Bayesian Optimization for Hyperparameter Tuning in Neural Networks

Avdhoot Golekar (23B0060), Geet Sethi (23B2258), Panav Shah (23B3323)

November 5, 2025

1 Implementation Details

1.1 Calculating the Optimal x_{next}

To find the next x to evaluate the blackbox function on, we need to find the argmax over the acquisition function. However, doing so can be infeasible. So we instead randomly sample 1000 (customizable param) points in the X space and evaluate the acquisition function α on all of them and choose the max out of those. This gives us a good approixmation of the best x.

2 Task 1: Simple Neural Network Optimization

2.1 Experimental Setup

The given `SimpleNN` architecture consists of:

- Input layer: 784 neurons (28×28 flattened MNIST images)
- Hidden layer: Variable size (100-500 neurons)
- Output layer: 10 neurons (MNIST classes)
- ReLU activation and dropout regularization

The hyperparameter search space for the `SimpleNN` is as follows:

- Hidden size: {100, 200, 300, 400, 500}
- Epochs: {1, 2, ..., 10}
- Learning rate: 10^{-5} to 10^{-1} (log scale, 20 points)
- Batch size: {16, 32, 64, 128, 256}
- Dropout rate: 0 to 0.5 (linear scale, 20 points)
- Weight decay: 10^{-6} to 10^{-2} (log scale, 20 points)

2.2 Results for Different Kernel-Acquisition Combinations

Table 1 summarizes the results for all six kernel-acquisition function combinations with $N = 25$.

Table 1: Best hyperparameters and validation accuracies for `SimpleNN` with different kernel-acquisition combinations ($N = 25$)

Configuration	**Hidden Size**	**Epochs**	**Learning Rate**	**Batch Size**	**Dropout**	**Accuracy**
RBF + EI	500	6	0.000298	32	0.474	0.9696
RBF + PI	500	6	0.000298	32	0.474	0.9696
Matern + EI	400	9	0.000298	32	0.447	0.9722
Matern + PI	400	7	0.000785	32	0.500	**0.9743**
RQ + EI	200	7	0.00207	128	0.395	0.9725
RQ + PI	200	7	0.00207	128	0.395	0.9725

2.2.1 Observations

1. **Matern + PI** achieved the highest validation accuracy (0.9743), followed closely by Rational Quadratic configurations (0.9725) and Matern + EI (0.9722).
2. **RBF kernel** performed slightly worse (0.9696), suggesting the smoother `RBF` kernel may not capture the hyperparameter landscape as effectively as the `Matern` kernel.
3. **Acquisition function performance**: For Matern kernel, PI outperformed EI (0.9743 vs. 0.9722), while for RQ kernel, both performed equally (0.9725). This suggests that the choice between EI and PI depends on the kernel function.

2.3 Progression Plots

Figure 1 shows the validation accuracy progression for all six configurations. The plots reveal:

- **RBF kernel**: Steady but slower convergence, reaching plateau around iteration 12-15
- **Matern kernel**: Faster convergence with EI, achieving high accuracy earlier
- **Rational Quadratic kernel**: Similar performance with both acquisition functions
- All configurations show improvement from the initial random sampling phase (iterations 1-10)

2.4 Budget Analysis

Using the best configuration (Matern + PI), we analyzed the impact of budget $N \in \{15, 25, 50\}$. Table 2 shows the results.

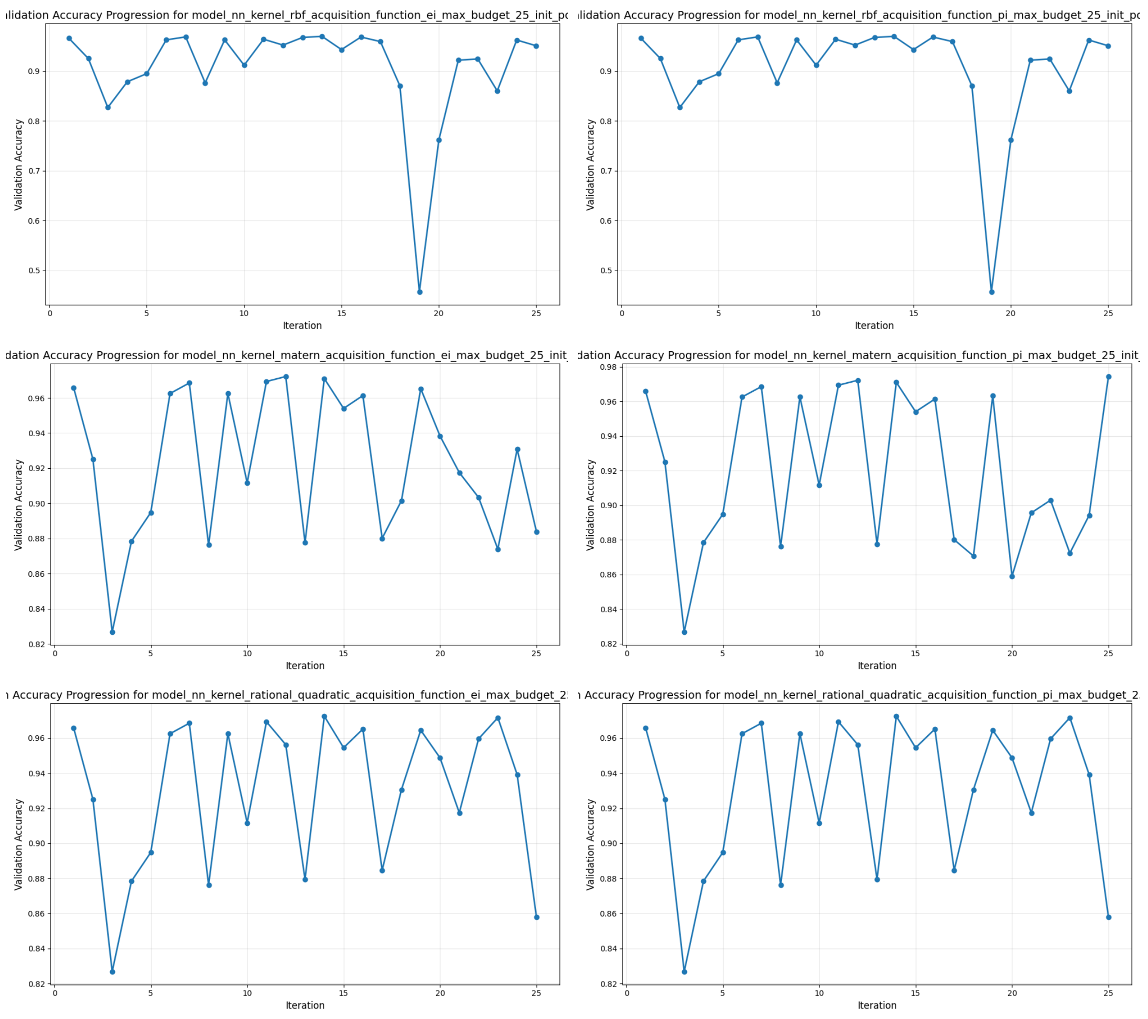

Figure 1: Validation accuracy progression for `SimpleNN` with different kernel-acquisition combinations ($N = 25$). Top row: `RBF` kernel; Middle row: `Matern` kernel; Bottom row: `Rational Quadratic` kernel. Left column: Expected Improvement; Right column: Probability of Improvement.

2.4.1 Analysis

1. **Progressive improvement**: The accuracy improved from 0.9722 ($N = 15$) to 0.9743 ($N = 25$) to 0.9771 ($N = 50$), showing consistent gains with more iterations.

2. **Computational trade-off**: While $N = 50$ yields better results, it requires twice the computational cost and the performance gain is not significant. For this problem, $N = 25$ provides a good balance between performance and computational efficiency.

Table 2: Impact of budget N on `SimpleNN` optimization (Matern + PI)

Budget N	**Best Validation Accuracy**
15	0.9722
25	0.9743
50	0.9771

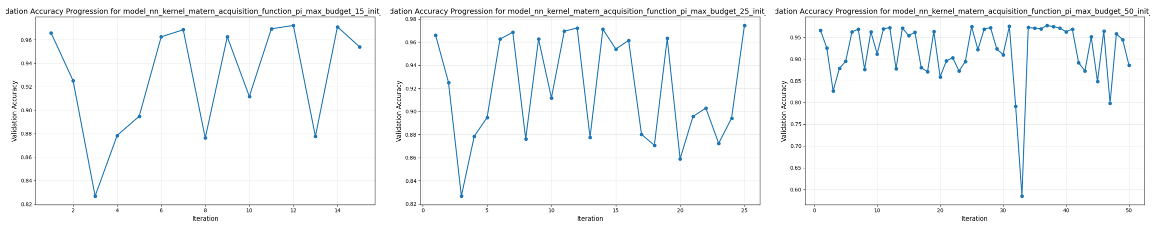

Figure 2: Validation accuracy progression for different budgets using Matern + PI. Left: $N = 15$, Middle: $N = 25$, Right: $N = 50$.

3 Task 2: Convolutional Neural Network Optimization

3.1 Model Architecture

The `CNN` architecture we came up with is as follows:

- **Block 1**: Two convolutional layers (32 filters each), BatchNorm, MaxPooling, Dropout
- **Block 2**: Two convolutional layers (64 filters each), BatchNorm, MaxPooling, Dropout
- **Classifier**: Fully connected layer (variable size), BatchNorm, Dropout, output layer

Key features:

- Batch normalization after each convolutional and fully connected layer
- Separate dropout rates for convolutional and fully connected layers
- Configurable kernel size for convolutional layers
- ReLU activation functions

3.2 Hyperparameter Search Space

The CNN hyperparameter search space includes:

- Hidden size FC: {100, 200, 300, 400, 500}
- Epochs: {1, 2, ..., 10}
- Learning rate: 10^{-5} to 10^{-1} (log scale, 20 points)

- Batch size: {16, 32, 64, 128, 256}
- Weight decay: 10^{-6} to 10^{-2} (log scale, 20 points)
- Dropout rate (conv): 0 to 0.5 (linear scale, 20 points)
- Dropout rate (FC): 0 to 0.5 (linear scale, 20 points)
- Kernel size: {3, 5, 7}

3.3 Results

Table 3 shows the best hyperparameters found for the `CNN` model across different kernel-acquisition combinations with $N = 50$.

Table 3: Best hyperparameters for `CNN` with different kernel-acquisition combinations ($N = 50$)

Configuration	**Best Validation Accuracy**
RBF + EI	0.9936
RBF + PI	0.9936
Matern + EI	**0.9941**
Matern + PI	0.9940
RQ + EI	0.9940
RQ + PI	0.9940

The hyperparameters for the best configuration (**Matern + EI**) are as follows:

- Hidden size FC: 500
- Epochs: 8
- Learning rate: 0.000070
- Batch size: 32
- Dropout rate (FC): 0.5
- Dropout rate (conv): 0.342
- Kernel size: 5

3.4 Progression Analysis

Figure 3 shows the validation accuracy progression for `CNN` optimization. Key observations:

- **Stable performance**: The CNN achieves high accuracy (above 0.99) very early in the optimization process, suggesting that the CNN architecture is well-suited for the MNIST dataset and that it is robust to hyperparameter variations and hence has less variance in performance across different hyperparameter configurations.
- **Matern + EI advantage**: Shows slightly better final performance and more consistent improvement over iterations.

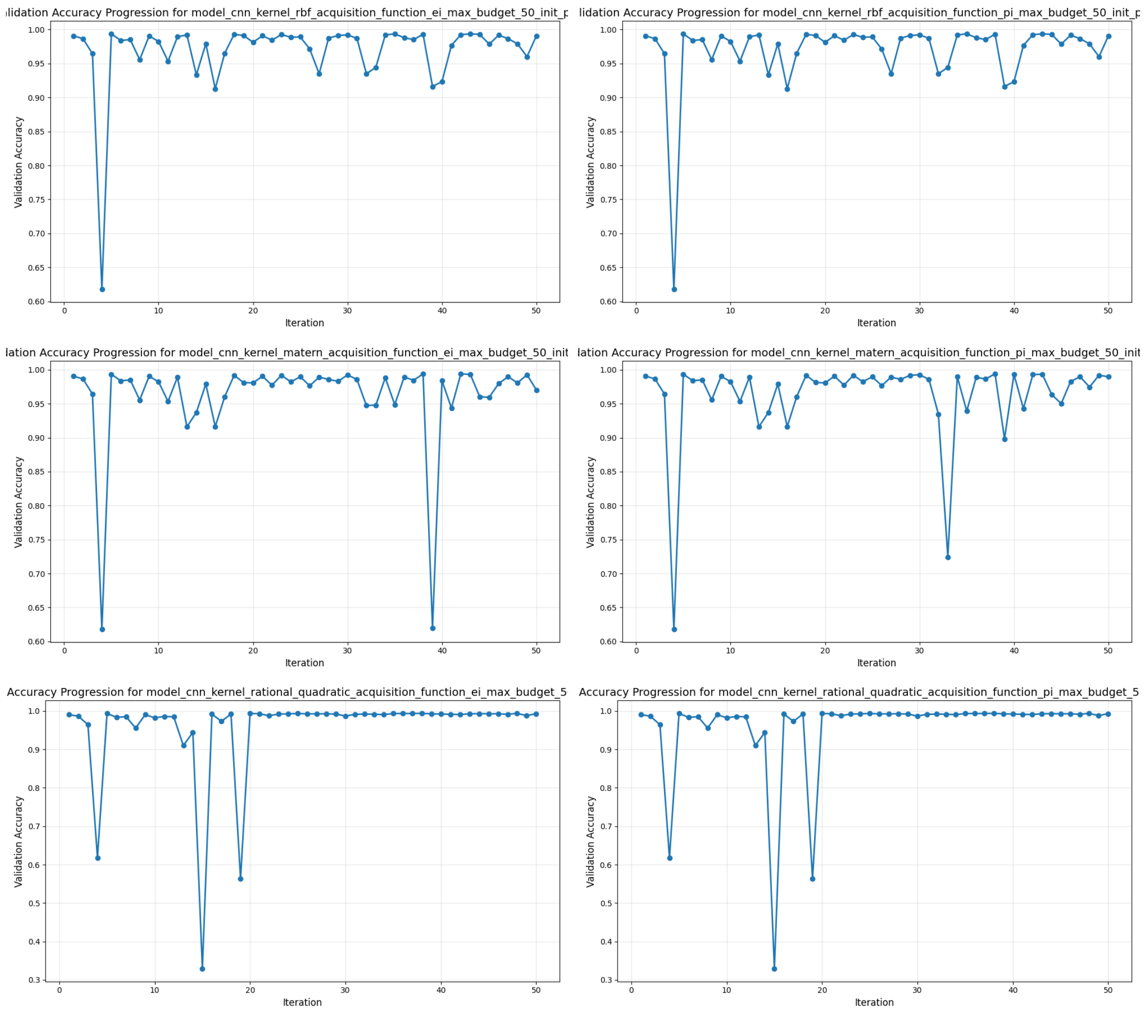

Figure 3: Validation accuracy progression for `CNN` with different kernel-acquisition combinations ($N = 50$). Top row: `RBF` kernel; Middle row: `Matern` kernel; Bottom row: `Rational Quadratic` kernel. Left column: Expected Improvement; Right column: Probability of Improvement.

4 Discussion and Conclusions

4.1 Key Findings

1. **Kernel selection matters**: The `Matern` kernel consistently outperformed the `RBF` and `Rational Quadratic` kernels, likely because it allows for less smooth functions and better captures the hyperparameter landscape.

2. **Acquisition function choice**: For SimpleNN, Probability of Improvement with Matern kernel performed best, while for CNN, Expected Improvement with Matern kernel achieved the highest accuracy.

3. **Budget impact**: Increasing the optimization budget from 25 to 50 iterations showed meaningful improvements, indicating that more exploration is beneficial, especially for complex search spaces.

4. **Architecture matters**: The CNN significantly outperformed the SimpleNN (99.41% vs. 97.43% best validation accuracy), demonstrating that convolutional architectures are better suited for image classification tasks.

5. **Hyperparameter patterns**: Optimal hyperparameters show consistent patterns:

   - Moderate learning rates (10^{-4} to 10^{-3})
   - Moderate to high dropout for regularization
   - Small to moderate batch sizes (32-128)
   - Appropriate network sizes for the task complexity