Deep Neural Network Report

Lorenzo Tucceri Cimini

July 5, 2025

1 Introduction

In this work, we aim to analyze how weight initialization and different training strategies can influence the behavior of Convolutional Neural Networks (CNNs). To this end, two different architectures were designed and tested using three configurations that differ in terms of initialization methods and the handling of the first layer. The experiments were conducted using the *FashionMNIST* dataset.

2 Architectures

Two different architectures, named **A1** and **A2**,were used in the experimentation. Both architectures were trained using a batch size of 128.

Architecture A1 consists of:

1. A convolutional layer with 5 filters of size 3x3;
2. A fully connected layer;
3. An output layer with *Softmax* activation function;
4. Loss function: *Cross Entropy*.

Architecture A2 has a deeper structure:

1. A first convolutional layer with 5 filters of size 3x3;
2. A second convolutional layer with 7 filters of size 3x3;
3. A fully connected layer;
4. An output layer with *Softmax* activation function;
5. Loss function: *Cross Entropy*.

Regarding initialization strategies, two main types were adopted: **By-Hand** and **Default**.

- **By-Hand**: kernels were manually initialized using only values 0 and 1. Each kernel has a different pattern, chosen manually based on simple visual logics. Specifically, the following five patterns (3x3 size) were used:
  - **Kernel 0** – Main diagonal:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
  - **Kernel 1** – Secondary diagonal:
$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$
  - **Kernel 2** – Top edge:
$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
  - **Kernel 3** – Left edge:
$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$
  - **Kernel 4** – All ones:
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
- **Default**: uses the automatic *Kaiming* initialization integrated within the network.

Based on these strategies, three main configuration schemes were defined:

- **HF (Hand-Frozen)**: the first layer of the network is initialized with the By-Hand strategy, while subsequent layers use the Default strategy. Only layers after the first are trained; the first layer remains frozen.
- **HT (Hand-Trained)**: as in the previous case, the first layer is initialized with the By-Hand strategy and the others with Default. However, in this scheme **all** layers, including the first, are trained.
- **DT (Default-Trained)**: all layers, including the first, are initialized with the Default strategy. All layers are trained.

3 Preliminary Analysis

Before the training phase, a preliminary analysis was performed on the network weights by evaluating the norm of the difference between corresponding weights. In particular:

- The weights of the **first layer** are identical across all **A* - H*** configurations, i.e.:
$$\|\mathbf{W}_1^{(i)} - \mathbf{W}_1^{(j)}\| = 0 \quad \forall i, j \in \text{H}^*$$
where $\mathbf{W}_1^{(i)}$ and $\mathbf{W}_1^{(j)}$ represent the first layer weights in different CNNs belonging to the same group (H*).
- Similarly, all **layers following the first** within the same group also have identical initial weights:
$$\|\mathbf{W}_k^{(i)} - \mathbf{W}_k^{(j)}\| = 0 \quad \forall k > 1,\ i, j \in \text{H}^*$$
- Finally, the weights of the **first layer** of the **A* − DT** configurations are also identical:
$$\|\mathbf{W}_1^{(i)} - \mathbf{W}_1^{(j)}\| = 0 \quad \forall i, j \in \text{DT}$$

This verification ensures that the weight initialization respects the constraints imposed by the defined strategies (By-Hand, Default, HF, HT, DT).

3.1 Weight Initialization and Comparison

The initialization of the model weights was managed through a Python script that ensured consistency across different configurations. The procedure involved the following steps:

- A folder `weights/` was created to save the weights of the initialized architectures.
- Functions were defined to:
  - save the weights of a specific layer using `torch.save`;
  - load saved weights into an existing layer using `torch.load`;
  - compare two layers by computing the norm of the difference: $\|\mathbf{W}_1 - \mathbf{W}_2\|$.
- The following layers were initialized and saved:
  - `NetA1ByHand`: `conv1`, `fc1`
  - `NetA2ByHand`: `conv2`, `fc1`
  - `NetA1Default`: `conv1`

This saving is performed only if the files do not already exist, preventing overwriting.

- Previously saved weights were reloaded into the models. For example, `A1-HF` and `A1-HT` loaded the same weights in `conv1` and `fc1`, while `A1-DT` loaded weights from `NetA1Default`.
- Finally, a comparison was performed between corresponding layers of different models. The function `compare_weights()` calculates the norm of the difference between weight tensors, showing the value $\|\mathbf{W}^{(i)} - \mathbf{W}^{(j)}\|$ between:
  - `conv1` and `fc1` between A1-HF and A1-HT;
  - `conv1` between A1-HF/HT and A2-HF/HT;
  - `conv2`, `fc1` between A2-HF and A2-HT;
  - `conv1` between A1-DT and A2-DT.

This procedure ensured that all architectures started from initial conditions consistent with the chosen strategy, and that differences in the final results can be exclusively attributed to configuration and training, not to random variations in the weights.

4 Experiments

All models were trained for a total of 20 epochs. At the end of training, the final values of accuracy and loss were recorded for each architecture-configuration combination. The following table and chart summarize the results obtained:

Model	**Accuracy (%)**	**Loss**
A1-HF	84.06	0.4647
A1-HT	84.26	0.4475
A1-DT	87.15	0.3668
A2-HF	89.03	0.3132
A2-HT	88.92	0.3129
A2-DT	87.64	0.3381

Table 1: Final results (after 20 epochs) for each model

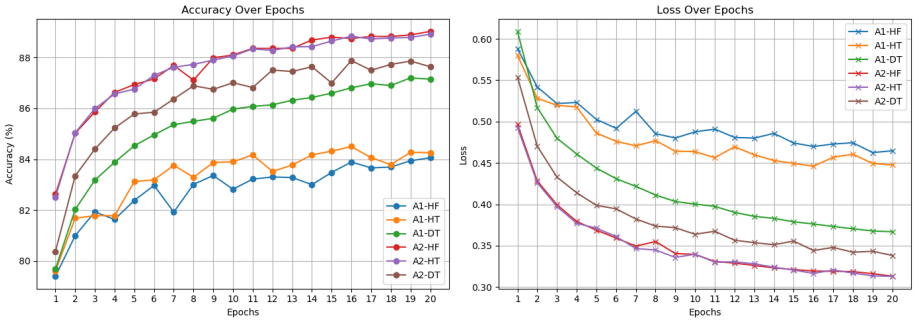

Figure 1: Total experiments

From the table, the following observations can be made:

- **A1 vs A2**: In general, models based on architecture **A2** perform better than those using **A1**, both in terms of accuracy and final loss. This is expected, since A2 includes a deeper structure (two convolutional layers), and thus a greater representational capacity than A1, which has only a single convolutional layer followed by a fully connected layer.
- **HF, HT, DT Strategies**: Within each architecture, differences emerge among the three initialization/training strategies (HF, HT, DT). These variations will be analyzed in more detail in the next section.

4.1 Experiment 1 – Inset Comparison

This experiment compares models within the same architecture, i.e., models that differ only by initialization/training strategy.

4.1.1 Set 1 (Architecture A1)

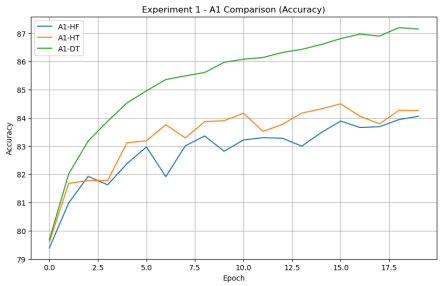

Figure 2: Set A1 Accuracy

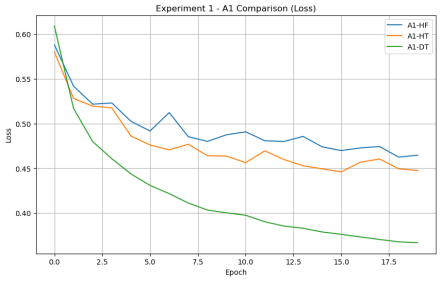

Figure 3: Set A1 Loss

Within the A1 set, model A1-DT achieves the best performance in terms of both accuracy (up to 87.20%) and loss (around 0.3668). This suggests that, even with a relatively simple network (only one convolutional layer), a dynamic training strategy (DT) can lead to more effective learning than static, non-trained (HF) or trained (HT) fixed initializations.

Looking at the progression over time, A1-DT shows a more consistent and steady growth in accuracy compared to the other models, with a sharper drop in loss during the early epochs. In contrast, A1-HF and A1-HT show more pronounced fluctuations and a slowdown in learning after the first 10 epochs. A1-HF has the flattest trend, indicating difficulty in improving.

4.1.2 Set 2 (Architecture A2)

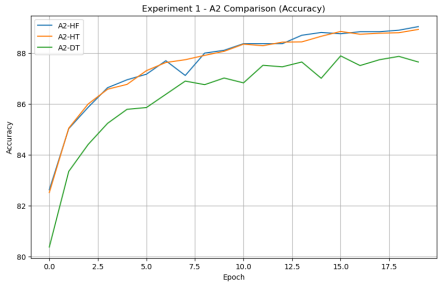

Figure 4: Set A2 Accuracy

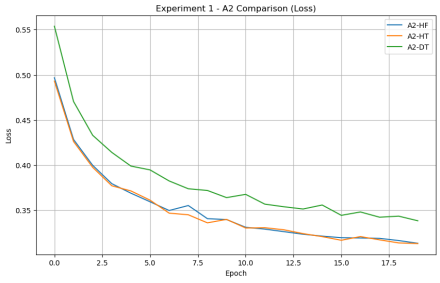

Figure 5: Set A2 Loss

In the A2 set, models A2-HF and A2-HT achieve very similar results (accuracy above 89% and loss around 0.31), slightly outperforming A2-DT. This is interesting, as in contrast with A1, the DT strategy does not seem to provide additional benefits in a deeper network.

The plots show that all A2 models converge quickly: already within the first 5–7 epochs, a significant drop in loss is observed. A2-HF and A2-HT

grow steadily, while A2-DT displays slightly more oscillations and a less decisive trend after the tenth epoch. This reinforces the idea that in complex networks, architectural capacity can compensate for differences due to training strategy.

4.2 Experiment 2 – Architecture Comparison

This experiment compares the two architectures directly, keeping the initialization/training strategy fixed.

4.2.1 HF: A1-HF vs A2-HF

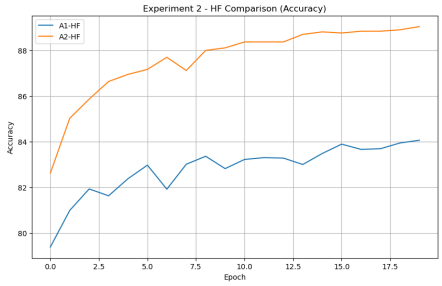

Figure 6: HF Accuracy

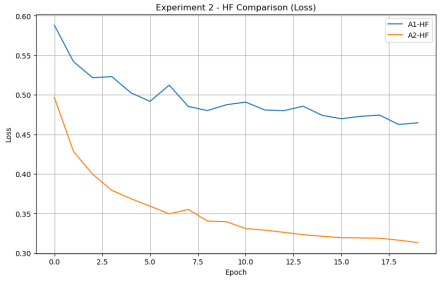

Figure 7: HF Loss

A2-HF clearly outperforms A1-HF: the final accuracy is about 5% higher (89.03% vs 84.06%), and the loss significantly lower.

Looking at the trends, A2-HF shows steady, linear growth across all epochs, while A1-HF is slower and less smooth. The loss for A2-HF drops rapidly within the first 5 epochs and then stabilizes, while A1-HF decreases more slowly and remains higher throughout. This highlights the effectiveness of deeper architectures even under static initialization.

4.2.2 HT: A1-HT vs A2-HT

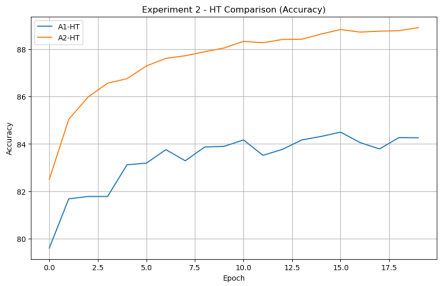

Figure 8: HT Accuracy

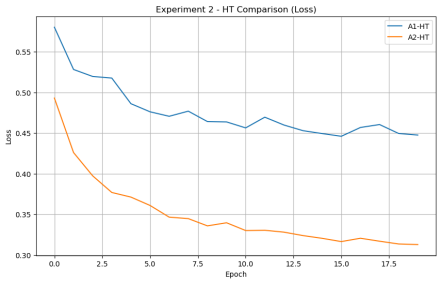

Figure 9: HT Loss

Once again, A2-HT outperforms A1-HT in both accuracy (88.92% 0.4475). Training the first layer alone in A1-HT is not enough to reach the performance level of A2-HT.

Over time, A2-HT shows steady and consistent improvement from the very beginning, with a rapid drop in loss by the 10th epoch. In contrast, A1-HT has a less stable progression: although it improves initially, its accuracy fluctuates in the middle epochs and stabilizes earlier, without reaching A2-HT's performance.

4.2.3 DT: A1-DT vs A2-DT

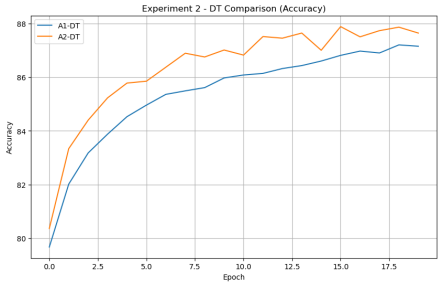

Figure 10: DT Accuracy

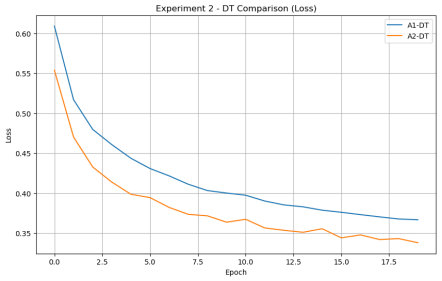

Figure 11: DT Loss

In this final comparison, A1-DT achieves very competitive results (87.15%) compared to A2-DT (87.64%), despite its simpler architecture. This suggests that the DT strategy is particularly beneficial for lighter models, helping compensate for the lack of depth.

Over time, A1-DT shows a steadily increasing accuracy curve, with consistently decreasing loss until the end. A2-DT behaves similarly but is slightly more irregular after the midpoint of training. Both converge effectively, but A1-DT stands out in terms of efficiency relative to its structural simplicity.

4.3 Experiment 3 – Recovery Comparison (A2-HF vs A1-*)

This experiment assesses whether a good initialization (HF) on a deeper architecture (A2) can "recover" the performance of less powerful models that use more advanced training strategies.

4.3.1 A2-HF vs A1-HF

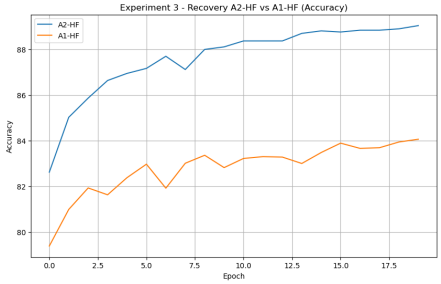

Figure 12: A2-HF vs A1-HF Accuracy

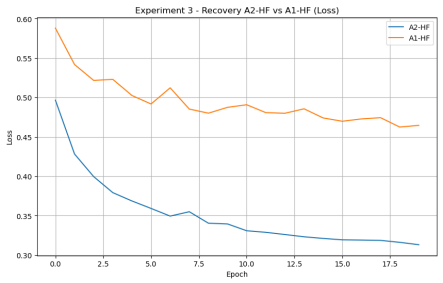

Figure 13: A2-HF vs A1-HF Loss

Here, A2-HF clearly outperforms A1-HF. The deeper network, even with static initialization, achieves significantly better performance in both accuracy and loss.

Over time, A2-HF grows quickly and steadily, with a sharp loss drop in the early epochs. A1-HF, on the other hand, shows slower learning with more oscillations, indicating optimization challenges and lower representational power.

4.3.2 A2-HF vs A1-HT

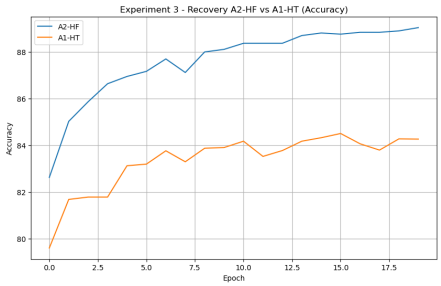

Figure 14: A2-HF vs A1-HT Accuracy

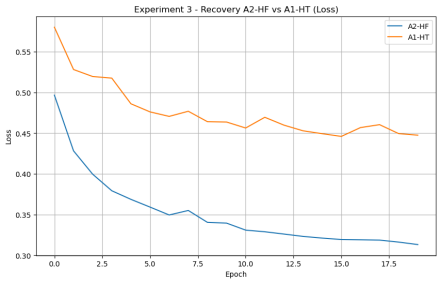

Figure 15: A2-HF vs A1-HT Loss

A2-HF also clearly outperforms A1-HT. Training the first layer in A1-HT is not enough to surpass the performance of A2-HF.

Over time, A2-HF displays a solid and continuous upward trend, while A1-HT develops more irregularly, with minor stagnations. The loss for A2-HF decreases more quickly and settles at better values.

4.3.3 A2-HF vs A1-DT

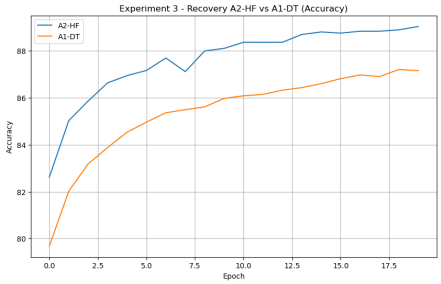

Figure 16: A2-HF vs A1-DT Accuracy

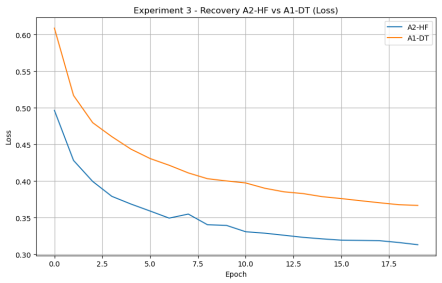

Figure 17: A2-HF vs A1-DT Loss

This is the most balanced comparison. A1-DT performs very well, but A2-HF still outperforms it, suggesting that a good architecture can make up for a lack of advanced training strategies, at least in this context.

Analyzing the plots, A1-DT improves steadily and gradually, while A2-HF shows a sharper, faster improvement in the early epochs. Both models converge well, but A2-HF benefits from architectural advantage despite the static strategy.

In summary, these experiments confirm that:

- The depth of the architecture has a significant impact on performance.

- Initialization and training strategies are especially influential for simpler models.

4.4 Confusion Matrix and Additional Analysis

To further investigate the performance of the best-performing model, **A2-HF**, a confusion matrix was generated using the test set predictions. The matrix provides insight into the distribution of correct and incorrect classifications across the 10 FashionMNIST classes.

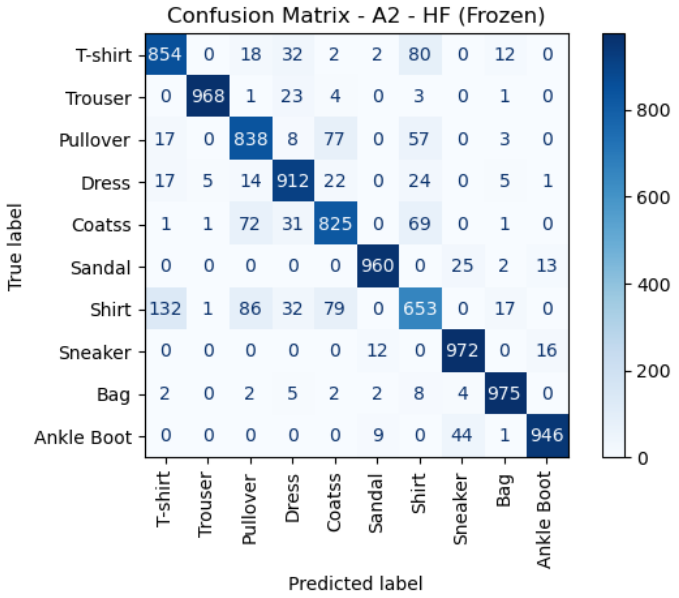

Figure 18: Confusion matrix for model A2-HF on the FashionMNIST test set.

The confusion matrix in Figure 18 shows that the model performs very well overall, with most predictions concentrated along the diagonal, indicating correct classifications.

The only notable—but expected—source of confusion is between the classes **Shirt** and **T-shirt**. Specifically, the model sometimes predicts *T-shirt* when the true label is *Shirt*, with 132 such misclassifications out of a total of over 1000 samples. However, this is relatively minor when compared to the 834 correct predictions for the *T-shirt* class, and can be attributed to the visual similarity between these two categories.

5 Results

The results of the experiments, evaluated in terms of final accuracy, best accuracy achieved during training, and minimum loss, are summarized in Table 2.

Model	**Last Accuracy (%)**	**Best Accuracy (%)**	**Min Loss**
A2-HF	89.03	89.03	0.3132
A2-HT	88.92	88.92	0.3129
A2-DT	87.64	87.88	0.3381
A1-DT	87.15	87.20	0.3668
A1-HT	84.26	84.50	0.4460
A1-HF	84.06	84.06	0.4625

Table 2: Performance metrics of the tested models on the FashionMNIST dataset.

From the table, it is evident that the deeper architecture **A2** generally achieves higher accuracy and lower loss compared to the shallower **A1** architecture. Moreover, within the **A2** architecture, the **DT** configuration shows slightly worse performance compared to the **HF** and **HT** schemes, highlighting the impact of manual kernel initialization on the network's behavior.