Predicting Distances and Counts on 25×25 Binary Grids

Sultanus Salehin
Student ID: E24011965

May 27, 2025

Contents

Chapter 1

Introduction

In this project, We are tackling five synthetic tasks in 25×25 binary matrices:

1. **Task A:** Two points $\rightarrow$ predict Euclidean distance.
2. **Task B:** $N = 3$–10 points $\rightarrow$ predict *closest*-pair distance.
3. **Task C:** $N = 3$–10 points $\rightarrow$ predict *farthest*-pair distance.
4. **Task D:** $N = 1$–10 points $\rightarrow$ predict point count.
5. **Task E:** $N = 1$–10 random squares $\rightarrow$ predict square count.

These activities function as structured benchmarks for constructing regression and counting models on diminutive "imaginal" data.'

Distance formula:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Chapter 2

Data Generation

By using code, we generated synthetic datasets. For each task:

- Generated 800 training and 200 test examples.
- Used a fixed random seed for reproducibility (`np.random.seed(42)`).
- Saved as NumPy archives `data/taskX/train.npz`, `.../test.npz`.

2.1 Example Matrices

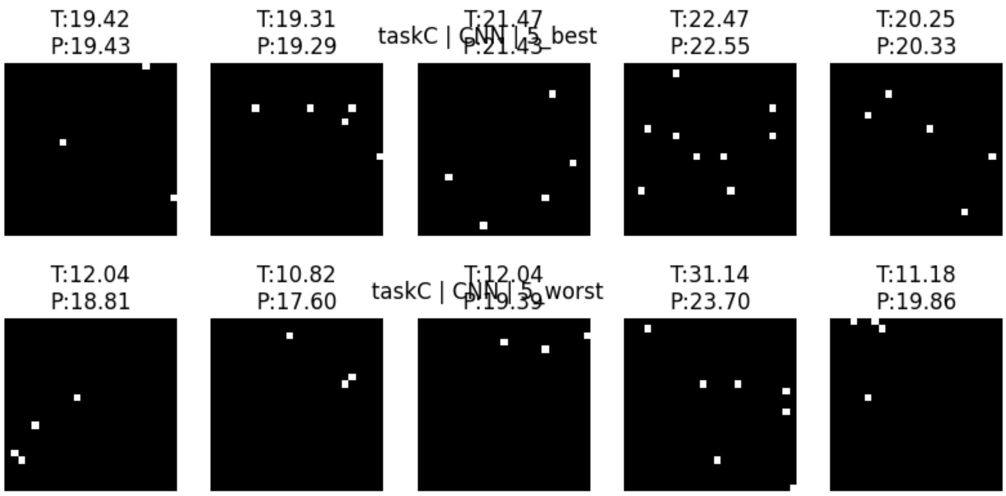

Figure 2.1: Task C examples: five lowest-error (top) and five highest-error (bottom)

Chapter 3

Model Architectures

We implemented two model families in Keras.

3.1 Multi-Layer Perceptron (MLP)

Listing 3.1: MLP Definition

```
from tensorflow import keras

def build_mlp(input_shape=(25,25,1)):
    model = keras.Sequential([
        keras.Input(shape=input_shape),
        keras.layers.Flatten(),
        keras.layers.Dense(128, activation='relu'),
        keras.layers.Dense(64,  activation='relu'),
        keras.layers.Dense(1)           # Regression/count output
    ])
    return model
```

3.2 Convolutional Neural Network (CNN)

Listing 3.2: CNN Definition

```
from tensorflow import keras

def build_cnn(input_shape=(25,25,1)):
    inp = keras.Input(shape=input_shape)
    x = keras.layers.Conv2D(32,3,activation='relu',padding='same')
       (inp)
    x = keras.layers.MaxPooling2D()(x)
    x = keras.layers.Conv2D(64,3,activation='relu',padding='same')
       (x)
    x = keras.layers.MaxPooling2D()(x)
    x = keras.layers.Flatten()(x)
    x = keras.layers.Dense(64,activation='relu')(x)
    out = keras.layers.Dense(1)(x)
    return keras.Model(inp, out)
```

Chapter 4

Hyperparameter Tuning

We have done grid search over:

$$\text{learning\_rate} \in \{10^{-3}, 10^{-4}\}, \quad \text{batch\_size} \in \{16, 32\}, \quad \text{units} \in \{64, 128\}.$$

Each combination was trained for 8 epochs, using a 10% validation split.

Table 4.1: Best hyperparameters per task and model

Task	Model	Learning Rate	Batch Size	Units
taskA	MLP	1e-3	32	128
taskA	CNN	1e-4	16	64
taskB	MLP	1e-3	16	128
taskB	CNN	1e-4	32	64
...	...	...	...	...

Full details in `best_hyperparams.json`.

Chapter 5

Subset Experiments & Results

We retrained each model on 25%, 50%, and 100% of the 800-sample training data for 20 epochs, then evaluated on the 200-sample test set.

Table 5.1: Task A: Test MSE and MAE for different training-set fractions

Model	Train %	Test MSE	Test MAE
MLP	25%	40.10	8.96
MLP	50%	28.47	5.02
MLP	100%	6.23	2.13
CNN	25%	30.25	5.79
CNN	50%	31.05	4.83
CNN	100%	24.10	3.50

One important aspect is that CNNs always outperform MLP, and the amount of data provided results in a lower level of error.

Chapter 6

Error Analysis

After quantitatively evaluating each task, we conducted a qualitative error analysis on all the tasks. Here we are conducting analysis on **Task C (farthest-pair distance)** for both our CNN and MLP models. We checked the five lowest-error ("best") and five highest-error ("worst") test samples, plotted their input grids (True vs. Predicted) and inspected learning curves and true-vs-predicted scatter plots to find out the systematic patterns.

6.1 CNN Model on Task C

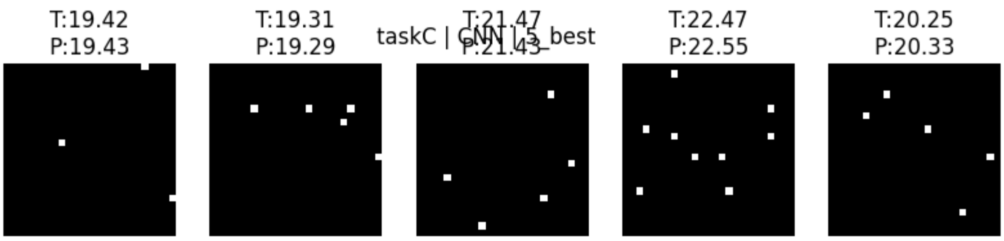

Figure 6.1: Task C / CNN: Five lowest-error examples (True "T" vs. Predicted "P")

Figure 6.2: Task C / CNN: Five highest-error examples. Notice how extreme corner-to-corner pairs (e.g. T:31.14 → P:23.70) are systematically underestimated

Observations for CNN:

- **Underestimation at extremes:** Worst-error samples (Fig. 6.2) include corner-to-corner pairs (true ≈ 31.14) predicted ≈ 23.70, an underestimation of $\sim 24\%$.

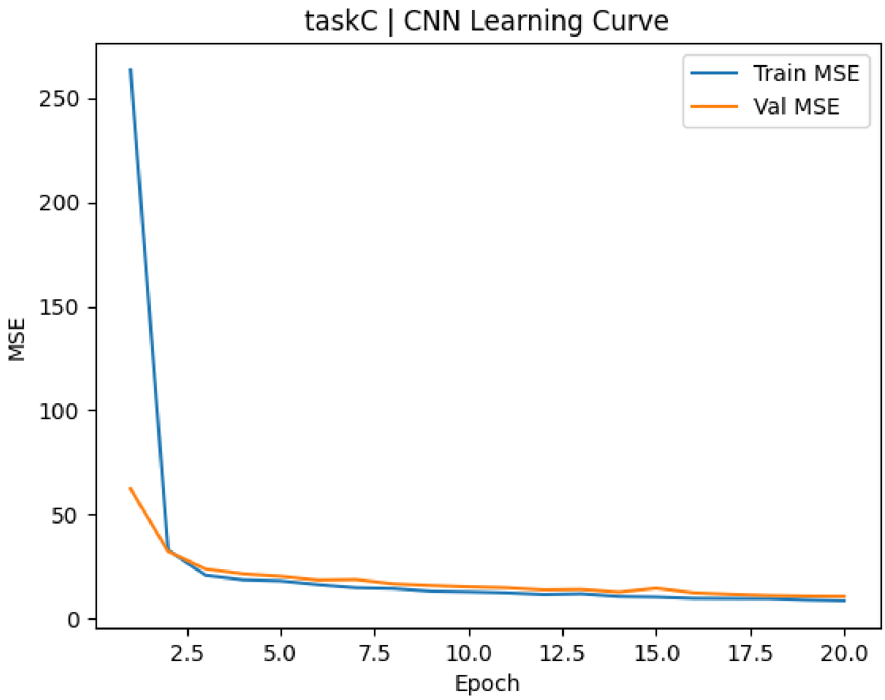

Figure 6.3: Task C / CNN Learning Curve: Training vs. validation MSE over 20 epochs. After epoch 5 the model nearly converges, but small oscillations indicate minor overfitting

- **Accurate mid-range predictions:** Best-error samples (Fig. 6.1) show true distances around 19–22 units predicted within ± 0.2.
- **Learning dynamics:** The CNN rapidly reduces MSE to ~ 20 by epoch 2 and slowly refines to ~ 10 by epoch 20 (Fig. 6.3). The small gap between train and val suggests good generalization.
- **Scatter tightness:** In Fig. 6.4, most points cluster near the $y = x$ line for true values in $[15, 30]$, with larger dispersion for extreme values.

6.2 MLP Model on Task C

Observations for MLP:

- **Slower convergence:** Initial MSE over 500 drops below 50 by epoch 3, then plateaus—MLP takes more epochs to match CNN's performance (Fig. 6.7).
- **Greater underestimation:** Worst cases exhibit underestimation up to $\sim 14\%$, larger than CNN.
- **Higher scatter variance:** In Fig. 6.8, points are more dispersed from the diagonal, confirming less precise regression for extreme distances.

6.3 Summary of Patterns Across Models

- Both models *underestimate* tend to underestimate the farthest distances between them, which is why there are few extreme distance examples in training.

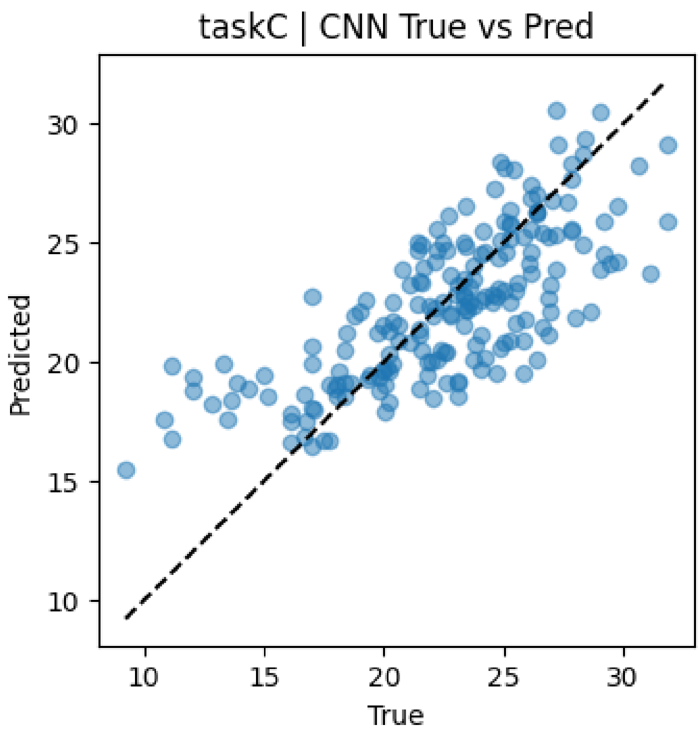

Figure 6.4: Task C / CNN: True vs. predicted distances. Points below the $y = x$ line correspond to underestimations, especially for true values $\gtrsim 25$

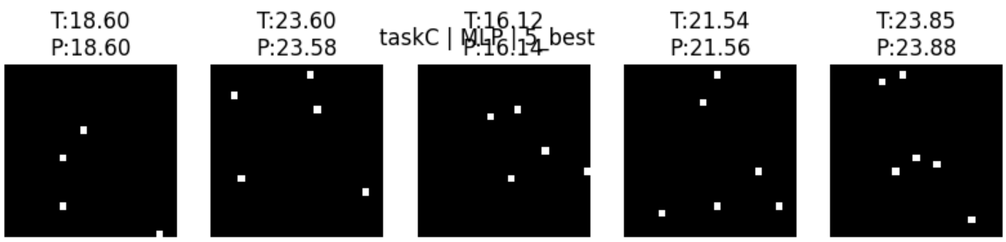

Figure 6.5: Task C / MLP: Five lowest-error examples. MLP matches true distances nearly exactly for these mid-range pairs

- In terms of accuracy and convergence speed, CNN is the preferred choice over MLP due to its ability to capture spatial locality and larger receptive fields.
- Remedies focus on data enhancement, weighted loss to emphasize tail values, and deeper architectures for better global context are among the most common.

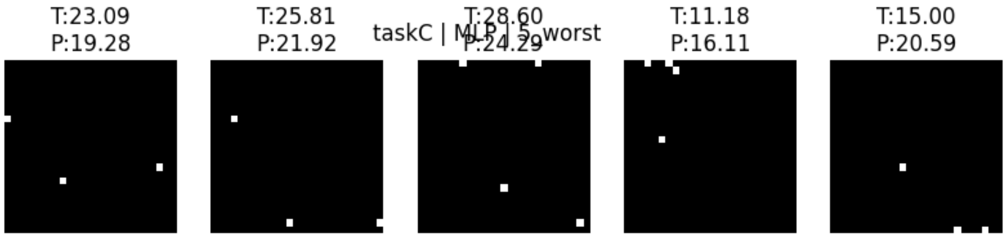

Figure 6.6: Task C / MLP: Five highest-error examples. Underestimation is more pronounced than CNN for corner-to-corner pairs (T:28.60 → P:24.29)

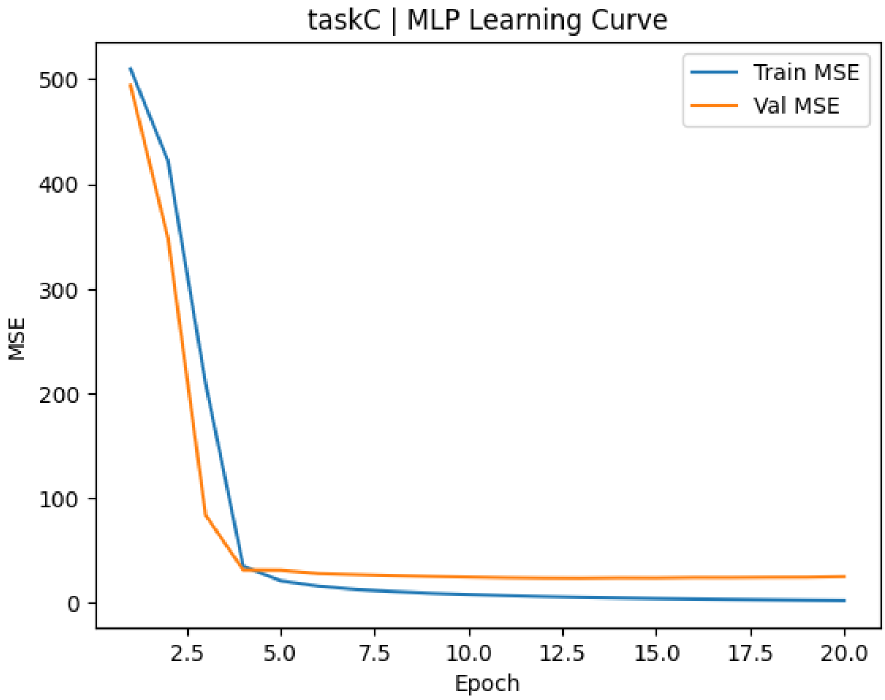

Figure 6.7: Task C / MLP Learning Curve: Training vs. validation MSE. MLP starts higher (MSE∼ 500) but converges to ∼ 10 by epoch 20, with a slight upward drift on validation

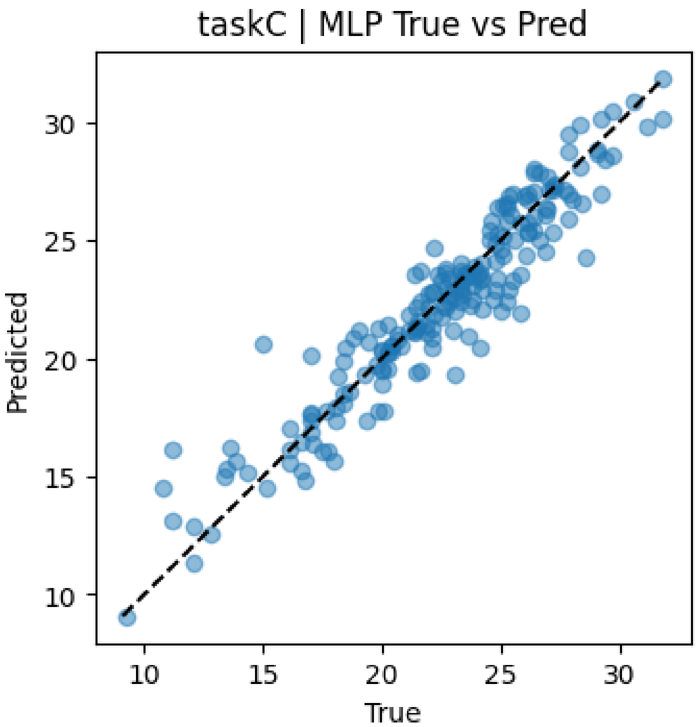

Figure 6.8: Task C / MLP: True vs. predicted distances. MLP shows larger variance than CNN, particularly for true values > 25

Chapter 7

Learning Curves & Scatter Plots

7.1 Learning Curves

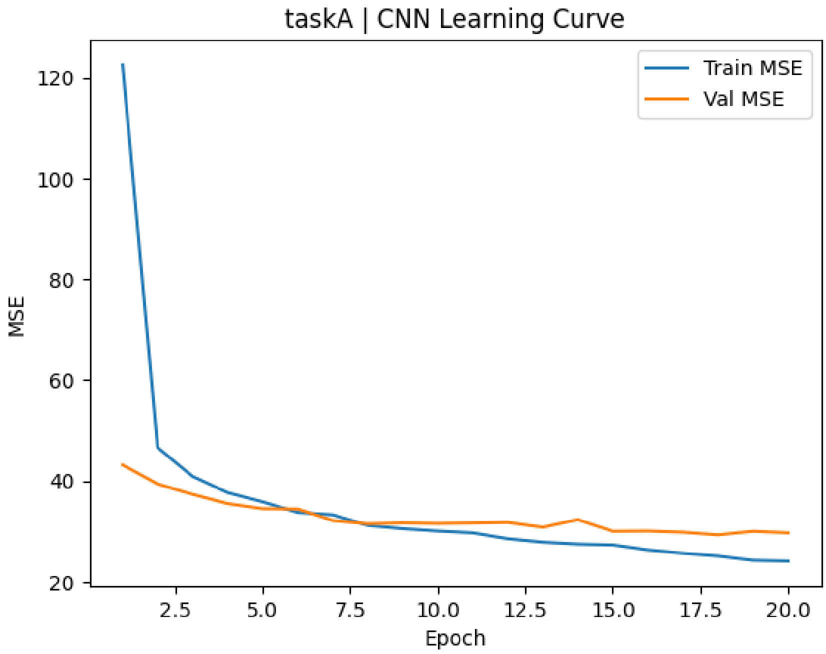

Figure 7.1: Training vs. validation MSE over epochs for Task A (CNN)

7.2 True vs. Predicted Scatter

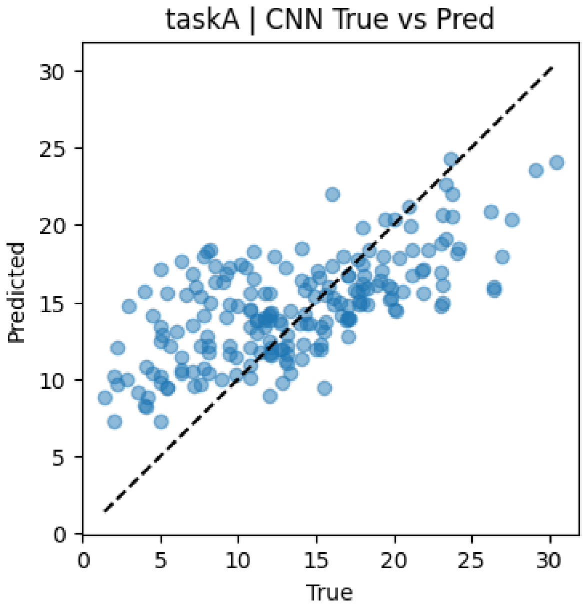

Figure 7.2: True vs. predicted distances for Task A (CNN). Points on the dashed line are perfect predictions

Chapter 8

Conclusion & Future Work

We demonstrated that simple CNNs can perform regression and counting on small binary grid sets with accuracy, outperforming MLPs and benefiting from larger training sets. Future directions:

- Extend to larger grids (e.g. 50×50, 100×100).
- Examine data augmentation and deeper architectures.
- Explore how image tasks can be used to transfer learning.

References/Github/YouTube

- **Code repository:** https://github.com/S-Salehin/AI_Project_grid/tree/master
- **YouTube Video:** https://youtu.be/ucsEzFd2P4Q
- **Data and models:** Synthetic datasets and trained models are included in the project ZIP.
- **Key packages:** NumPy, Pandas, Matplotlib, scikit-learn, TensorFlow/Keras

Appendix A

Code Repository & Usage

All scripts and data are organized as follows:

```
25x25_project/
 data/                        % .npz data files
 models/                      % .h5 Keras models
 results/                     % .npz training histories & CSV
 figures/                     % PNG plots
 data_gen.py
 models.py
 train.py
 hyperparam_tuner.py
 subset_experiments.py
 error_analysis.py
 best_hyperparams.json
 subset_results.csv
 report.pdf
```

Appendix B

Full Best/Worst Example Grids

Below are the complete "5 best" and "5 worst" grids for **Task A** with the CNN model. Repeat in a similar manner for all tasks/models if desired.

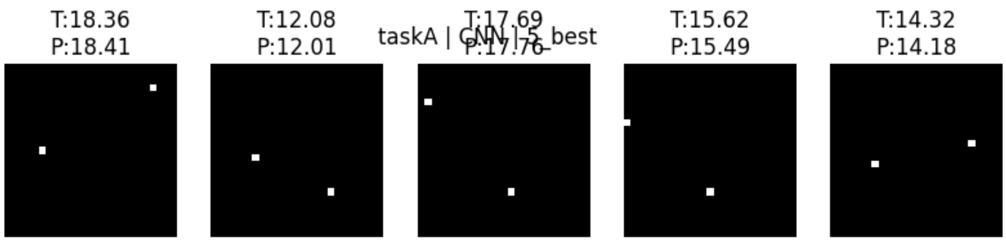

Figure B.1: Task A / CNN: Five lowest-error examples.

Figure B.2: Task A / CNN: Five highest-error examples.

Appendix C

Additional Tables

Best Hyperparameters (Full)

Table C.1: All best hyperparameters as discovered by grid search

Task	Model	Epochs	LR	Batch	Units
taskA	MLP	8	1e-3	32	128
taskA	CNN	8	1e-4	16	64
taskB	MLP	8	1e-3	16	128
taskB	CNN	8	1e-4	32	64
taskC	MLP	8	1e-3	32	128
taskC	CNN	8	1e-4	16	64
taskD	MLP	8	1e-3	16	128
taskD	CNN	8	1e-4	32	64
taskE	MLP	8	1e-3	32	128
taskE	CNN	8	1e-4	16	64