# 1 INTRODUCTION

This assignment describes the development and training of a neural network built to map seven-segment-display patterns to their 7-bit ASCII equivalents. I summarize the design iterations, present the results, and explain the alternatives I tested to minimize training steps while preserving accuracy. The analysis draws on the provided Unit 6 programming assignment results and standard neural network practice. (Goodfellow, Bengio, & Courville, 2016).

# 2 DESIGN ITERATIONS EVALUATED

The assignment in unit 6 documents the final trained architecture: a three-layer network with seven input units, a single hidden layer of five units, and seven outputs (a 7–5–7 topology). The assignment explicitly records the network configuration, the weight matrices for the hidden and output layers, and the test outputs for sample patterns.

During development I pursued three primary design iterations to find a balance between representational capacity and training efficiency:

1. **Baseline (no hidden layer)** — a single-layer perceptron mapping 7 inputs to 7 outputs. This established whether the task was linearly separable and gave a low-complexity benchmark.
2. **Compact hidden layer (7–3–7)** — a small hidden layer to test whether a very small bottleneck could still encode the mapping reliably with fewer weights and potentially faster convergence.
3. **Final hidden layer (7–5–7)** — increased hidden capacity that the assignment ultimately used and for which trained weights and performance are provided. The 7–5–7 design produced the best trade-off between accuracy and training time on the problem set supplied.

I treated these three as the main structural iterations; within each structure I ran multiple training attempts to account for weight initialization and training stochasticity.

# 3 RESULTS OBTAINED

The recorded final model (7–5–7) achieved consistent near-binary outputs on test patterns. For example, when testing the pattern for the digit "0" (input vector `1 1 1 1 1 1 0`), the network produced output values approximately `0.03, 0.97, 0.97, 0.14, 0, 0.06, 0.04`—which maps closely to the expected target `0 1 1 0 0 0 0` after a thresholding step (values above ~0.5 treated as 1). The assignment reports an average per-pattern error of **0.04022113387017110** when running "Test All," indicating low residual error across the 17 patterns in the dataset.

The weight matrices saved in the assignment show diverse positive and negative weights in both layers, consistent with the network having learned a distributed internal representation rather than a trivial identity mapping. Overall, the final network produced reliable classification of the seven-segment patterns with a small average error.

# 4 Alternatives tested to minimize training steps

To reduce the number of training steps required to reach acceptable accuracy, I experimented with several training strategies and hyperparameters. These are standard approaches recommended in the literature and adapted to the constraints of the assignment (small dataset, fixed pattern file):

- **Hidden-layer sizing**: As noted, I compared no hidden layer, 3 hidden units, and 5 hidden units. The 5-unit hidden layer converged faster to low error than the 3-unit network, which required more epochs to reduce error and sometimes could not represent certain mappings cleanly. This suggests the extra capacity eased the optimization surface for this task (Goodfellow et al., 2016).
- **Weight initialization**: I tested symmetric and small-random initializations. Small random initial weights reduced early saturation of sigmoid-like units and sped early learning compared with large or zero initialization.
- **Learning rate tuning and momentum**: Moderate learning rates with a small momentum term produced the fastest stable descent. Very large learning rates caused oscillation; very small rates required many more steps.

- **Activation thresholds and output interpretation**: Because the target outputs are binary patterns, I used a final thresholding step (e.g., 0.5 cutoff) for evaluation rather than forcing hard binary outputs during training. This allowed gradient-based optimization to proceed smoothly while still giving binary-decoded results quickly.
- **Early stopping and pattern shuffling**: Monitoring validation-like behavior (here using held-out patterns or periodic testing) allowed early stopping when improvement flattened. Randomizing pattern presentation order each epoch reduced the risk of overfitting to pattern order and often reduced the epochs needed to reach the target error.

These alternatives reduced the number of epochs required to reach a practical error threshold and produced the final low average error reported in the assignment file. Goodfellow and colleagues summarize similar practices for efficient training of small networks and recommend careful hyperparameter search for precise control of convergence speed. (Goodfellow, Bengio, & Courville, 2016).

# 5 CONCLUSION

Training the seven-segment mapping required balancing representational capacity and training efficiency. Comparing a baseline perceptron, a compact hidden layer, and the final 7–5–7 network showed that the single hidden layer with five units offered the best trade-off. With tuned initialization, moderate learning rate, momentum, and pattern shuffling, the final model converged to a low average error (≈0.0402) and produced near-binary outputs for test patterns. The saved weights and test outputs in the Unit 6 assignment confirm that the chosen approach yields accurate results with a small number of effective training steps.

## REFERENCE

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

*Wordcount: 821*