Benchmarking Lightweight CNN Architectures on FashionMNIST

Oskar N. L. Edén Wallberg [*] Alexander E. Karolin [†]

NTU Nanyang Technological University
School of Computer Science and Engineering

Abstract

This report presents a comparative analysis of lightweight neural network architectures for image classification on the FashionMNIST dataset. The study evaluates six models: LeNet-5, ResNet-Mini, DenseNet-Tiny, MobileNet-Lite, ViT-Tiny, and a custom baseline CNN, each adapted for 28×28 grayscale inputs. All models were trained using the same early-stopping protocol and fixed train/test split, ensuring fair benchmarking across architectures.

Performance was assessed in terms of training and test accuracy, convergence behavior, computational efficiency, and parameter count. Results show that ResNet-Mini achieved the best overall trade-off between accuracy and efficiency, outperforming deeper or more complex models like ViT-Tiny despite having fewer parameters. MobileNet also performed well in accuracy but incurred higher training and inference costs. The custom CNN provided a strong performance-to-complexity ratio, while LeNet-5 demonstrated rapid training but signs of overfitting.

These findings reinforce the importance of architectural design in achieving efficient and accurate learning under resource constraints and highlight that parameter count alone is not a sufficient proxy for performance in small-scale vision tasks.

GitHub: The code for this project is available at:
`github.com/codezart44-school/SC4001_PROJECT`

1 Introduction

Convolutional Neural Networks (CNNs) have become the cornerstone of modern computer vision, enabling breakthroughs in image classification, object detection, and segmentation tasks. As deep learning has matured, the focus has shifted not only toward accuracy but also toward efficiency, particularly in applications requiring deployment on edge devices, mobile platforms, or web services, such as e-commerce image tagging or real-time product classification.

The FashionMNIST dataset serves as a common benchmark for grayscale image classification tasks. Consisting of 70,000 images across 10 classes of clothing items, it offers a clean and compact platform for evaluating model design, training dynamics, and generalization capabilities. Unlike large-scale benchmarks, FashionMNIST allows rapid experimentation while still reflecting meaningful classification challenges.

This study investigates and compares lightweight implementations of several established deep learning architectures, including LeNet-5, ResNet, DenseNet, MobileNet, and Vision Transformers (ViT), alongside a custom-designed baseline CNN. Each model is adapted to handle 28×28 grayscale inputs, and all are evaluated using a unified training pipeline, fixed train/test split, and early stopping criteria.

The primary objective is to benchmark these models in terms of:

- **Accuracy**, both training and test performance
- **Training and inference time**, practical computational cost
- **Parameter count**, model complexity and capacity
- **Efficiency**, training time normalized by number of parameters

By exploring the trade-offs between model complexity, generalization, and runtime, this work aims to highlight which architectures are best suited for small-scale classification tasks where resource constraints and deployment feasibility are important factors. The study also investigates how architectural decisions, such as skip connections, depthwise convolutions, or token-based attention, influence model behavior beyond raw parameter count.

2 Related Work

Several foundational architectures have shaped the development of deep learning for image classification, each introducing key innovations in neural network design.

LeNet-5 was one of the earliest convolutional neural networks, introduced by LeCun et al. [1], demonstrating the feasibility of end-to-end learning on pixel data for character recognition.

ResNet, proposed by He et al. [2], introduced residual connections, enabling the successful training of very deep networks by mitigating vanishing gradient issues.

DenseNet, developed by Huang et al. [3], extended this concept by connecting each layer to all subsequent layers within a dense block, promoting feature reuse and reducing the number of parameters.

MobileNet, presented by Howard et al. [4], targeted mobile and embedded applications using depthwise separable convolutions to dramatically reduce computational cost.

ViT, introduced by Dosovitskiy et al. [5], brought the Transformer architecture to vision tasks by treating image patches as tokens and applying attention-based modeling.

These architectures provide the foundation for the lightweight variants benchmarked in this study.

3 Methods

To evaluate the performance of various lightweight architectures, all models were trained and tested under a unified experimental setup. This section outlines the dataset used, the training procedure, and the architectural details of each model. Emphasis was placed on consistency and comparability across models to ensure fair benchmarking of convergence behavior, predictive performance, and computational efficiency.

[*] ORCID: 0009-0006-6684-7847
[†] ORCID: 0009-0003-7570-1960

3.1 Dataset

The experiments were conducted on the FashionMNIST dataset, which consists of grayscale images of clothing items in 10 categories. Each image has a resolution of 28×28 pixels and a single channel, with input shape $(B, 1, 28, 28)$ where B is the batch size.

The dataset was acquired directly from the PyTorch `torchvision` library and comes pre-divided into a training set of 60,000 samples and a test set of 10,000 samples. No additional data augmentation or preprocessing beyond normalization was applied. All models were trained using this fixed train/test split without cross-validation.

3.2 Training Setup

All models were trained using a unified training and evaluation pipeline to ensure consistency across benchmarks. Training was performed on the same fixed training and test split provided by FashionMNIST, without cross-validation.

Early stopping was used with a patience of 5 epochs, monitoring the test loss to prevent overfitting. For each epoch, both training and test metrics (loss, accuracy, time) were recorded. Inference time was measured on the test set immediately after each training epoch. The same training loop and stopping criteria were applied to all architectures.

3.3 Models

This study benchmarks lightweight versions of several established model architectures, adjusted for the FashionMNIST dataset. The following subsections describe the design and modifications made to each model compared to their original forms. Full implementation details are available in the project repository.

3.3.1 LeNet-5 (Lite Version)

LeNet-5 is a classical convolutional neural network architecture originally introduced by LeCun et al. (1998) for handwritten digit recognition. It uses two convolutional layers with Tanh activations and average pooling, followed by three fully connected layers for classification. The model emphasizes early feature extraction via spatial reduction and nonlinear mapping through dense layers.

Modifications: This version retains the original structure and design of LeNet-5, with the only adjustment being input resolution, adapted from 32×32 to 28×28 to accommodate the FashionMNIST dataset. All other elements, including Tanh activations and average pooling, are preserved for historical fidelity.

```
    Input (28×28) → Conv(6) → AvgPool →
Conv(16) → AvgPool → Flatten → FC(120) →
FC(84) → FC(10)
```

3.3.2 ResNet-Mini (Lightweight)

The ResNet-Mini architecture is a lightweight adaptation of the original ResNet design proposed by He et al. (2015), incorporating residual learning through skip connections to ease optimization and mitigate vanishing gradients. The model is organized into three stages, each composed of two residual blocks with convolutional, batch normalization, and ReLU layers. Downsampling occurs between stages, halving the spatial dimensions and doubling the number of feature channels progressively. A global average pooling layer and a single fully connected layer finalize the classification.

Modifications: Compared to the original deeper ResNet models, ResNet-Mini significantly reduces depth and width, starting with only 16 channels and using a total of six residual blocks. Kernel sizes, activation functions, and residual connections are preserved in concept, but adapted for the 28×28 grayscale FashionMNIST input size.

```
    Input (28×28) → Conv(16) → [ResBlock ×2]
→ Downsample → [ResBlock ×2] → Downsample →
[ResBlock ×2] → AvgPool → FC(10)
```

3.3.3 DenseNet-Tiny (Lightweight)

DenseNet-Tiny is a compact version of the DenseNet architecture introduced by Huang et al. (2018), which uses dense connectivity by concatenating feature maps from all preceding layers within a block. This promotes feature reuse and mitigates vanishing gradients by providing direct paths between layers. The model is composed of three dense blocks, each followed by a transition layer that compresses the number of channels and downsamples the spatial resolution. A final global average pooling layer is followed by a linear classifier.

Modifications: Compared to the original DenseNet, this lightweight version reduces both the number of dense blocks and layers per block, lowers the growth rate, and compresses spatial dimensions more aggressively. The model is also adapted to 28×28 grayscale inputs rather than 224×224 RGB.

```
    Input (28×28) → Conv(8) → [DenseBlock ×1]
→ Transition → [DenseBlock ×1] → Transition
→ [DenseBlock ×1] → AvgPool → FC(10)
```

3.3.4 MobileNet-Lite (Lightweight)

MobileNet-Lite is a streamlined version of the MobileNet architecture introduced by Howard et al. (2017), which uses depthwise separable convolutions to drastically reduce parameter count and computational cost. Each block factorizes a standard convolution into a depthwise convolution for channel-wise filtering and a pointwise convolution for combining channels. The model consists of three such blocks with progressively increasing feature dimensions and two downsampling steps, followed by global average pooling and a linear classifier.

Modifications: This lightweight version reduces the number of depthwise separable blocks, lowers the number of output channels, and uses grayscale 28×28 inputs instead of 224×224 RGB. The architecture omits the deeper tail blocks common in the original MobileNet to minimize training and inference time.

```
    Input (28×28) → Conv(16) → [DWConvBlock
×1] → Downsample → [DWConvBlock ×1] → Downsample
→ [DWConvBlock ×1] → AvgPool → FC(10)
```

3.3.5 ViT-Tiny (Lightweight)

ViT-Tiny is a compact version of the Vision Transformer (ViT) introduced by Dosovitskiy et al. (2021), which models image classification as a sequence modeling task. The input image is divided into fixed-size patches, each projected into an embedding space and treated as a token. A learnable [CLS] token is prepended to the sequence, and positional encodings are added. The sequence is then passed through a stack of transformer encoder blocks, each consisting of multi-head self-attention (MHSA) and a feedforward MLP, with residual connections and layer normalization. The output corresponding to the CLS token is used for classification.

Modifications: This version reduces the number of encoder layers, downsizes the hidden dimensions and number of attention heads, and operates on 28×28 grayscale images with a patch size of 4. These changes significantly reduce the model's depth and parameter count, making it suitable for low-resolution image classification.

```
    Input (28×28) → PatchEmbed(4×4) → Tokenize
→ [EncoderBlock ×3] → CLS Readout → FC(10)
```

3.3.6 Baseline CNN (Custom)

The custom baseline model is a standard convolutional neural network architecture built for simplicity and efficiency. It consists of three convolutional blocks with ReLU activations, batch normalization, and max pooling for downsampling. After the final block, global spatial information is aggregated using adaptive average pooling, followed by a dropout-regularized linear layer for classification. This design balances feature extraction depth with computational efficiency, serving as a lightweight benchmark alongside the other established models.

```
    Input (28×28) → Conv(8) → MaxPool →
Conv(16) → MaxPool → Conv(32) → MaxPool →
AvgPool → FC(10)
```

3.4 Evaluation Metrics

The models are benchmarked using a combination of training performance, generalization ability, and computational efficiency. The following metrics were recorded:

- **Loss (train/test)** – Measures convergence quality and optimization success.
- **Accuracy (train/test)** – Assesses predictive performance and generalization.
- **Time (train/inference)** – Captures computational efficiency in both training and deployment.
- **Number of parameters** – Represents model capacity and potential expressiveness.
- **Time per parameter** – Contextualizes efficiency relative to model size.

These metrics together provide insight into both learning dynamics and efficiency. For instance, large discrepancies between training and test accuracy can indicate underfitting or overfitting depending on parameter count. Smaller models may be constrained by their limited capacity (high bias), while deeper or more expressive models may require more training epochs to reach their potential. Trade-offs between performance and efficiency are made visible by comparing time per parameter alongside accuracy. These evaluations help contextualize whether increased training patience or architectural capacity would yield further improvements.

4 Results

This section presents the training performance, test accuracy, runtime, and parameter efficiency of each model. Two main tables summarize the final loss, accuracy, and timing statistics, while line plots visualize the evolution of accuracy and training time over epochs.

Table 1: Train and test loss and accuracy for each model architecture, along with the number of trainable parameters. Accuracy values correspond to the final epoch before early stopping. Models like ResNet and MobileNet achieve high test accuracy with relatively few parameters, while LeNet-5 shows signs of overfitting due to a significant train-test gap.

Model	Train Loss	Test Loss	Train Acc.	Test Acc. (Epoch)	Parameters
LeNet-5 (lite)	0.1931	0.3203	92.74%	89.07% (E=16)	62K
ResNet (lite)	0.2596	0.2488	90.62%	91.10% (E=29)	11K
DenseNet (lite)	0.3372	0.3009	88.01%	89.71% (E=18)	12K
MobileNet (lite)	0.2259	0.2464	91.91%	91.55% (E=41)	14K
ViT (tiny)	0.2265	0.3306	91.41%	88.09% (E=25)	106K
Custom CNN	0.2616	0.2790	90.53%	90.04% (E=13)	6K

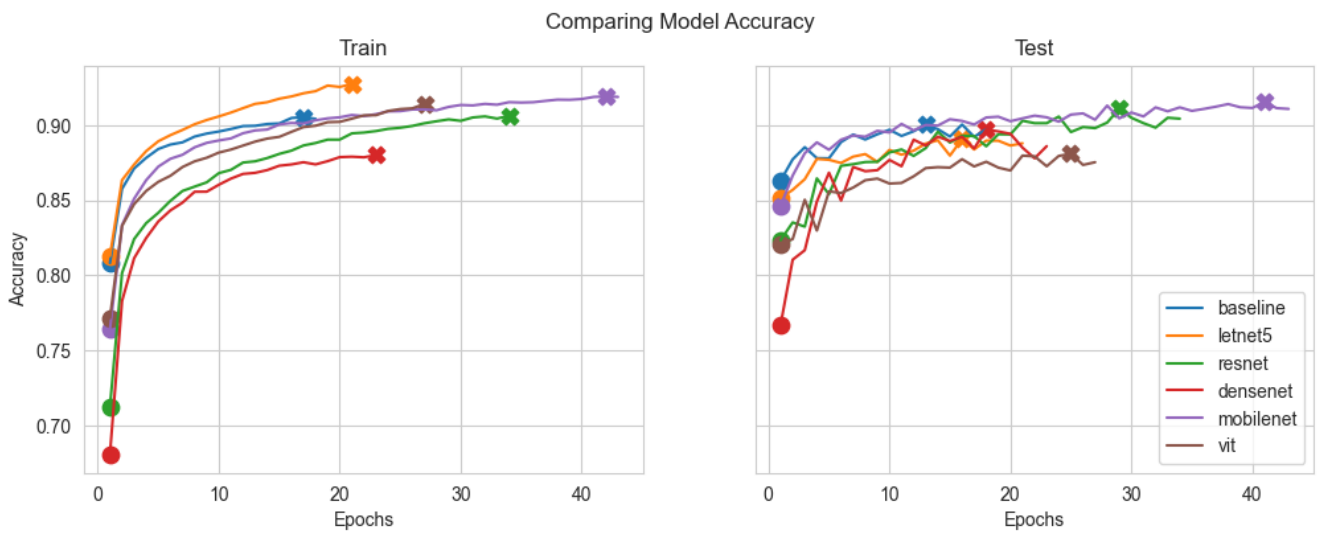

Figure 1: Increase in model accuracy over training.

Table 2: Per-epoch training time (mean ± standard deviation), inference time, total number of epochs, and total training time for each model. Also shown are the number of parameters and a normalized training efficiency metric (train time per 1K parameters). ResNet and Custom CNN are particularly efficient, while ViT-Tiny incurs higher cost per epoch but has much greater capacity.

Model	Train Time	Infer. Time	Epochs	Total Train	Train Time/Param.
LeNet-5 (lite)	20.29 ± 0.65 s	1427 ± 77 ms	21	426 s	329 µs/K
ResNet (lite)	81.93 ± 2.77 s	3981 ± 205 ms	34	2786 s	7186 µs/K
DenseNet (lite)	138.31 ± 3.33 s	6958 ± 320 ms	23	3181 s	11856 µs/K
MobileNet (lite)	164.19 ± 2.78 s	9590 ± 428 ms	43	7060 s	11548 µs/K
ViT (tiny)	136.28 ± 4.78 s	7103 ± 439 ms	27	3680 s	1291 µs/K
Custom CNN	25.8 ± 1.3 s	1755 ± 66 ms	18	464 s	4076 µs/K

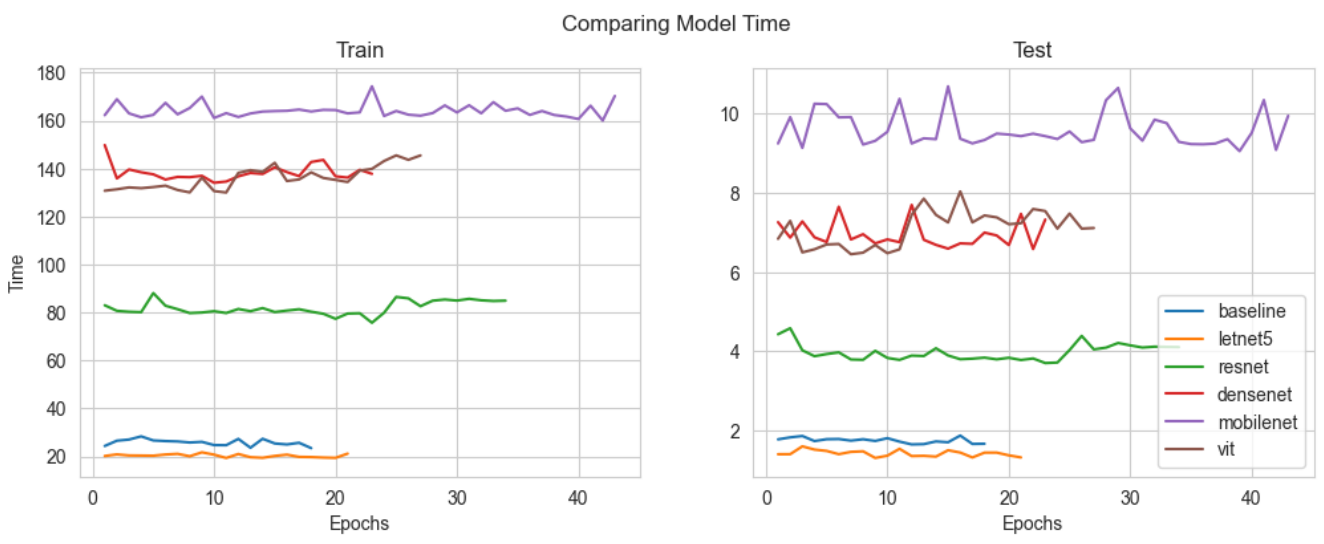

Figure 2: Time per epoch over training.

Table 1 shows that ResNet and MobileNet achieved the highest test accuracy, with ResNet slightly outperforming despite having significantly fewer parameters. LeNet-5 achieved the best training accuracy, but with a notable drop on the test set, suggesting signs of overfitting. The custom CNN achieved a competitive balance between performance and simplicity, using only 6K parameters.

Figure 1 plots model accuracy over time. Most models converge within 20–30 epochs, with ViT showing slower gains and lower generalization despite its large capacity. LeNet-5 and MobileNet show rapid early gains, though LeNet's test performance plateaus early.

Table 2 details the computational costs. MobileNet and DenseNet were the most time-intensive to train and infer, while LeNet-5 and the baseline CNN were the fastest. When normalized by parameter count, ResNet and the custom CNN demonstrate the most efficient learning.

Figure 2 visualizes per-epoch training and inference time. MobileNet exhibits consistently high times per epoch, while ResNet and the baseline maintain low and stable timings throughout training.

These results highlight trade-offs between model complexity, performance, and efficiency, which are discussed in more depth in the next section.

5 Discussion

5.1 Overview and Study Objective

This study evaluated several lightweight neural network architectures on the FashionMNIST dataset, a benchmark for grayscale image classification of apparel items across 10 classes. The goal was to compare and assess the trade-offs between model complexity, training and inference time, and predictive performance.

Lightweight variants of LeNet-5, ResNet, DenseNet, MobileNet, and ViT were implemented and benchmarked alongside a custom-designed CNN baseline. All models were adjusted to accept 28×28 grayscale input and trained using the same early-stopping criteria and evaluation pipeline. By comparing these diverse architectures under uniform conditions, this study offers insight into which designs are most efficient and effective for resource-constrained computer vision tasks typical in e-commerce or mobile deployment contexts.

5.2 Accuracy vs. Model Complexity

A central question in model design is how parameter count relates to predictive performance. As noted in related work [6], there is often a correlation between the number of trainable parameters and the accuracy a model can achieve on FashionMNIST. However, this trend is not absolute.

In this study, ResNet-Mini achieved one of the highest test accuracies (91.10%) while using only 11K parameters, far fewer than ViT-Tiny's 106K parameters, which resulted in a lower 88.09% test accuracy. Similarly, MobileNet performed well despite its relatively modest size. These results demonstrate that architectural design, such as the use of skip connections or efficient convolution types, can lead to more effective feature extraction than simply increasing the model's capacity.

By contrast, LeNet-5, while fast and reasonably accurate, showed a notable gap between training and test accuracy, suggesting a tendency to overfit due to its reliance on large fully connected layers. This is consistent with its original design, which predates more modern regularization and feature extraction strategies.

These results highlight that **architectural efficiency** often matters more than parameter count, especially when working within constrained datasets or real-time application settings.

5.3 Time Efficiency and Training Dynamics

Beyond accuracy, training and inference time are critical considerations for practical deployment. As expected, models with more layers or higher complexity generally took longer to train. MobileNet and DenseNet were the most time-consuming per epoch, both exceeding 130 seconds, while LeNet-5 and the custom CNN completed epochs in roughly 20–25 seconds.

However, the relationship between model size and runtime was not strictly linear. For instance, MobileNet and DenseNet had similar parameter counts (14K and 12K respectively), yet MobileNet exhibited significantly longer training and inference times. This can be attributed to MobileNet's depthwise separable convolutions, which, while efficient in parameter usage, can incur greater overhead in practice due to their less optimized computation paths on some hardware.

Inference time also diverged from training cost in some models. ViT, with the highest parameter count, had a slower training loop but only moderately high inference time, owing to its token-based architecture. ResNet and the custom CNN stood out as particularly efficient, achieving competitive performance while maintaining some of the lowest per-epoch and per-parameter training times.

5.4 Early Stopping and Training Stability

Early stopping was used with a patience of five epochs in this study to prevent overfitting and reduce unnecessary training cycles. While this is generally effective for lightweight models, it may have prematurely halted training in models with higher capacity or slower convergence, particularly ViT and MobileNet.

Figure 1 shows that these models were still improving steadily when stopped. A longer patience setting could have allowed more of their learning potential to be realized, especially given their smoother, gradual training curves. Conversely, smaller models like the custom CNN and LeNet converged quickly, making them well-suited to early stopping and short training cycles.

Notably, test loss trends suggest that ViT may have been over-regularized early on, or experienced noise in validation loss that triggered stopping before convergence. In future studies, adaptive patience or validation smoothing could improve robustness for such models.

These observations underscore that stopping criteria should be adjusted in proportion to model complexity. Fixed patience may penalize deeper or attention-based models disproportionately, limiting their ability to reach full capacity.

5.5 Model Design Observations

Each model architecture in this study demonstrates distinct design choices that influence both learning behavior and computational efficiency. LeNet-5, the oldest architecture tested, relies heavily on fully connected layers, accounting for the majority of its parameters, and shows signs of overfitting despite its relatively small size. Its feature extraction capacity is limited, and modern replacements like batch normalization or ReLU activations are absent, affecting generalization.

ResNet-Mini leverages residual connections to enable deeper effective learning without gradient degradation. This design proves highly efficient, allowing the model to generalize well with very few parameters. The downsampling after each residual stage also helps reduce spatial resolution and thus computational load, making it one of the fastest and most effective models in the benchmark.

DenseNet-Tiny, while conceptually elegant due to its dense connectivity pattern, showed diminished efficiency relative to its performance. The cost of repeated concatenation and growth in channel width may outweigh its benefits for small-scale tasks like FashionMNIST.

MobileNet-Lite demonstrated strong test performance but suffered from longer training and inference times, potentially due to less optimized implementations of depthwise separable convolutions in the runtime environment. Nevertheless, it offers a good balance between accuracy and parameter compactness.

Finally, ViT-Tiny stands apart as a transformer-based model without convolutional layers. Its performance lagged behind the CNNs despite having the most parameters. This is consistent with literature noting that ViTs generally require more data to fully leverage their attention mechanisms and positional encodings, which may be suboptimal on small-resolution, low-diversity datasets like FashionMNIST.

5.6 Summary of Trade-offs

The results highlight the classic engineering trade-off between accuracy, training time, and model complexity. While larger models often hold more representational power, their benefits are not always realized unless supported by sufficient data and training patience. In contrast, simpler models like ResNet-Mini and the custom CNN can perform competitively with far fewer parameters when carefully structured.

Figure 1 and Table 1 show that test accuracy alone does not reflect training efficiency or generalization behavior. Similarly, Figure 2 and Table 2 indicate that runtime is influenced not just by model size, but by architectural details and implementation factors.

Ultimately, model selection should be guided by application constraints: where inference speed is critical, simpler architectures like LeNet-5 or the custom CNN may be preferred. Where accuracy is paramount and time is less constrained, ResNet-Mini or MobileNet may offer better returns. For transformer-based models like ViT, further tuning, data augmentation, or larger datasets may be necessary to fully unlock their potential.

6 Future Work

Recommendations for future work:

- Training with MixUp or other data augmentation techniques
- Analyzing class-wise performance using confusion matrices and F1 scores
- Introducing cross-validation for more statistically robust comparisons
- Performing architectural ablations and layer/parameter sensitivity studies
- Extending training patience to better assess learning capacity of larger models

This study provides a baseline comparison of several lightweight model architectures under a unified training pipeline, but there are several promising directions for extending the work.

First, introducing data augmentation strategies such as MixUp could improve generalization and provide insight into how different architectures respond to increased data variation. This is particularly relevant for small models, which may benefit from more diverse training signals.

Second, evaluating per-class performance metrics—such as precision, recall, specificity, and class-wise F1 scores—would enable a more granular understanding of where each model succeeds or fails. A confusion matrix analysis could reveal systematic errors, such as certain classes being commonly misclassified, and whether such patterns vary across architectures.

Another important extension would be to incorporate controlled cross-validation splits during training. This would improve the statistical robustness of results by reducing sensitivity to a particular train-test division, and provide confidence intervals on the reported metrics.

In addition, more detailed ablation studies on specific model families (e.g., adding layers to ResNet-Mini or increasing the growth rate in DenseNet-Tiny) could reveal how architectural changes affect learning dynamics and performance trade-offs. Such controlled experiments could guide the design of task-specific lightweight networks.

Finally, future experiments could explore the effect of training for longer periods by increasing early stopping patience. Some models—especially those with more parameters—may not reach their full potential under the limited training time used here. Analyzing how patience interacts with model capacity would help disentangle underfitting due to optimization time from architectural limitations.

7 Conclusion

This study compared lightweight implementations of five popular neural network architectures, LeNet-5, ResNet, DenseNet, MobileNet, and ViT, alongside a custom CNN baseline, all trained on the FashionMNIST dataset. The evaluation focused on predictive performance, training and inference time, and model complexity, measured by the number of trainable parameters.

ResNet-Mini emerged as the strongest overall performer. Despite having only 11K parameters, it achieved the highest test accuracy (91.10%) and exhibited excellent generalization, efficiency, and convergence behavior. Its use of skip connections and downsampling enabled both fast training and robust learning with minimal overfitting.

MobileNet-Lite also performed well in terms of accuracy (91.55%) but required significantly longer training and inference times. ViT-Tiny, while parameter-rich, underperformed on the test set, likely due to its architectural bias toward large datasets and higher resolution inputs. LeNet-5 was computationally fast and achieved strong training accuracy but showed signs of overfitting and limited feature capacity. The custom CNN, although the simplest model tested, achieved solid accuracy (90.04%) with the fewest parameters, making it a compelling option for highly constrained environments.

In summary:

- **Best Accuracy:** ResNet (91.10%)
- **Fastest Training:** LeNet-5 and Custom CNN
- **Most Parameter-Efficient:** Custom CNN (6K params, 90.04%)
- **Most Balanced Performer: ResNet-Mini**, offering a strong trade-off between accuracy, speed, and simplicity.

These findings suggest that well-structured convolutional networks, particularly those using residual connections, remain highly effective for small-scale vision tasks, outperforming more complex or modern designs when data and compute are limited.

References

[1] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[2] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[3] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[4] Andrew G Howard et al. "MobileNets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[5] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[6] Kalyani Kadam, Apurva Deshmukh, and Nilesh Patil. "Comparative Analysis of Lightweight Deep Learning Models for Fashion Image Classification". In: *Mathematics* 12.20 (2024), p. 3174. DOI: `10.3390/math12203174`. URL: `https://www.mdpi.com/2227-7390/12/20/3174`.

8 Appendex

Model Architecture Comparison (Lite vs. Original)

LeNet-5 Architecture Comparison

Feature	LeNet-5 (Original)	LeNet-5 (Lite)
Input size	32×32 grayscale	28×28 grayscale
Conv layers	2	2
Activation	Tanh	Tanh
Pooling	Average	Average
FC layers	3 (120 → 84 → 10)	3 (120 → 84 → 10)
Classifier head	FC(10)	FC(10)

ResNet Architecture Comparison

Feature	ResNet (Original)	ResNet-Mini	Comment
Input size	224×224 RGB	28×28 grayscale	FashionMNIST adaptation
Initial channels	64	16	Downscaled for efficiency
Residual blocks	34–101	6	Depth reduction
Stem conv	7×7, stride 2	1×1, stride 1	Simplified stem

DenseNet Architecture Comparison

Oskar N. L. Edén Wallberg & Alexander E. Karolin
oskarew@kth.se & akarolin@kth.se

Feature	**DenseNet (Original)**	**DenseNet-Tiny**
Input size	224×224 RGB	28×28 grayscale
Dense blocks	4	3
Layers per block	6–32	3
Growth rate	32	8
Initial channels	64	8
Compression factor	∼2	4
Classifier head	FC(1000)	FC(10)

MobileNet Architecture Comparison

Feature	**MobileNet (Original)**	**MobileNet-Lite**
Input size	224×224 RGB	28×28 grayscale
Depthwise separable blocks	13+	3
Initial channels	32	16
Final channels	1024	128
Classifier head	FC(1000)	FC(10)
Global average pooling	Yes	Yes

Vision Transformer (ViT) Architecture Comparison

Feature	**ViT (Original)**	**ViT-Tiny**
Input size	224×224 RGB	28×28 grayscale
Patch size	16×16	4×4
Embedding dimension	768	64
Attention heads	12	8
Encoder layers	12	3
MLP hidden dim	3072	128
Tokens (incl. CLS)	197	50
Classifier head	FC(1000)	FC(10)

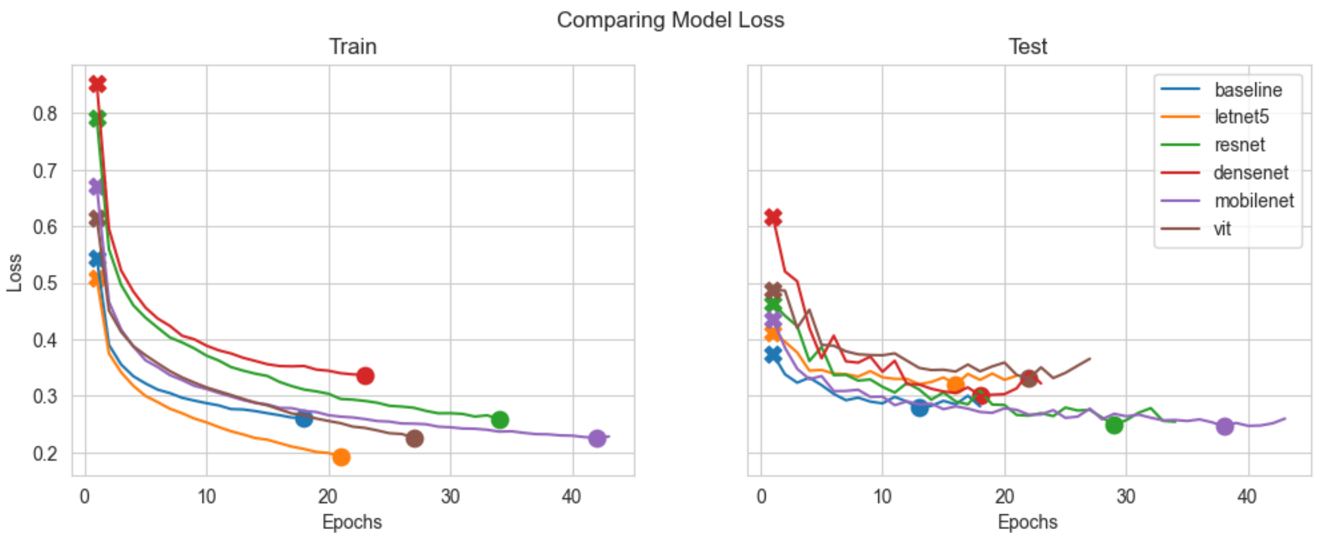

Figure 3: Reduction in model loss over training.