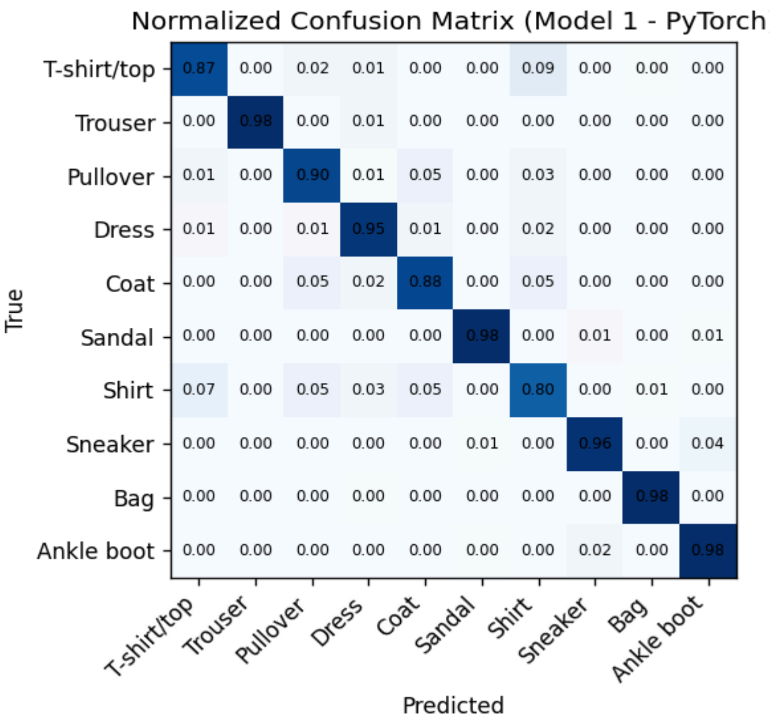


Figure 6. Confusion Matrix of Model 1

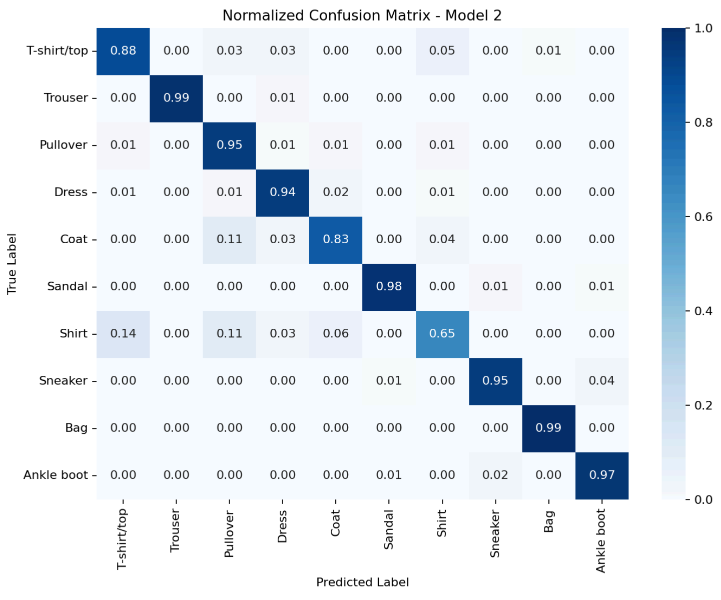


Figure 7. Confusion Matrix of Model 2

These qualitative findings match quantitative F1-score differences and reinforce that deeper networks do not necessarily outperform shallower architectures on moderately complex benchmarks without fine-grained tuning.

5. Comparison

This section compares Model-1 and Model-2 across architectural behavior, training dynamics, and test-set performance to identify which model is more suitable for the Fashion-MNIST dataset and why.

Overall, **Model-1 achieves higher test accuracy (92.72% vs. 91.37%)** and stronger consistency across all classes, making it the better-performing model for this specific dataset. The most notable performance gap appears in categories where fine-grained textural differences matter, such as *T-shirt/top*, *Pullover*, and *Shirt*. These classes are visually similar and require the model to preserve subtle pixel-level variations that differentiate necklines or sleeve styles. Model-1's shallower architecture retains more of these high-frequency features, resulting in stronger discrimination where texture dominates over shape.

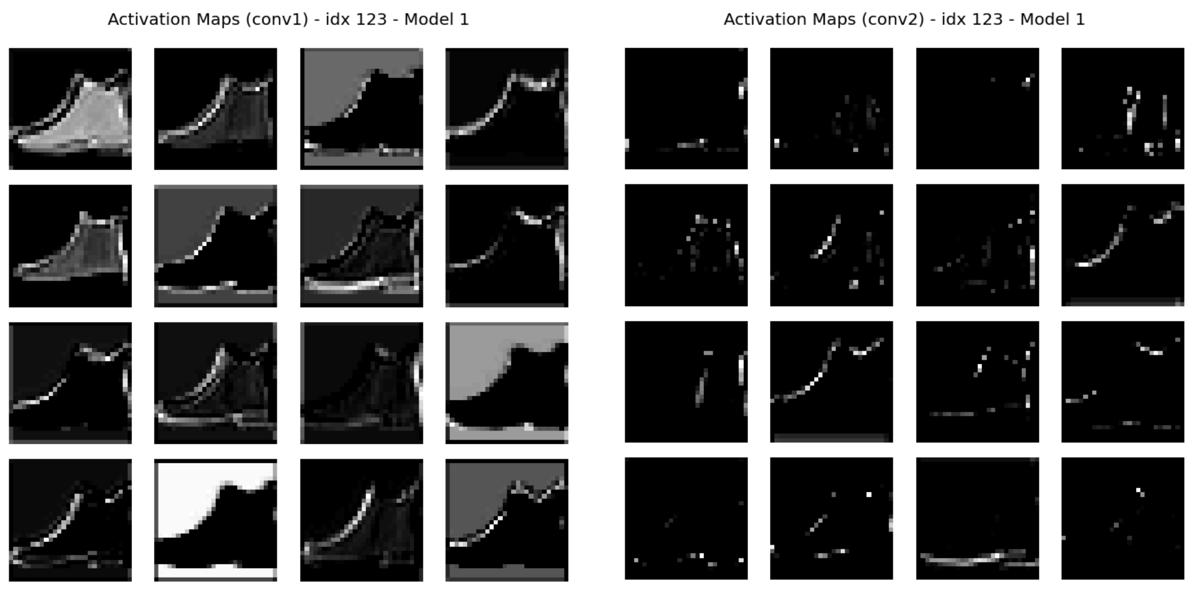

Activation Maps (conv1) - idx 123 - Model 1
Activation Maps (conv2) - idx 123 - Model 1

Figure 8. Activation Maps of Model 1's First Convolutional Layer and Second Convolutional Layer

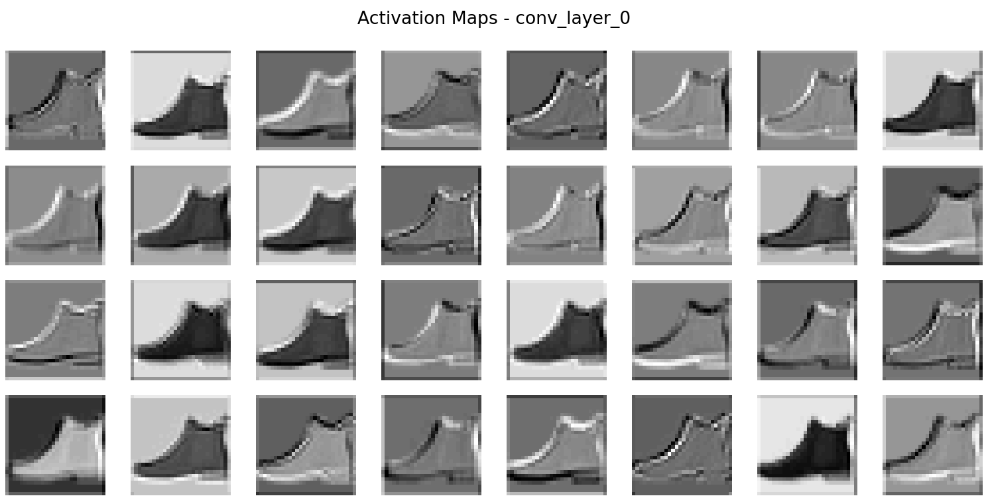


Figure 9. Activation Map of Model 2 In The First ConvBNReLU Block

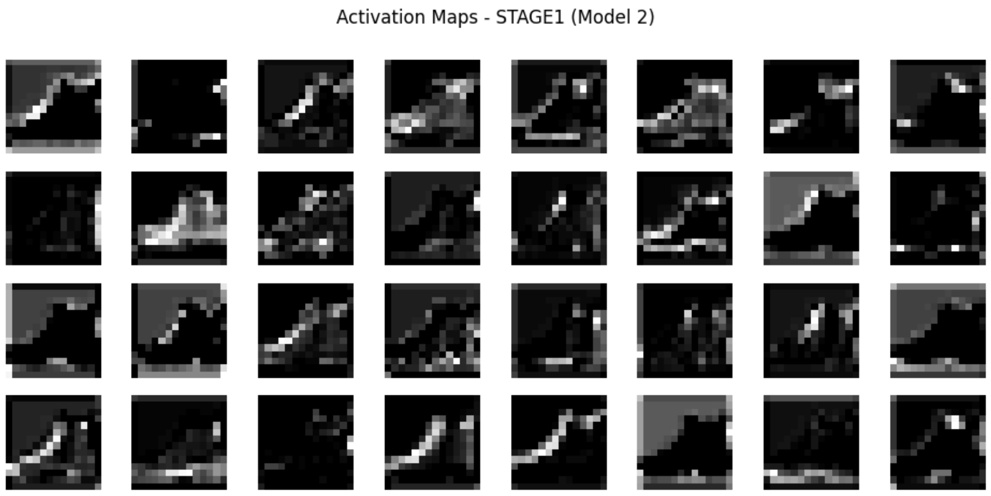


Figure 10. Activation Map of Model 2 After The First ConvBNReLU Block

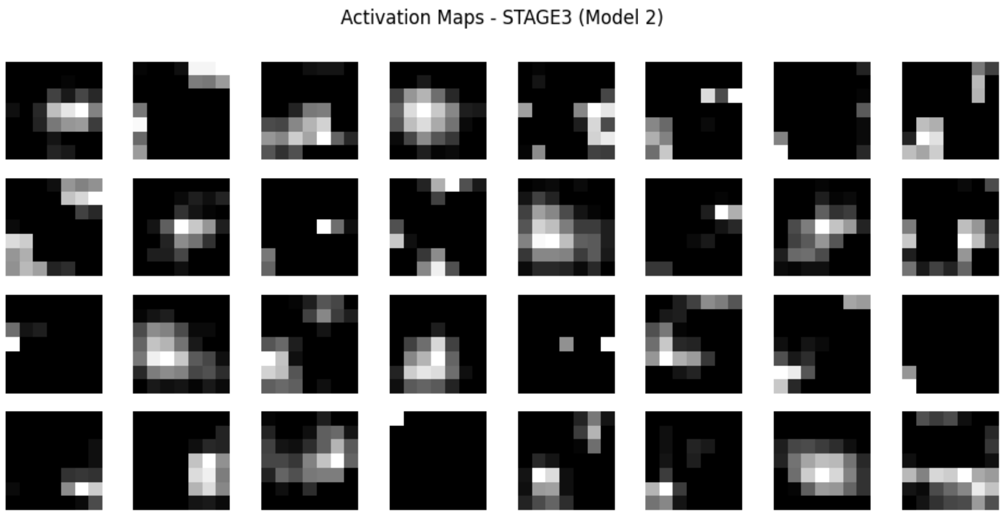


Figure 11. Activation Map of Model 2 After The Third ConvBNReLU Block

However, while Model-2 does not outperform Model-1 in global accuracy, it presents several meaningful advantages that align with modern CNN design practice. Model-2 delivers higher class-wise f1-scores in shape-driven categories (e.g., *Trouser*, *Sneaker*, *Bag*) where structural outlines are more informative than texture. This suggests that the deeper architecture enhances the network's ability to extract richer semantic representations when silhouette and form are the primary discriminative cues. Furthermore, Model-2 incorporates stronger regularization through staged dropout and weight decay, which reduces overfitting tendencies in later epochs. Although this regularization slightly lowers training accuracy, it improves stability under stronger data augmentation and results in more controlled convergence. In terms of extensibility, Model-2 resembles architectures used in contemporary CNNs (e.g., progressively deepened feature extractors), making it a stronger foundation for future scaling, especially if the dataset complexity increases or if color/high-resolution inputs are introduced.

The unexpected outcome is that deeper capacity did not translate into improved performance on Fashion-MNIST. This limitation primarily stems from dataset characteristics: the images are low-resolution and grayscale, and several classes differ only in subtle texture patterns that deeper layers tend to suppress, especially under progressive dropout. Additionally, the dataset's scale (60k images) and balanced class distribution reduce the need for architectural depth; gains from increased complexity diminish once the dataset's variability is sufficiently modeled. As a result, Model-1 remains optimally aligned with the dataset's representational demands.

In conclusion, Model-1 performs better for Fashion-MNIST due to its ability to preserve texture-level information essential for differentiating visually similar classes. For low-resolution grayscale datasets with limited intra-class variability, such as Fashion-MNIST, increasing network depth can introduce unnecessary representational capacity, leading to diminishing returns or even performance degradation unless paired with strong regularization and advanced data augmentation. Meanwhile, Model-2 demonstrates structural advantages, improved robustness, deeper semantic extraction, and architectural scalability, which position it as a more forward-looking design despite lower accuracy in this specific setting. Future work incorporating higher-resolution data, color channels, or domain-specific augmentations may allow Model-2's deeper architecture to unlock its theoretical strengths more fully.

6. Access to Project Repository

All source code, Jupyter notebooks, model outputs, and the final version of this report are available in the project's GitHub repository:
https://github.com/fashionmnist-teamCansuCuluMinaEzoAyci/fashionMNIST

This repository contains the implementation of both models, training pipelines, visualization components, and all related project files.

7. References

Ioffe, S., & Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. In Proceedings of the 32nd International Conference on Machine Learning (ICML).

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

Lin, M., Chen, Q., & Yan, S. (2014). *Network In Network*. arXiv preprint arXiv:1312.4400.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). *Dynamic Routing Between Capsules.* Advances in Neural Information Processing Systems.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.* arXiv preprint arXiv:1708.07747.

PyTorch MNIST Baseline Repository. (2020). **GitHub**.
https://github.com/LucasVandroux/Fashion-MNIST-PyTorch

PyTorch MNIST Baseline Repository. (2024). **GitHub**.
https://github.com/SatvikPraveen/FashionMNIST-Analysis?utm