MNIST Digit Recognition: Naive Bayes Approaches and CNN Enhancements

Tanishque Zaware

I. Introduction

We study digit classification on the MNIST dataset (28×28 grayscale images, digits 0–9). As a baseline, we implement a simple statistical classifier on low-dimensional features(2D) for binary classification (digit 0 vs 1); we then transition to a CNN extended to 10 classes for improved accuracy. Our goals are to (i) establish a clear baseline, (ii) incrementally increase model capacity (kernel size and number of feature maps), and (iii) quantify the effect on error rates.

II. Solution

A. Data and Preprocessing

Dataset:

- **Part 1:** Number of samples in the training set: "0": 5923 ; "1": 6742. Number of samples in the testing set: "0": 980; "1": 1135.
- **Part 2**: Training samples: 60,000 and Testing samples: 10,000

For the statistical baseline, we extract compact features per image. For the deep model, We normalize pixel intensities to $[0, 1]$ by dividing the values by 255 and labels are one-hot encoded for 10 classes.

The digit labels (0-9) are transformed into one-hot encoded binary vectors, enabling the softmax output layer to handle multi-class predictions. For example, the label "8" is represented as:

$$[0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0]$$

where the position of "1" corresponds to the class index.

B. Statistical Baseline (Naive Bayes)

Here MNIST images dataset of handwritten digits is used and we are concerned only with digits 0 and 1 making it binary classification. The classification is based on the assumption that the features for each digit are drawn from a 2D normal distribution. It involves calculating density and classification using Naive Bayes classifier.

We compute two features per image: mean pixel value and standard deviation of pixel values.

1) **Average pixel value:**

$$f_1 = \frac{\sum_{i=1}^{n} x_i}{n}$$

where x_i represents the intensity of the i^{th} pixel and n is the total number of pixels in the image.

2) **Standard deviation of pixel values:**

$$f_2 = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

where μ is the mean pixel value of the image.

Thus, each image is represented as a compact 2D feature vector:

$$[\text{mean}, \text{standard deviation}]$$

Assuming class-conditional independence and Gaussianity, we estimate per-class means and standard deviations for each feature on the training set.

1) Mean of feature 1 (mean of pixel means)
2) Standard deviation of feature 1 (standard deviation of pixel means)
3) Mean of feature 2 (mean of pixel standard deviations)
4) Standard deviation of feature 2 (standard deviation of pixel standard deviations)

Classification uses the maximum a posteriori (MAP) decision with priors.

For each class, the mean and standard deviation are calculated as:

$$\mu_c = \frac{\sum f_i}{n_c}$$

$$\sigma_c = \sqrt{\frac{\sum (f_i - \mu_c)^2}{n_c}}$$

where μ_c and σ_c are the mean and standard deviation of each class c, and n_c is the total number of samples in class c.
Using Bayes' theorem:

$$P(C \mid x) = \frac{P(x \mid C)\, P(C)}{P(x)}$$

Where $P(C \mid x)$ is the posterior probability of class C. $P(x \mid C)$ is the likelihood of the feature vector under class C, modeled as the product of two independent normal distributions for f_1 and f_2. $P(C)$ is the prior probability of class C, computed from the training data. $P(x)$ is the evidence term, the same for all classes, and can be ignored during classification.

Thus, the decision rule becomes:

$$C^* = \arg\max_c P(x \mid C)\, P(C)$$

The class with the highest probability will be selected as the result.

Here, $P(x \mid C)$ is computed as:

$$P(x \mid C) = P([f_1, f_2] \mid C) = P(f_1 \mid C) \cdot P(f_2 \mid C)$$

Since both features are assumed to be independent, $P(f_1 \mid C)$ and $P(f_2 \mid C)$ are modeled as normal distributions.

For completeness, we present the univariate Gaussian likelihood for feature f and class c:

$$p(f \mid c) = \frac{1}{\sigma_c \sqrt{2\pi}} \exp\left(-\frac{(f-\mu_c)^2}{2\sigma_c^2}\right)$$

Equation: Class-conditional Gaussian probability density function used in the Naive Bayes baseline.

Here, f_i represents the feature of class C, and μ_c , σ_c are the mean and standard deviation of each class C.

The predicted class label is determined by selecting the class for which the product of probabilities is the highest.

Next, the prediction accuracy is calculated by taking the mean of the correctness of each prediction, where a value of 1 indicates a correct prediction and 0 indicates an incorrect one, compared to the expected result. Finally, both the individual class-wise accuracy and the overall accuracy are reported.

C. *Convolutional Neural Network (CNN)*

Our baseline CNN uses two convolutional blocks with ReLU and 2×2 pooling, followed by two fully connected layers and a softmax output. We then conduct two controlled modifications: (i) increase kernel sizes from 3×3 to 5×5, and (ii) increase the number of feature maps in the convolutional layers.

*1) **Baseline CNN**:* The CNN structure is as follows:

- **Conv Layer 1:** 6 feature maps, 3×3 kernel, stride 1, ReLU activation.
- **Max Pooling 1:** 2×2 pool size, stride 1.
- **Conv Layer 2:** 16 feature maps, 3×3 kernel, stride 1, ReLU activation.
- **Max Pooling 2:** 2×2 pool size, stride 1.
- **Fully Connected Layer 1:** 120 units, ReLU activation.
- **Fully Connected Layer 2:** 84 units, ReLU activation.
- **Output Layer:** 10 units (softmax activation).

Optimizer: Adadelta (learning rate = 0.1, ρ = 0.95)
Loss Function: Categorical cross entropy
Batch Size: 128
Epochs: 12

*2) **Experiments**:*

*a) **Experiment 1**:* Changing the kernel size to 5×5 The CNN structure is as follows:

- **Conv Layer 1:** 6 feature maps, 5×5 kernel, stride 1, ReLU activation.
- **Max Pooling 1:** 2×2 pool size, stride 1.
- **Conv Layer 2:** 16 feature maps, 5×5 kernel, stride 1, ReLU activation.
- **Max Pooling 2:** 2×2 pool size, stride 1.
- **Fully Connected Layer 1:** 120 units, ReLU activation.
- **Fully Connected Layer 2:** 84 units, ReLU activation.
- **Output Layer:** 10 units (softmax activation).

Optimizer: Adadelta (learning rate = 0.1, ρ = 0.95)
Loss Function: Categorical cross entropy
Batch Size: 128
Epochs: 12

*b) **Experiment 2**:* Changing the number of feature maps The CNN structure is as follows:

- **Conv Layer 1:** 12 feature maps, 5×5 kernel, stride 1, ReLU activation.
- **Max Pooling 1:** 2×2 pool size, stride 1.
- **Conv Layer 2:** 20 feature maps, 5×5 kernel, stride 1, ReLU activation.
- **Max Pooling 2:** 2×2 pool size, stride 1.
- **Fully Connected Layer 1:** 120 units, ReLU activation.
- **Fully Connected Layer 2:** 84 units, ReLU activation.
- **Output Layer:** 10 units (softmax activation).

Optimizer: Adadelta (learning rate = 0.1, ρ = 0.95)
Loss Function: Categorical cross entropy
Batch Size: 128
Epochs: 12

III. RESULTS

We report the estimated parameters for each class, along with test accuracy and error across different configurations. These results summarize both the Naive Bayes baseline and CNN experiments, highlighting the effect of kernel size and feature map variations.

A. *Naive Bayes Classifier*

The estimated parameters for f_1 and f_2 features are:

1) **Digit "0"**:
   - Mean (μ_0): [0.17339933, 0.34289137]
   - Standard deviation (σ_0): [0.04210271, 0.03950485]
2) **Digit "1"**:
   - Mean (μ_1): [0.07599864, 0.24066195]
   - Standard deviation (σ_1): [0.02199118, 0.03566054]

Test Accuracy:

- Digit "0": 91.43%
- Digit "1": 92.42%
- **Overall accuracy:** 91.93%

These results indicate that the Naive Bayes classifier performs well for this binary classification task, achieving over 90% accuracy for both digits "0" and "1".

B. *Baseline CNN*

- **Test loss:** 0.0674
- **Test accuracy:** 97.88%
- **Test error:** 2.12%

- **Observation:** The network showed steady convergence with accuracy plateauing near 98%.
- **Learning behavior:** The model converged quickly and steadily, with validation loss closely following training loss, indicating minimal overfitting. Test error plateaued early (around epoch 8).

C. *Experiment 1: Increased Kernel Size* (5×5)

- **Change:** Increased kernel size in both convolution layers from 3×3 to 5×5.
- **Test loss:** 0.0576
- **Test accuracy:** 98.19%
- **Test error:** 1.81%
- **Observation:** Larger kernels slightly improved accuracy, likely due to capturing more spatial context per feature map.
- **Learning behavior:** Validation loss remained below training loss, suggesting good generalization and that the model performed well on unseen data despite minor noise in training data.

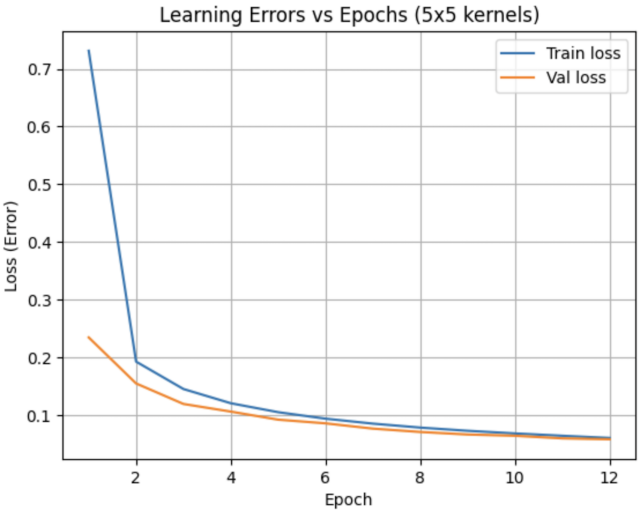

Fig. 1: Training and validation loss versus epochs for Experiment 1 using 5×5 kernels. The validation loss remains consistently below the training loss, indicating that the model generalizes well to unseen data while avoiding overfitting.

D. *Experiment 2: Increased Number of Feature Maps*

- **Change:** Increased filters in Conv1 from 6 to 12 and in Conv2 from 16 to 20, keeping 5×5 kernels.
- **Test loss:** 0.0538
- **Test accuracy:** 98.40%
- **Test error:** 1.60%
- **Observation:** Increasing the number of feature maps improved the network's representational capacity, yielding the best accuracy across all configurations.
- **Learning behavior:** Larger feature maps enabled finer feature extraction and reduced error significantly compared to the baseline.

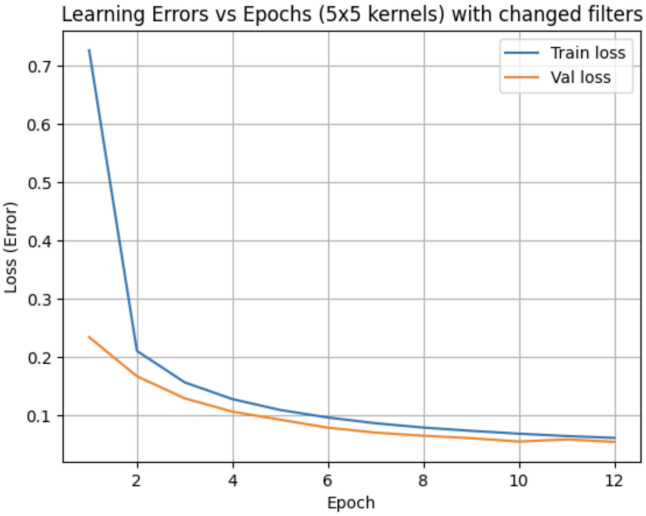

Fig. 2: Training and validation loss versus epochs for Experiment 2, where the number of features in Conv1 and Conv2 was increased while using 5×5 kernels. The model converges smoothly, and the validation loss remains consistently below the training loss, suggesting strong generalization and minimal overfitting.

Model	Accuracy (%)	Error (%)
Naive Bayes (2 features)	91.93	8.07
CNN Baseline (3×3)	97.88	2.12
CNN (5×5 kernels)	98.19	1.81
CNN (more feature maps)	98.40	1.60

TABLE I: Performance comparison across models on MNIST.

IV. Contribution

I solely implemented the baseline feature extraction and Naive Bayes training, including parameter estimation and test evaluation. I modified the CNN by increasing kernel sizes and feature maps, executed training runs, and produced the learning-curve plots. I also prepared the comparison table and analyzed where model capacity helped most.

V. Lessons Learned

We observed consistent gains from moderate increases in CNN capacity (kernel size and number of filters). However, diminishing returns suggest balancing complexity and overfitting risk. We also learned that even simple statistical baselines offer a transparent reference point and can highlight what the CNN learns beyond global intensity statistics. Future work includes data augmentation, regularization (dropout), and exploring deeper architectures.

Acknowledgments

This work uses the publicly available MNIST handwritten digit dataset [1].

References

[1] Y. LeCun and C. Cortes, "MNIST handwritten digit database," *ATT Research*, 2010. [Online]. http://yann.lecun.com/exdb/mnist/

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.