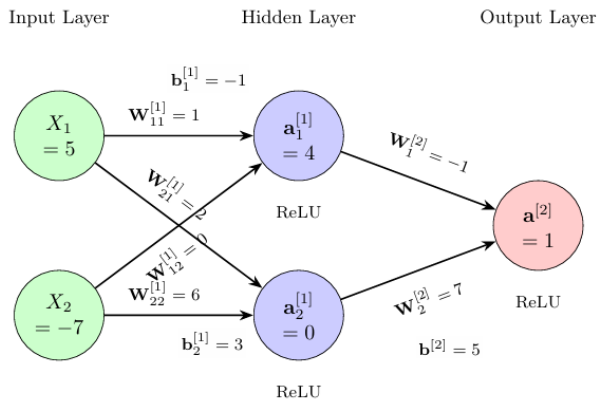
Question 1 – Forward Pass and Convolutional Filters

STAT448 – Assignment 3

David Ewing (82171149)   Lillian Li (32198314)

Question 1A

The input vector passes through two neurons in the hidden layer, each computing a linear transformation using the given weights and biases. ReLU activation is applied to yield hidden layer outputs, which are then passed to a single output neuron. A final ReLU gives the scalar output. The network diagram and final result are shown in Figure 1.



Calculation Steps:

1. **Hidden Layer Computation:** 4 $z_1^{[1]} = W_{11}^{[1]}X_1 + W_{12}^{[1]}X_2 + b_1^{[1]} = 4$
$z_2^{[1]} = W_{21}^{[1]}X_1 + W_{22}^{[1]}X_2 + b_2^{[1]} = -29$
$a_1^{[1]} = ReLU(z_1^{[1]}) = 4$
$a_2^{[1]} = ReLU(z_2^{[1]}) = 0$

2. **Output Layer Computation:** 2 $z^{[2]} = W_1^{[2]}a_1^{[1]} + W_2^{[2]}a_2^{[1]} + b^{[2]} = 1$
$a^{[2]} = ReLU(z^{[2]}) = 1$

Figure 1: Forward pass through two-layer ReLU network (Question 1A)

Question 1B

Each of the three 3×3 filters was convolved with the input image using stride 1 and no padding. ReLU activation followed each convolution. Feature maps were generated and the calculations for the bottom-right output pixel are shown.

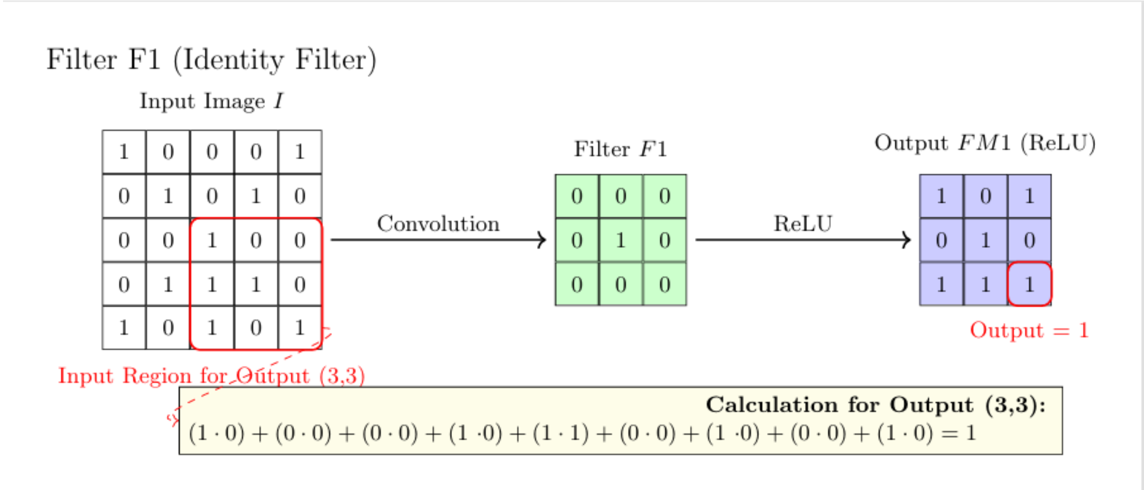


Figure 2: Filter F1 – Identity filter and its feature map (Question 1B)

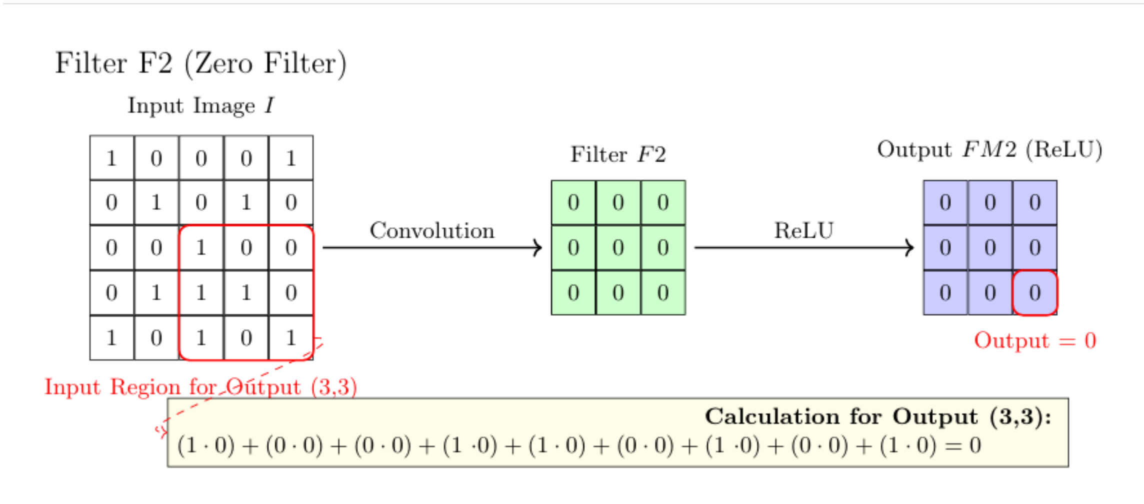


Figure 3: Filter F2 – Zero filter and its feature map (Question 1B)

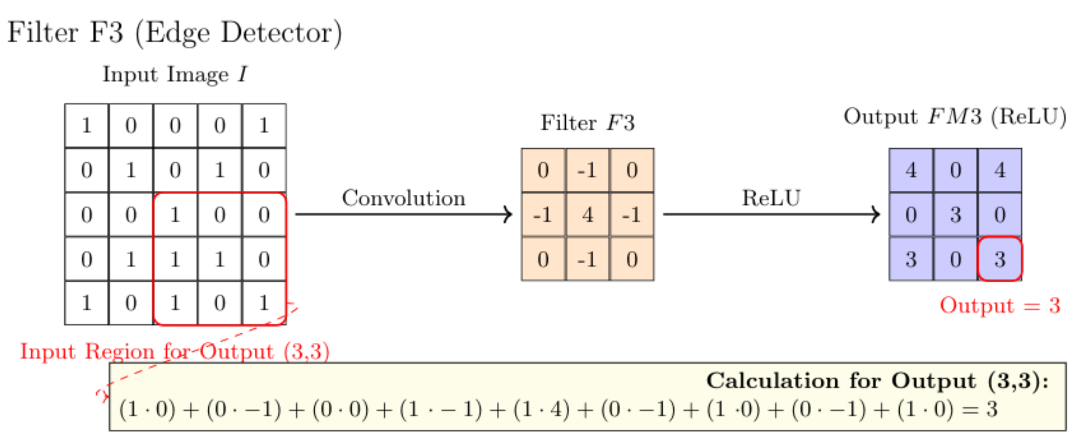


Figure 4: Filter F3 – Edge detection filter and its feature map (Question 1B)

Question 2 - Effect of Architectural and Training Hyper-parameters on Fashion-MNIST Test Accuracy

2025-05-26

Abstract

We revisit CNN hyperparameter tuning on Fashion-MNIST using an Optuna–seeded baseline (filters: 32, kernel-size: 3, dense-units: 512, batch-size: 64, optimiser: Adam). Five one-parameter sweeps were executed. Early stopping kept every run below 90 s while sustaining 91% validation accuracy in all but the `sgd` trial. Merged figures and an impact table summarise the accuracy–speed trade-off in a compact form.

1 Introduction

Bayesian hyper-parameter optimisation has proven effective for neural networks (Akiba, Sano, Yanase, Ohta, & Koyama, 2019). Here, 30 Optuna TPE trials gave a *stable local* starting point; the goal was interpretability, not a global optimum. Five follow-up runs each altered one setting, so any performance delta could be attributed directly to that variable. Similar one-factor studies have been advocated for early-stage tuning (Kingma & Ba, 2015; LeCun, Bottou, Bengio, & Haffner, 1998).

2 Experimental design

2.1 Configuration list

Table 1: Evaluated configurations (one change at a time)

Run	Filters	Kernel	Dense	Batch	Optimiser
baseline	32	3	512	64	adam
filters_256	256	3	512	64	adam
dense_units_32	32	3	32	64	adam
batch_size_16	32	3	512	16	adam
optimizer_sgd	32	3	512	64	sgd
kernel_size_6	32	6	512	64	adam

3 Results

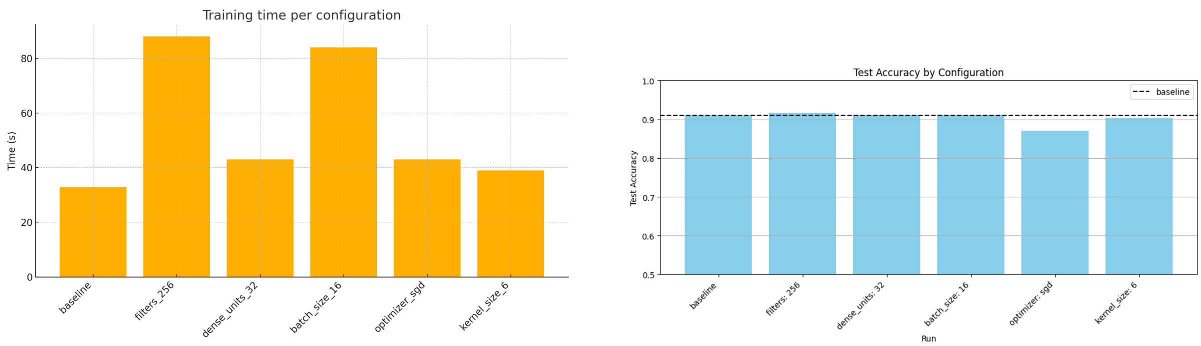
3.1 Accuracy and runtime

Table 2: Performance summary

Run	Val. Acc. %	Test Acc. %	Epochs	Time (s)
baseline	91.63	90.95	3	33
filters_256	91.54	91.54	4	88
dense_units_32	91.10	91.10	10	43
batch_size_16	91.33	91.21	3	84
optimizer_sgd	87.08	87.08	10	43
kernel_size_6	91.17	90.37	5	39

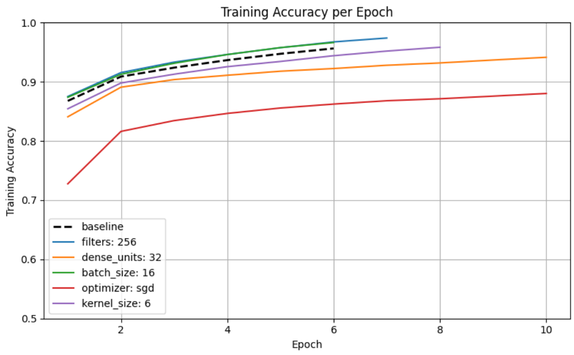
Table 3: Incremental effect of each single change relative to the baseline

Run	ΔVal. Acc. (pp)	ΔTime (s)
filters_256	−0.09	+55
dense_units_32	−0.53	+10
batch_size_16	−0.30	+51
optimizer_sgd	−4.55	+10
kernel_size_6	−0.46	+6

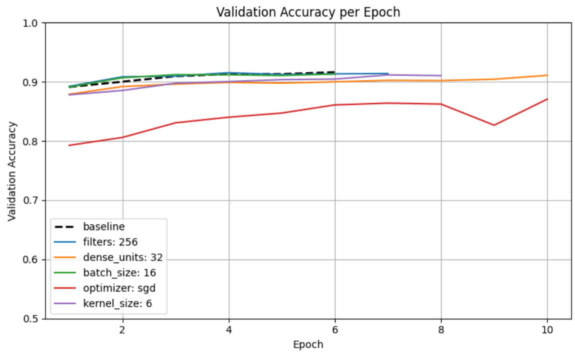


(a) **Runtime**   (b) **Test accuracy**

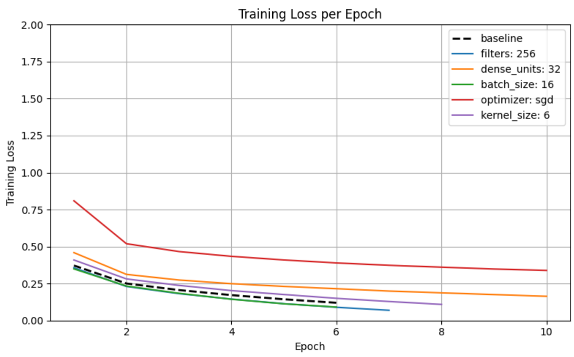
Figure 5: Speed/accuracy trade-off for the six configurations.



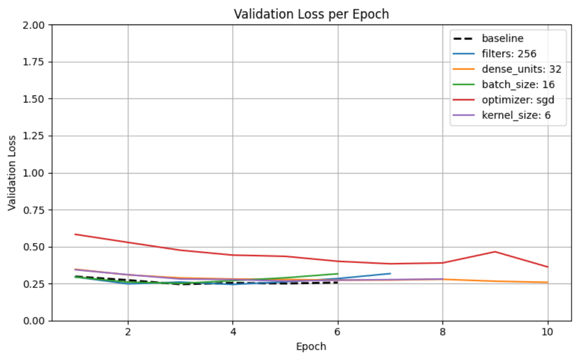
(a) **Training accuracy**



(b) **Validation accuracy**



(c) **Training loss**



(d) **Validation loss**

Figure 6: Learning curves for the baseline CNN.

4 Discussion

Table 3 shows that optimiser and filter count delivered the largest shifts, whereas kernel size, dense width and batch size had secondary effects:

- **Filter count** – Doubling filters slowed training without a proportional accuracy gain.
- **Kernel size** – Larger kernels give diminishing returns beyond 3×3.
- **Dense width** – Shrinking dense units speeds inference but requires more epochs to converge.
- **Optimiser** – Adam dominates SGD for both speed and accuracy.
- **Batch size** – Very small batches add runtime with no payoff.

Several Optuna trials were needed to find a suitable baseline. Although a few distinct baselines appeared during the search, the configuration reported here surfaced most frequently and behaved the most stably. In the rarer cases where a different baseline emerged, it was either slower to train or showed inconsistent responses when individual hyper-parameters were altered.

5 Conclusion

The Optuna seed → one-at-a-time protocol yielded interpretable deltas under tight runtime constraints however, there were inconsistencies when other local minimums resulted. For Fashion-MNIST on this compact CNN, optimiser choice and filter count are the primary levers. Future work should rank variables by global importance (e.g., SHAP) and refine filter granularity.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *25th ACM sigkdd international conference on knowledge discovery and data mining* (pp. 2623–2631). ACM.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations.*

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. doi: 10.1109/5.726791