Classification of Rotated MNIST Digits using Convolutional Neural Networks

Noppachon Chaisongkhram
School of Engineering and Computer Science, Victoria University of Wellington
chaisonopp@myvuw.ac.nz

***Abstract*—This report addresses a multi-class classification task on the MNIST dataset, where digits are rotated by 0°, 90° left, or 90° right. A convolutional neural network (CNN) was trained using stochastic gradient descent (SGD) with momentum. The model achieved a test accuracy of 0.9938, demonstrating robust invariance to rotational transformations. Code is publicly accessible via GitHub and Google Colab.**

***Index Terms*—Deep Learning, Convolutional Neural Networks, Stochastic Gradient Descent, MNIST Dataset, Image Rotation Classification, Reproducible Machine Learning, Deterministic Training, Spatial Invariance, TensorFlow**

I. Introduction

Handwritten digit recognition is a canonical computer vision problem, but model robustness to spatial transformations remains challenging [1, 2]. This report tackles a rotation-invariant classification task: given MNIST digits rotated by 0°, 90° left, 90° right, predict the rotation class. The problem tests a model's ability to disentangle spatial transformations from content, where we frame this as a 3-class classification problem using a CNN optimised via SGD with momentum [3, 4].

II. Theory

A. Problem Formulation

Let $\mathbf{x} \in \mathbb{R}^{28\times28}$ be an input image [1]. The target class $c \in \{0, 1, 2\}$ corresponds to rotation angles 0°, 90° left, 90° right. The goal is to learn a mapping:

$$f_\theta : \mathbf{x} \mapsto \hat{\mathbf{y}} \tag{1}$$

where $\hat{\mathbf{y}}$ is a probability vector over classes.

B. Softmax Activation for Multi-Class Probability

Since outputs must be probabilities summing to 1, we use the softmax function [5]:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2}$$

Softmax converts logits $\mathbf{z}$ into class probabilities because:

1) Non-negativity: $e^{z_i} > 0$ ensures $\sigma(\mathbf{z}_i) > 0$.
2) Normalisation: $\sum_i \sigma(\mathbf{z})_i = 1$.
3) Monotonicity: Larger $z_i \rightarrow$ higher probability.

C. Conditional Probability Model

Let $\mathbf{z} = f_\theta(\mathbf{x})$ be the network's logits. The conditional probability of class c given $\mathbf{x}$ is [5]:

$$q_\theta(c|\mathbf{x}) = \sigma(f_\theta(\mathbf{x}))_c = \frac{\exp(f_\theta(\mathbf{x})_c)}{\sum_{k=0}^{2} \exp(f_\theta(\mathbf{x})_k)} \tag{3}$$

This satisfies $\sum_{c=0}^{2} q_\theta(c|\mathbf{x}) = 1$, forming a valid probability distribution.

D. Cross-Entropy Loss from KL Divergence

To measure fit between true labels $p(c|\mathbf{x})$ and predictions $q_\theta(c|\mathbf{x})$, we minimise Kullback-Leibler (KL) divergence [6]:

$$D_{\text{KL}}(p\|q_\theta) = \sum_c p(c|\mathbf{x}) \log \frac{p(c|\mathbf{x})}{q_\theta(c|\mathbf{x})} \tag{4}$$

For one-hot labels (where $p(c|\mathbf{x}) = 1$ if $c = c_{\text{true}}$, else 0), this simplifies to cross-entropy loss [5]:

$$\mathcal{L}(\theta) = -\log q_\theta(c_{\text{true}}|\mathbf{x}) \tag{5}$$

For a batch of N samples:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log q_\theta(c^{(i)}|\mathbf{x}^{(i)}) \tag{6}$$

Minimising $\mathcal{L}(\theta)$ maximises the likelihood of correct labels.

E. Stochastic Gradient Descent with Momentum

Parameters θ are updated iteratively [4]. Let $\nabla_\theta \mathcal{L}$ be the gradient computed via backpropagation. The momentum-SGD update uses velocity $\mathbf{m}$:

$$\mathbf{m}_{t+1} = \beta \mathbf{m}_t + \eta \nabla_\theta \mathcal{L}(\theta_t) \tag{7}$$

$$\theta_{t+1} = \theta_t - \mathbf{m}_{t+1} \tag{8}$$

with $\beta = 0.9$ (momentum decay), $\eta = 0.01$ (learning rate). Momentum accelerates convergence in consistent gradient directions [4].

F. Architecture Choice

A CNN is ideal due to [2, 3]:

1) Translation invariance (convolutional filters)
2) Hierarchical feature learning (edges $\rightarrow$ shapes)
3) Parameter efficiency (weight sharing)

III. Experiments

A. Data Preparation

The MNIST dataset (60,000 training, 10,000 test images) was normalised to [0,1] through division by 255 [1]. This preprocessing step:

$$|\mathbf{x}| = \frac{\mathbf{x}_{\text{raw}}}{255} \tag{9}$$

ensures numerical stability during optimisation by preventing large gradient magnitudes that occur with high-input values [7]. Without normalisation, weight updates become unstable due to exploding gradients, particularly in deep architectures where chain rule amplification exacerbates sensitivity to input scale.

Rotation transformations were applied with precise geometric consistency [8]:

- 0° rotation: Identity transformation preserving original orientation
- 90° left (CCW): Linear transformation matrix $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ applied per pixel
- 90° right (CW): Linear transformation $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

The balanced 1:1:1 class distribution prevents model bias while maintaining the original data manifold structure. The 80/20 train/validation split provides:

1) Sufficient training samples (38,400) for parameter learning.
2) Representative validation set (9,600) for unbiased generalisation assessment.
3) Statistical power to detect overfitting.

B. Model Architecture

The architecture implements a visual processing hierarchy [2, 3]:

1) Conv2D(32, 3×3):
   - Kernel size 3×3 provides optimal receptive fields for detecting strokes and curves [3]
   - 32 filters enable learning diverse rotation-sensitive features
   - ReLU activation ($\max(0, x)$) induces sparsity while preserving gradient flow [9]:

$$\frac{d}{dx}\text{ReLU}(x) = \begin{cases} 1 & x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

2) MaxPooling2D(2×2):
   - Implements spatial invariance through downsampling [2]
   - Computes $\text{output}(i, j) = \max(\text{window22})$
   - Reduces spatial dimensions $28\times28 \mapsto 14\times14$ (75% dimensionality reduction)
   - Preserves dominant activation patterns for rotation discrimination
3) Feature Integration:
   - Flattening converts spatial features to vector representation
   - Dense(128) with ReLU enables non-linear feature combination
   - Final Dense(3) with softmax produces probabilistic outputs [5]

C. Training Configuration

SGD with Momentum ($\beta = 0.9$) [4]:

$$\mathbf{v}t = \beta\mathbf{v}t - 1 + (1-\beta)\nabla_\theta J(\theta) \tag{11}$$

$$\theta_t = \theta_{t-1} - \eta\mathbf{v}_t \tag{12}$$

accelerates convergence in low-curvature directions while damping oscillations. A batch size of 64 balances gradient estimation noise (regularisation effect) and computational efficiency per [7]:

$$\text{Var}(\nabla_\theta J_{\text{batch}}) \propto \frac{1}{\sqrt{N_{\text{batch}}}} \tag{13}$$

An epoch of 15 is determined by early stopping criteria monitoring validation loss plateau, preventing overfitting while ensuring convergence [10].

IV. Conclusion

The CNN achieved 99.38% test accuracy by integrating: 1) A geometric hierarchy where convolutional layers learned orientation-sensitive filters and max-pooling induced transformation invariance, 2) Probabilistic optimisation via softmax outputs minimising cross-entropy (equivalent to KL divergence reduction), and 3) Stable convergence through momentum SGD and activation normalisation. The solution demonstrates CNNs' inherent capability to learn geometric transformations, enabling applications in medical imaging orientation analysis and robotic vision systems.

V. Statement

This work represents my original implementation. The solution was developed using TensorFlow, NumPy, and Matplotlib within PyCharm Jupyter Notebook. The complete code, including data generation and visualisation routines, is accessible via GitHub and Google Colab at:

- https://github.com/jescsk/AIML425/tree/main/Assignment2
- https://shorturl.at/1HBJR

References

[1] LeCun, Y., Cortes, C. and Burges, C.J.C. (1998) The MNIST Database of Handwritten Digits. New York, USA. http://yann.lecun.com/exdb/mnist/

[2] T. Cohen and M. Welling, "Group Equivariant Convolutional Networks," PMLR, pp. 2990–2999, Jun. 2016, Accessed: Aug. 17, 2025. [Online]. Available: https://proceedings.mlr.press/v48/cohenc16.html

[3] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[4] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," proceedings.mlr.press, May 26, 2013. https://proceedings.mlr.press/v28/sutskever13.html

[5] C. Bishop, Pattern recognition and machine learning. Springer Verlag, 2006.

[6] T. M. Cover and J. A. Thomas, Elements of Information Theory. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2005. doi: https://doi.org/10.1002/047174882x.

[7] S. Ruder, “An overview of gradient descent optimization algorithms,” arXiv.org, Jun. 15, 2017. https://arxiv.org/abs/1609.04747?ref=ruder.io

[8] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, 2004.

[9] V. Nair and G. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” 2010. Available: https://www.cs.toronto.edu/ fritz/absps/reluICML.pdf

[10] Prechelt, L. (1998). Early Stopping - But When?. In: Orr, G.B., Müller, KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-49430-8_3