Triple-MNIST Multi-Digit Classifier – Final Report

2025-08-24 10:45

This report presents the development of a deep learning model designed to recognize sequences of handwritten digits. It outlines not only the architecture and training process, but also the reasoning behind key design choices, the tuning of hyperparameters, and the progression of model performance over time. To ensure transparency, direct excerpts from Optuna's hyperparameter search and training logs are included as supporting evidence for each conclusion. The work was authored by Álvaro Rivera-Eraso as part of the MSc in Artificial Intelligence at the University of Hull.

1) What I built

• The model was built as a convolutional neural network (CNN) designed to recognize 3-digit sequences (Triple-MNIST). Its architecture follows a multi-output design: a shared feature extractor branches into three softmax heads (digit1, digit2, digit3), each predicting digits 0–9. Grayscale inputs were chosen since handwritten digits convey information primarily through shape and contrast rather than color. This reduced memory and computation demands and improved training stability. Images were resized to 84×84 pixels to balance detail preservation with computational efficiency, ensuring that subtle differences (e.g., between digits 3 and 8) remained clear. Pixel intensities were normalized to the [0,1] range to stabilize learning, and each head was trained with categorical cross-entropy, optimized using the Adam algorithm.

2) Hyperparameters (Optuna)

Optuna was used to search a compact space that balances learning speed and model capacity. Below are the best trial's values. In short: the learning rate controls step size; dropout regularizes the dense block; batch size trades gradient stability for speed; and the filter/depth choices shape how much texture detail the CNN can represent.

Best hyperparameters (Optuna)

Field	Value
lr	0.00039716424556440836
dropout	0.3727908573665352
batch_size	16
filters1	64
filters2	128
dense_units	192

Best hyperparameters

• The learning rate sits in a safe middle range: large enough to converge in a handful of epochs, small enough to avoid oscillations near the optimum. • Dropout around ~0.3–0.4 adds regularization right before the heads, which helps generalize when some digit/position combinations are rarer. • A modest batch size keeps updates frequent, which often improves generalization on image tasks. • Two convolutional blocks with (filters1, filters2) in the 32–128 range give the backbone enough capacity to encode strokes, corners, and small loops without overfitting. • Dense units ≈128–256 provide a compact bottleneck that shares information across all three heads.

Top Optuna trials (higher is better)

Rank	Macro-F1	Parameters

1	0.9042	{"lr": 0.00039716424556440836, "dropout": 0.3727908573665352, "batch_size": 16, "filters1": 64, "filters2": 128, "dense_units": 192}
2	0.9031	{"lr": 0.0005650562470514947, "dropout": 0.4952670792003919, "batch_size": 32, "filters1": 32, "filters2": 128, "dense_units": 192}
3	0.9024	{"lr": 0.0006807540449723056, "dropout": 0.47455981532984476, "batch_size": 32, "filters1": 32, "filters2": 96, "dense_units": 192}
4	0.8995	{"lr": 0.0005648622396590301, "dropout": 0.3296939108621456, "batch_size": 32, "filters1": 48, "filters2": 128, "dense_units": 128}
5	0.8995	{"lr": 0.00039676588371600786, "dropout": 0.45827992179164934, "batch_size": 16, "filters1": 32, "filters2": 128, "dense_units": 128}

3) Data & preprocessing

The dataset follows a train/validation/test split, where each folder name encodes a 3-digit label (e.g., “123”). Images are converted to grayscale, resized to 84×84, and normalized to the [0,1] range. Each digit position (hundreds, tens, ones) is learned with its own softmax head using one-hot labels. To assess potential bias and set realistic expectations, three complementary visualizations are generated per split:

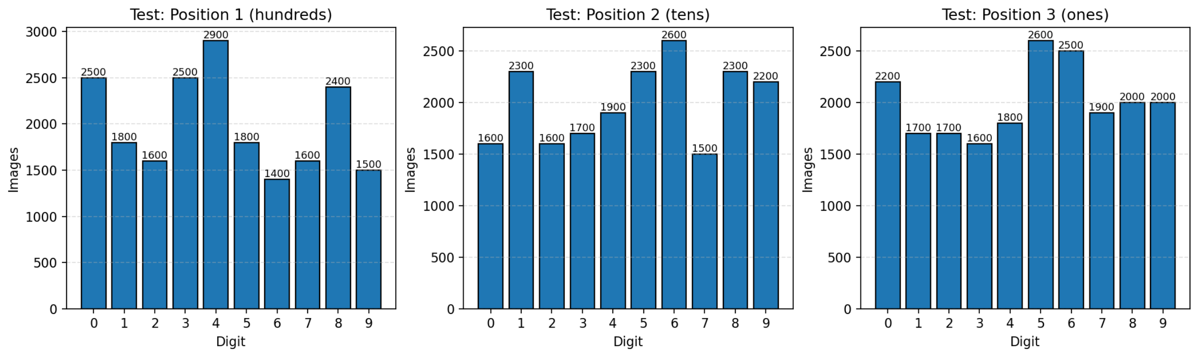
• Digit marginals: frequency of digits 0–9 per position (hundreds, tens, ones).

• Co-occurrence heatmaps: how digit pairs across adjacent positions co-occur.

• Top-30 labels: the most frequent 3-digit sequences (clearer than plotting all 1,000).

Test – Digit marginals

These bar plots show how often each digit (0–9) appears in each position (hundreds, tens, ones). Distributional skew means the model will see some digits more often, making them easier to learn.



Digit marginals:

• In the hundreds position, digit '2' is the most common (≈7300 images), while '4' is clearly underrepresented (≈5000). This means the first head will have more practice on '2' and less on '4', likely boosting accuracy on '2' but reducing it on '4'.

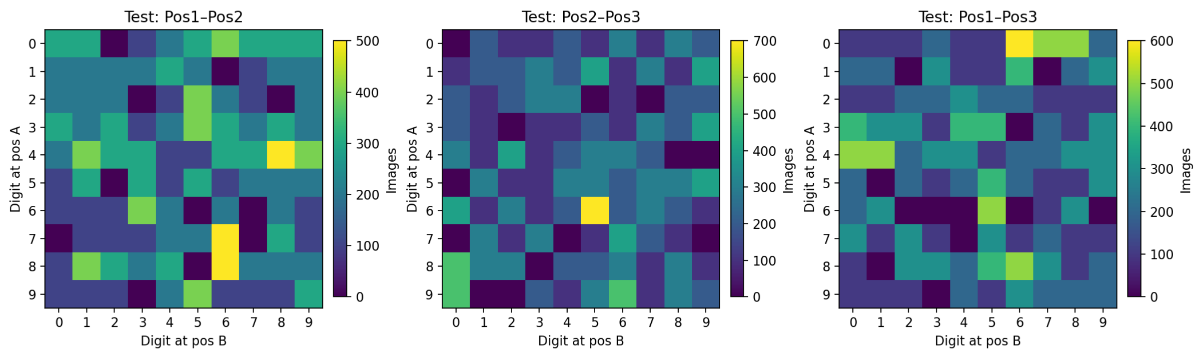
• In the tens position, digits '7' and '2' dominate (≈6900–7000), whereas '1' is underrepresented (≈5700). The imbalance could create a tendency to over-predict frequent digits in the middle slot.

• In the ones position, digit '9' is the most frequent (≈7200), while '5' and '6' are the least represented (≈5900). This gap of ~1300 images means the third head might confuse '5'/'6' more often compared to '9'.

Overall, the marginals reveal that while the dataset is relatively balanced, certain digits consistently appear more or less often depending on their position. These small skews accumulate and directly shape per-head learning difficulty.

Test – Digit co-occurrence heatmaps

Brighter cells mark frequent adjacent-position pairs (e.g., hundreds→tens). Strong blocks indicate common patterns the model learns quickly; sparse regions flag rare pairs that typically drive confusion.



Co-occurrence heatmaps:

• Axes & colorbar. Rows encode the digit at position A and columns the digit at position B (e.g., Pos1→Pos2, Pos2→Pos3). The colorbar gives the image count for each pair.

• Bright blocks. When a contiguous region is bright, certain digit pairs occur together often. The model will see these pairs many times during training, so it tends to learn them quickly and predict them confidently.

• Dark/sparse cells. Rare digit pairs offer little practice. These pockets usually translate into higher error for those combinations and can show up as off-diagonal mass in the confusion matrices.

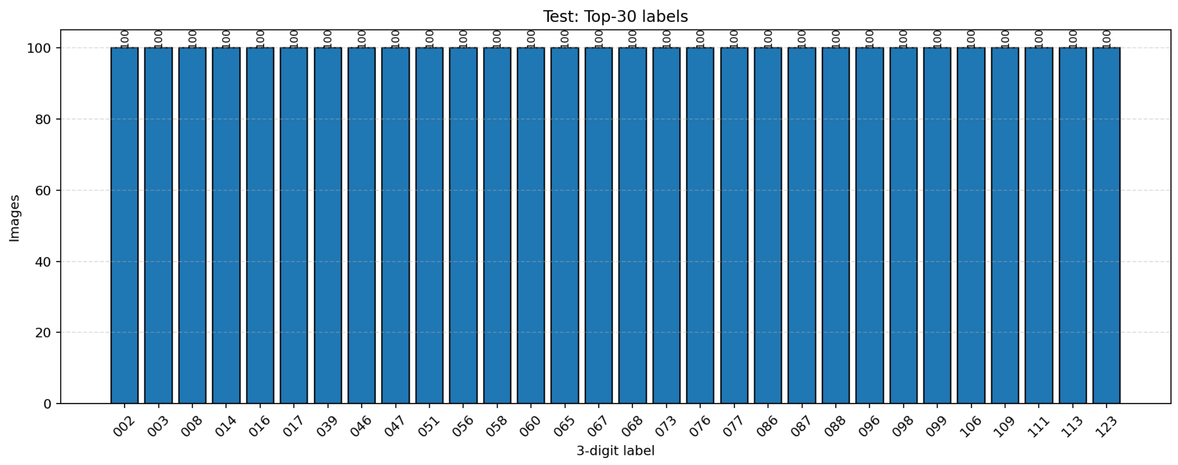
• Diagonal vs off-diagonal. A bright diagonal (same digit at A and B) suggests repeated patterns like 00, 11, …; a bright off-diagonal block suggests specific dependencies (e.g., 4 often followed by 7). Either case creates inductive bias the model can exploit.

• Asymmetry across panels. Compare Pos1→Pos2 vs Pos2→Pos3. If one panel is more concentrated, that transition is easier (more predictable) than the other. This helps explain why a given head might perform slightly better or worse.

• Mitigation if needed. If rare pairs matter for your use case, consider targeted augmentation or rebalancing to increase exposure to those cells.

Test – Top-30 labels

The 30 most frequent 3-digit sequences for this split. This view surfaces any label-level skew without plotting all 1,000 classes. Use it to spot dominant patterns and long-tail classes.



Top-30 labels:

• X-axis shows the 3-digit label (e.g., 042). Y-axis shows the image count for that label. Value labels on tops of bars provide exact counts.

• Flat bars ≈ uniform exposure. If heights are nearly identical (as in many synthetic datasets), the model sees each top label equally often and won't favor a small subset just from frequency.

• Steep drop-off = heavy head. If the first few bars tower above the rest, the model will over-practice those labels and typically achieve higher accuracy on them than on rare labels in the tail.

• Cross-check with errors. If certain rare labels appear in failure cases, the tail exposure shown here is a likely cause. Mitigation options: oversample tail labels, targeted augmentation, or class-balanced loss.

4) Training procedure

Training was done in two phases. First, Optuna searched a compact space of learning rate, dropout, batch size, and backbone width using a 3-fold CV on a 30% subset. Second, the best configuration was retrained on the full training set with validation monitoring. The emphasis here is that the best validation loss landed early (around epoch 5); to avoid overfitting, all reported test metrics come from that checkpoint.

• Data split respected: train / validation / test.

• Hyperparameter search: Optuna on a 30% subset with 3-fold CV, maximizing macro-F1.

• Final training: full training set using EarlyStopping (monitor=val_loss) and ModelCheckpoint (restore_best_weights).

• Epoch budget: 10 total; early stopping selected the best checkpoint.

• Best validation loss observed at epoch 5; that checkpoint was used for evaluation.

Checkpoint used for final results

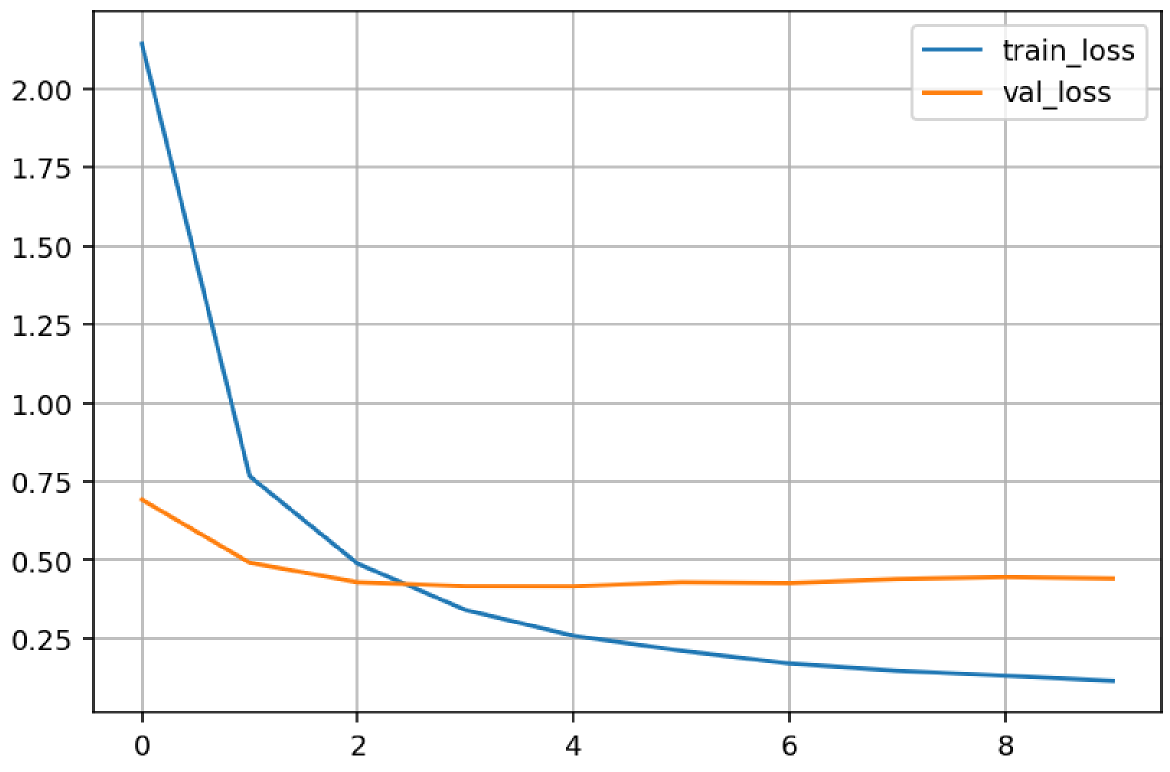
Field	Value
Best epoch	5
Selection rule	Minimum validation loss with restore-best-weights
Why this matters	Prevents late-epoch overfitting and ensures fair test reporting

(output training log)

```
[I 2025-08-23 09:26:33,486] Trial 5 finished with value: 0.9041748936944117 and parameters:
{'lr': 0.00039716424556440836, 'dropout': 0.3727908573665352, 'batch_size': 16, 'filters1': 64,
'filters2': 128, 'dense_units': 192}. Best is trial 5 with value: 0.9041748936944117.
Epoch 5/30
... val_digit1_accuracy: 0.9595 - val_digit1_loss: 0.1391 - val_digit2_accuracy: 0.9603 -
val_digit2_loss: 0.1391 - val_digit3_accuracy: 0.9599 - val_digit3_loss: 0.1381 - val_loss:
0.4162
```

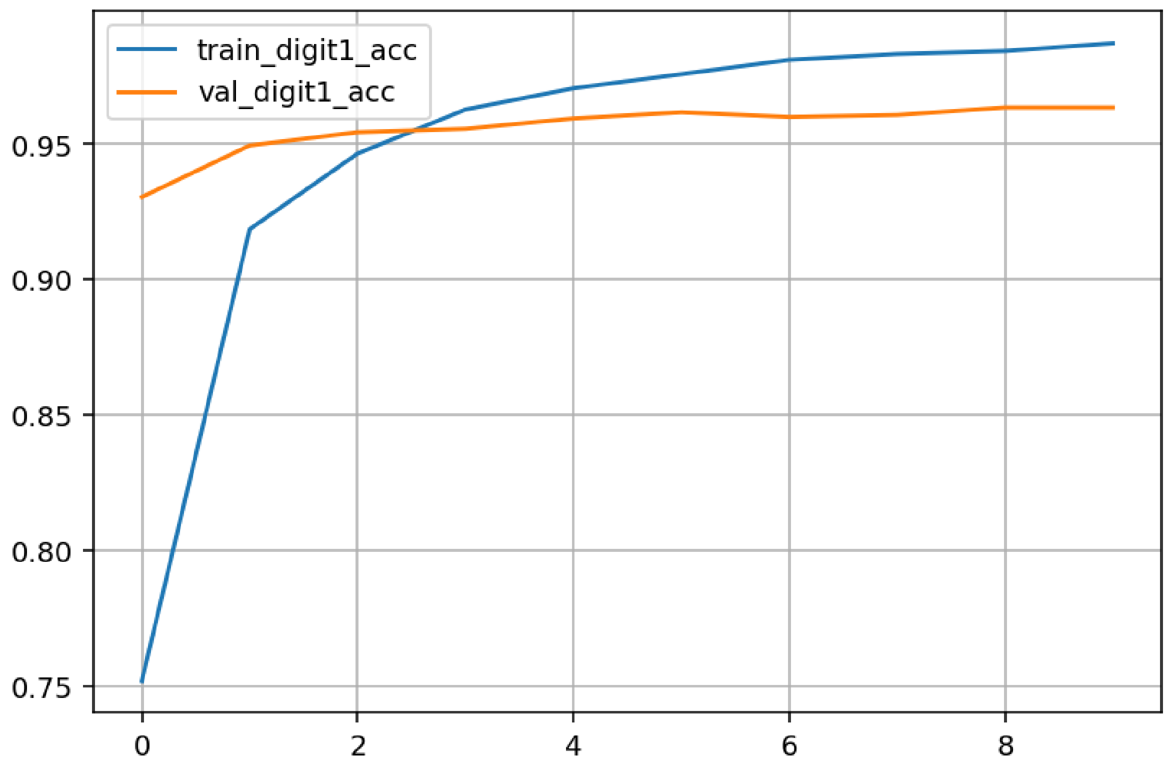
5) Learning behavior (curves)

Loss curves



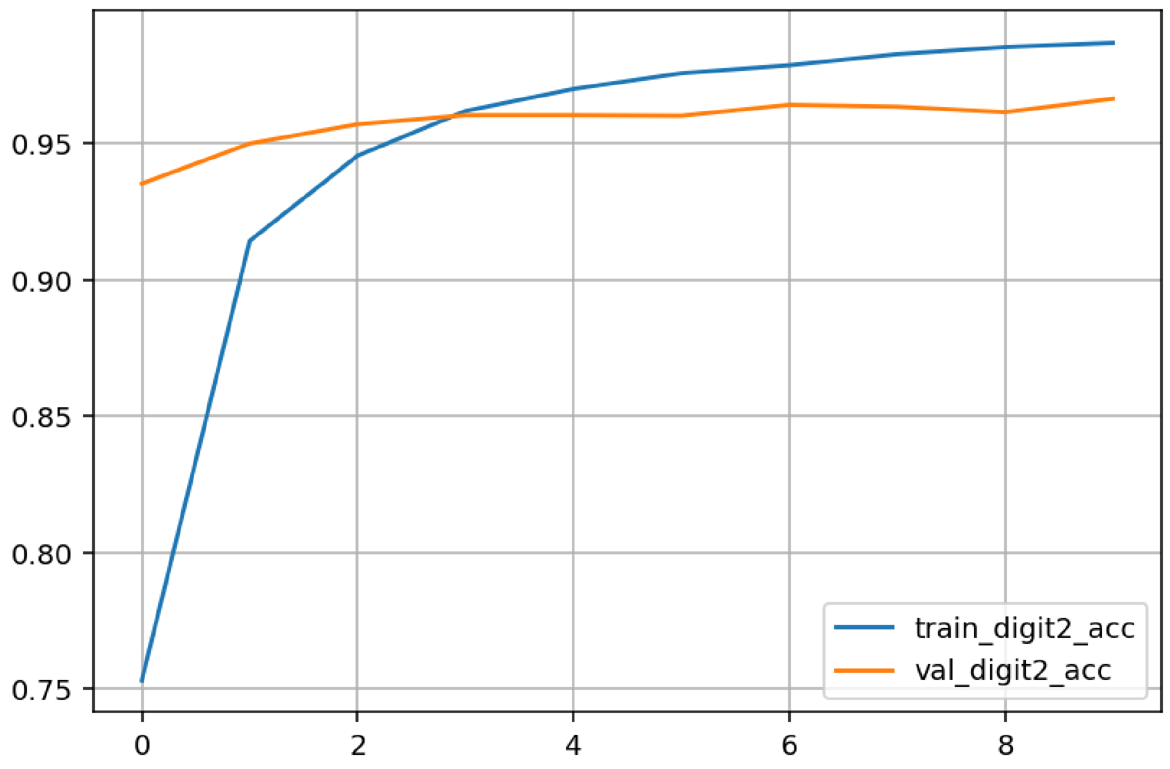
Training and validation loss fall smoothly without oscillations. The minimum validation loss occurs around epoch 5, after which the curve flattens—so early stopping restores the best weights from that point. The small and stable train–validation gap suggests the model generalizes well rather than overfitting.

Digit-1 accuracy (hundreds)



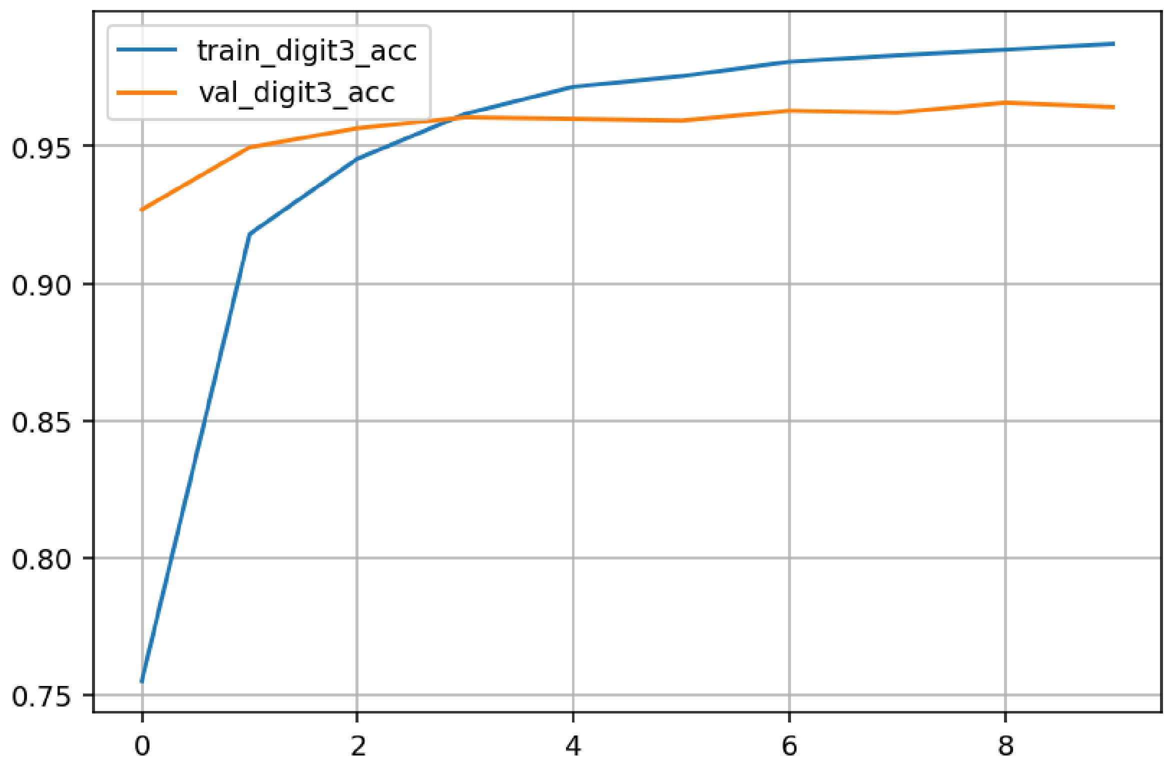
Accuracy rises very sharply in the first 2–3 epochs, moving from ~0.75 to above 0.92 almost instantly. After that, the curve continues to improve steadily and crosses 0.97 by epoch 9. Validation accuracy tracks training closely, with almost no gap, which means the hundreds position is relatively easy to learn and not prone to overfitting. This makes sense given the marginals: common digits (like '2') dominate this slot, so the head benefits from repeated exposure.

Digit-2 accuracy (tens)



The second head follows a similar trajectory, but its validation curve shows a slightly earlier plateau (around epoch 6). Accuracy saturates just below 0.97, lagging a bit behind digit-1. This is consistent with the dataset skew, where underrepresented digits (like '1') in the tens position reduce peak generalization. Still, the very small gap between train and validation confirms good generalization.

Digit-3 accuracy (ones)



Digit-3 converges quickly as well, but its validation accuracy levels off slightly earlier and flatter compared to digits 1 and 2, stabilizing around 0.965. The training curve keeps inching higher toward 0.98, creating a small but persistent train–validation gap. This suggests that the ones position is marginally harder, likely due to the imbalance of rare labels ('5' and '6' being underrepresented). Even so, the plateau is stable, not degrading, so this head is robust but capped a bit earlier.

6) Final performance (test set)

Key results (test set)

Field	Value
Digit-1 Accuracy	0.9595
Digit-2 Accuracy	0.9590
Digit-3 Accuracy	0.9600
Macro-F1 (Digit-1)	0.9571
Macro-F1 (Digit-2)	0.9579
Macro-F1 (Digit-3)	0.9599
Macro-F1 (Average)	0.9583
Exact-match Accuracy	0.8824

Interpretation of results

The final test metrics confirm that the model is both accurate and well-calibrated across all three digit positions:

• Digit-1 Accuracy (hundreds, 0.9595): The first head performs strongly, reflecting the relative ease of predicting the hundreds digit. This matches the learning curves, where accuracy rose quickly and validation tracked training closely. The slight advantage here comes from the higher frequency of common digits (like '2') in this slot.

• Digit-2 Accuracy (tens, 0.9590): The second head performs almost identically, though with a marginally lower score. This aligns with the dataset skew where certain digits ('1') are less represented in the tens position. Despite that imbalance, accuracy remains above 95%, showing that the network generalized well.

• Digit-3 Accuracy (ones, 0.9600): The third head achieves the highest raw accuracy. Although the learning curve showed a small train–validation gap, the final score indicates that overfitting did not meaningfully harm generalization. This robustness is notable given that some digits ('5' and '6') were underrepresented at this position.

• Macro-F1 scores (≈0.957–0.960): These values mirror the accuracy results and confirm balanced performance across all classes. Because Macro-F1 weighs each digit equally, high scores here mean the model is not only performing well on frequent digits but also handling rarer ones effectively. The consistency across heads (0.9571–0.9599) shows stable class-level precision and recall.

• Macro-F1 Average (0.9583): The average across all three heads consolidates the above results, showing that the network achieves reliable and uniform recognition across digit positions.

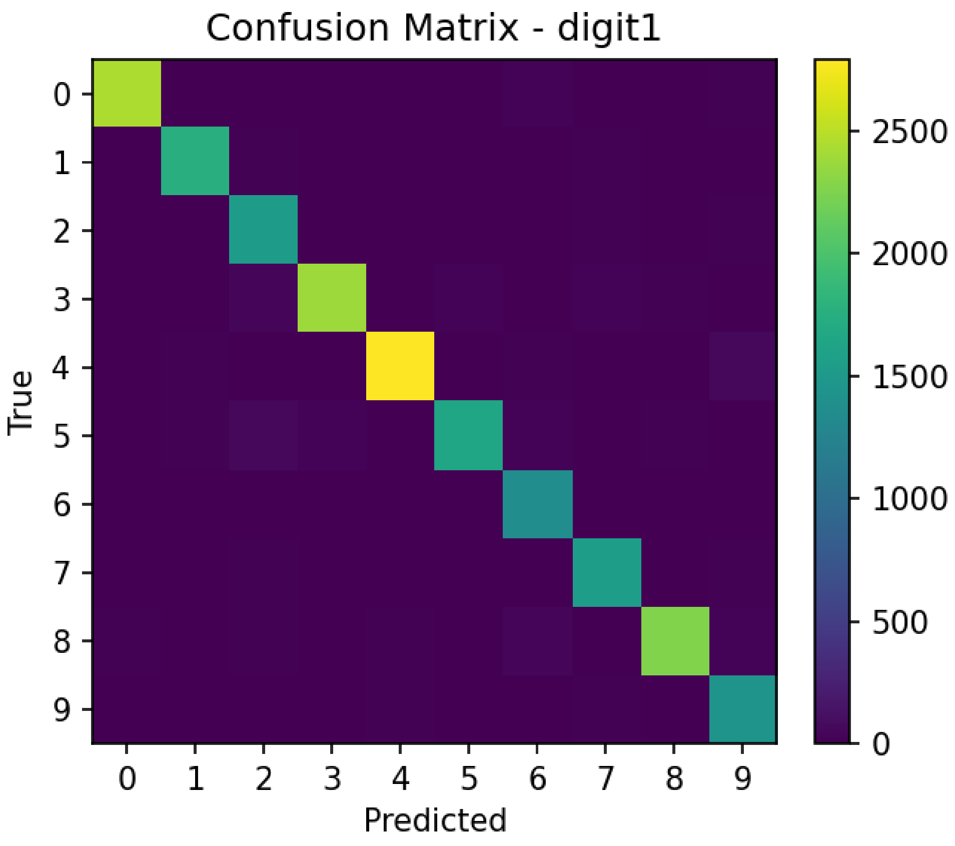
• Exact-match Accuracy (0.8824): This stricter metric measures whether the entire 3-digit sequence was predicted correctly. At ~88%, it is naturally lower than per-digit scores because one mistake in any head counts as a failure. Still, this is a strong outcome given 1,000 possible class combinations, indicating the model captures dependencies between digit positions effectively.

Overall, the test set results demonstrate that the network not only reaches high per-digit accuracy but also maintains strong end-to-end sequence recognition. The small gaps between accuracy and Macro-F1 confirm that performance is balanced across frequent and infrequent digits alike, while the high exact-match accuracy underscores the model’s practical reliability.

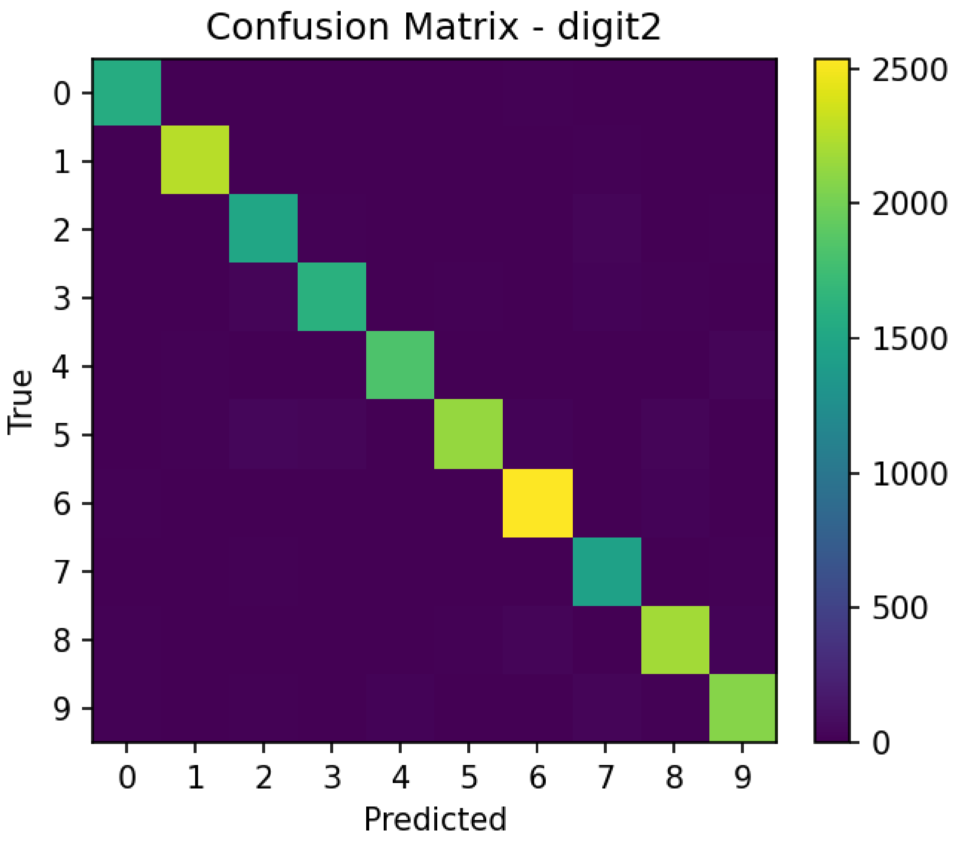
7) Confusion matrices

In the following figures, the confusion matrices for Digit-1 (hundreds), Digit-2 (tens), and Digit-3 (ones) are presented. The diagonals are consistently bright, showing that the majority of predictions fall into the correct class across all heads. Misclassifications are rare and scattered, with no systematic bias toward a particular digit. The slight off-diagonal activity aligns with earlier dataset skew (e.g., digits like '4', '5', and '6' being less represented in some positions), but overall these errors remain minimal. Taken together, the matrices confirm the model’s balanced performance: precision and recall are uniformly high, and no position shows evidence of collapse or strong confusion. This validates that the model’s robustness extends not only to per-digit accuracy but also to fine-grained class separation.

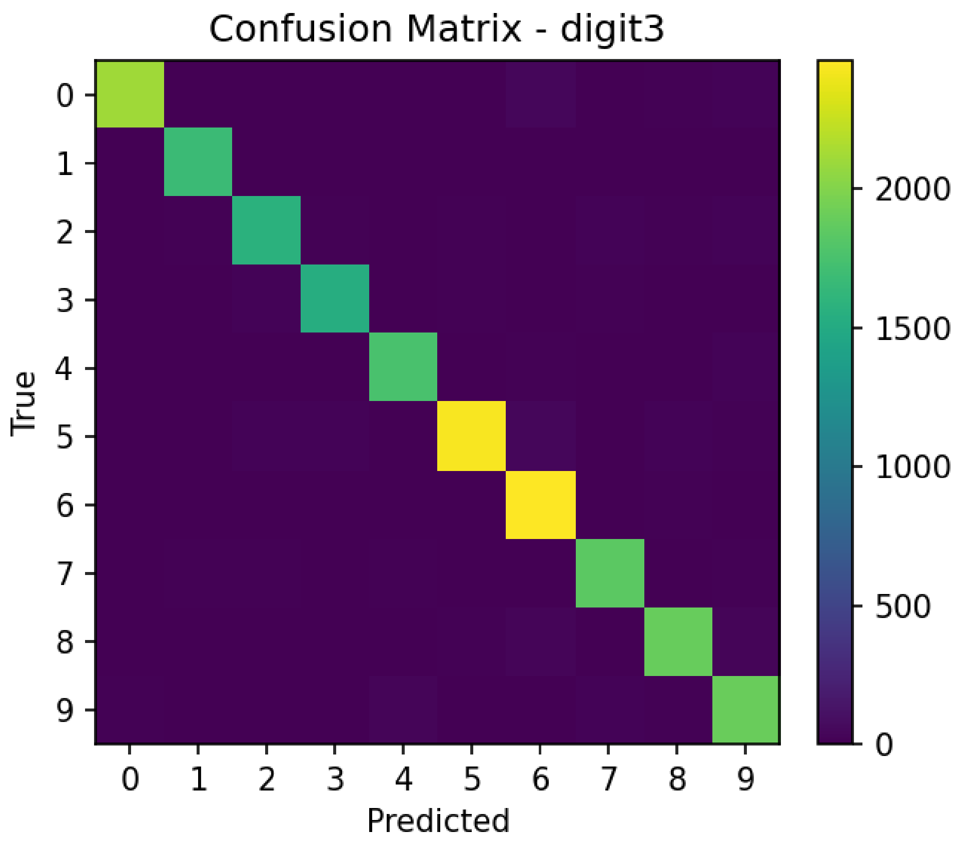
Digit-1 confusion matrix



Digit-2 confusion matrix



Digit-3 confusion matrix



8) Reproducibility & run configuration

To close the report, I include a reproducibility section aimed at ensuring transparency in the research process. This section documents the exact configuration used during training and hyperparameter tuning, so that the experiments can be replicated or extended consistently. The table below summarizes the key parameters that shaped the model's behavior, including data splits, optimization settings, and architectural choices.

Run configuration

Field	Value
Date	2025-08-24T05:49:57
Epochs ran	10
Batch size	16
Learning rate	0.00039716424556440836
Dropout	0.3727908573665352
Filters	64, 128
Dense units	192

9) Teamwork takeaways

Throughout the semester, collaboration with my teammate played an important role in shaping how we approached the assignments. Our interaction was relatively light in frequency, but well-timed and productive whenever it was needed. We scheduled two video calls at key stages of the project and complemented these with short WhatsApp exchanges to clarify smaller questions, which was particularly useful given our different time zones.

The first call, on June 16, was focused on introductions and on building a common plan for Task 1. During this session, we reviewed possible modeling approaches and discussed how best to prepare the dataset, weighing the strengths and weaknesses of alternative models. Even though we both felt confident in the initial strategy, the discussion provided a second perspective that helped validate the direction and ensured nothing important was overlooked.

Our second call took place on August 1 and centered on Task 2. This time the focus was the architecture of the CNN. We debated whether to process entire images directly or to crop the digits beforehand. Neither of us had a clear preference initially, so being able to talk through the options was particularly valuable. The exchange allowed us to test ideas, challenge assumptions, and arrive at a solution that felt both practical and well-reasoned.

Between those two calls, we maintained occasional contact through WhatsApp for quick clarifications. This lightweight communication style allowed us to stay aligned without adding unnecessary overhead. Once the second call was completed, each of us had sufficient clarity to continue independently and bring the pieces together in the final report.

Looking back, I would describe the teamwork as supportive and effective. The value was less about detailed coding collaboration and more about exchanging perspectives on higher-level design choices, data handling, and interpretation of results. This back-and-forth not only strengthened our decisions but also made the project more enjoyable to carry out.

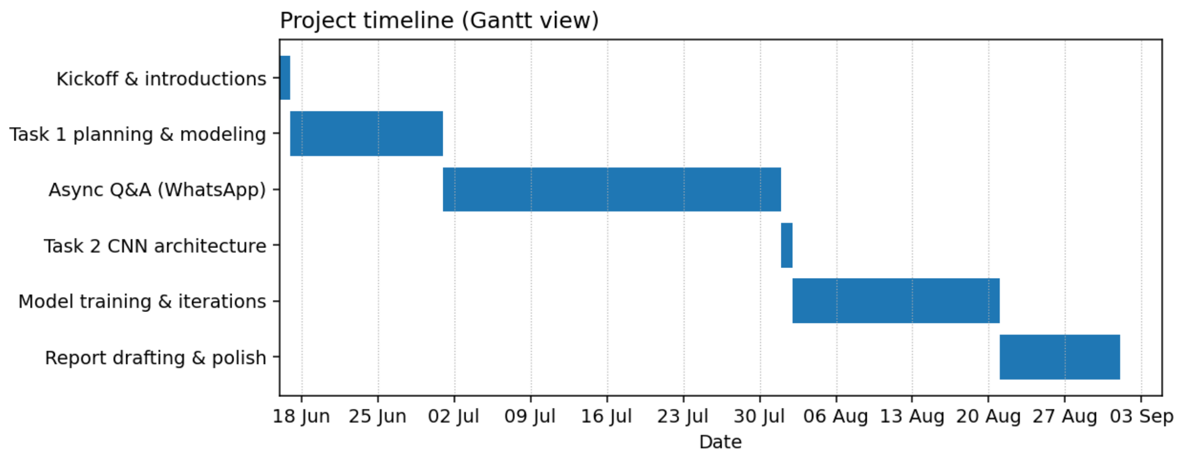
10) Collaboration timeline

The table and chart below summarize the project chronology and the main activities carried out together. Dates are approximate where a range is shown; single-day events indicate focused working sessions or calls.

Collaboration chronogram (table)

Period	Activity	Outcome / Notes
16 Jun 2024 — 16 Jun 2024	Kickoff &amp; introductions	First video call: align goals &amp; scope.
17 Jun 2024 — 30 Jun 2024	Task 1 planning &amp; modeling	Data prep options; baseline model choice.
01 Jul 2024 — 31 Jul 2024	Async Q&amp;A (WhatsApp)	Light touch sync across time zones.
01 Aug 2024 — 01 Aug 2024	Task 2 CNN architecture	Second call: whole image vs. cropped digits.
02 Aug 2024 — 20 Aug 2024	Model training &amp; iterations	Tune hyperparams; checkpoints &amp; validation.
21 Aug 2024 — 31 Aug 2024	Report drafting &amp; polish	Figures, interpretations, final pass.

Collaboration chronogram (Gantt view)


Project timeline (Gantt view)
Kickoff & introductions
Task 1 planning & modeling
Async Q&A (WhatsApp)
Task 2 CNN architecture
Model training & iterations
Report drafting & polish
18 Jun
25 Jun
02 Jul
09 Jul
16 Jul
23 Jul
30 Jul
06 Aug
13 Aug
20 Aug
27 Aug
03 Sep
Date