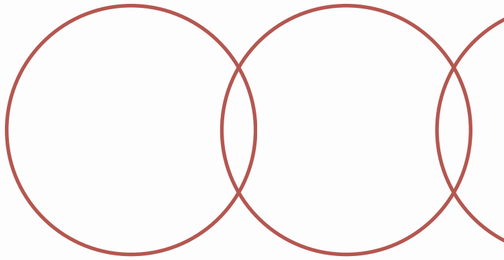
# Nhận dạng chữ số và hình học đơn giản bằng CNN

Nhóm 12
Sinh viên:
Võ Sỹ Tài - B22DCCN706
Lê Đức Toàn - B22DCCN730

# Nhận dạng chữ số viết tay và hình học bằng CNN

Đề tài này tập trung vào việc hiểu quy trình xử lý ảnh, trích xuất đặc trưng và ứng dụng thực tế của CNN trong nhận dạng chữ viết tay và hình học.

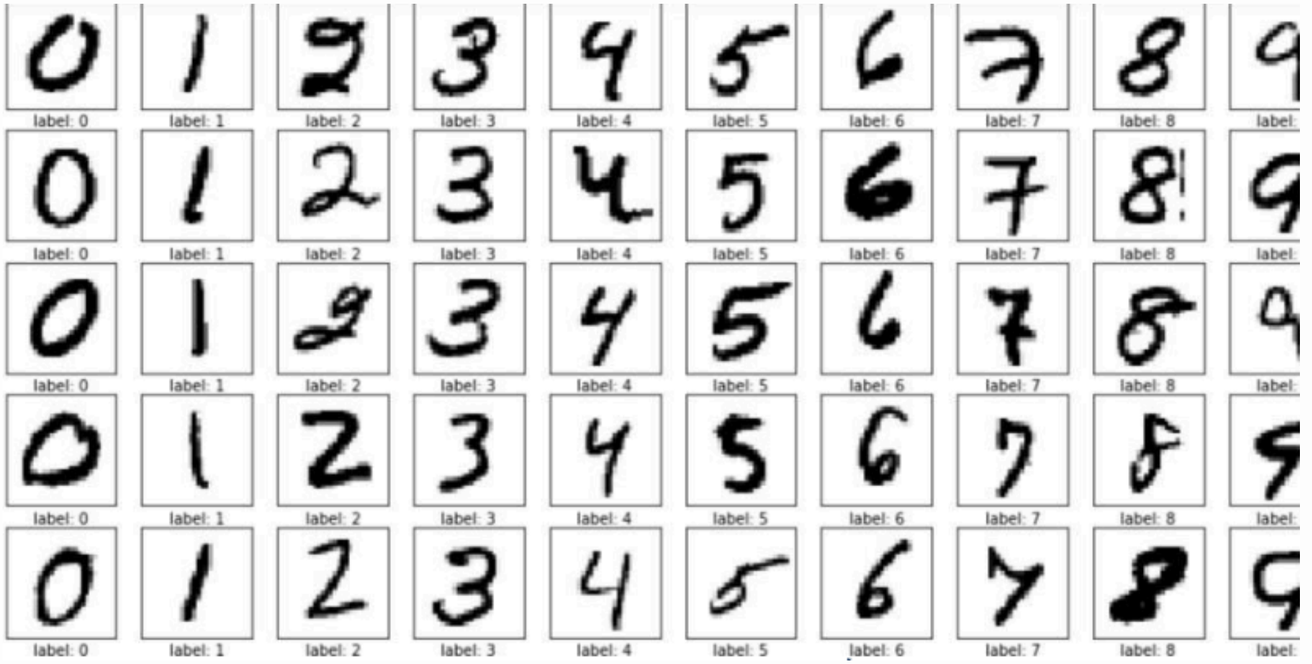


# Tập dữ liệu MNIST

Dataset chuẩn (benchmark) để kiểm thử các kiến trúc mạng nơ-ron cơ bản

Sử dụng 70.000 ảnh tổng cộng:

- Training: 48.000 ảnh
- Validation: 12.000 ảnh
- Test: 10.000 ảnh



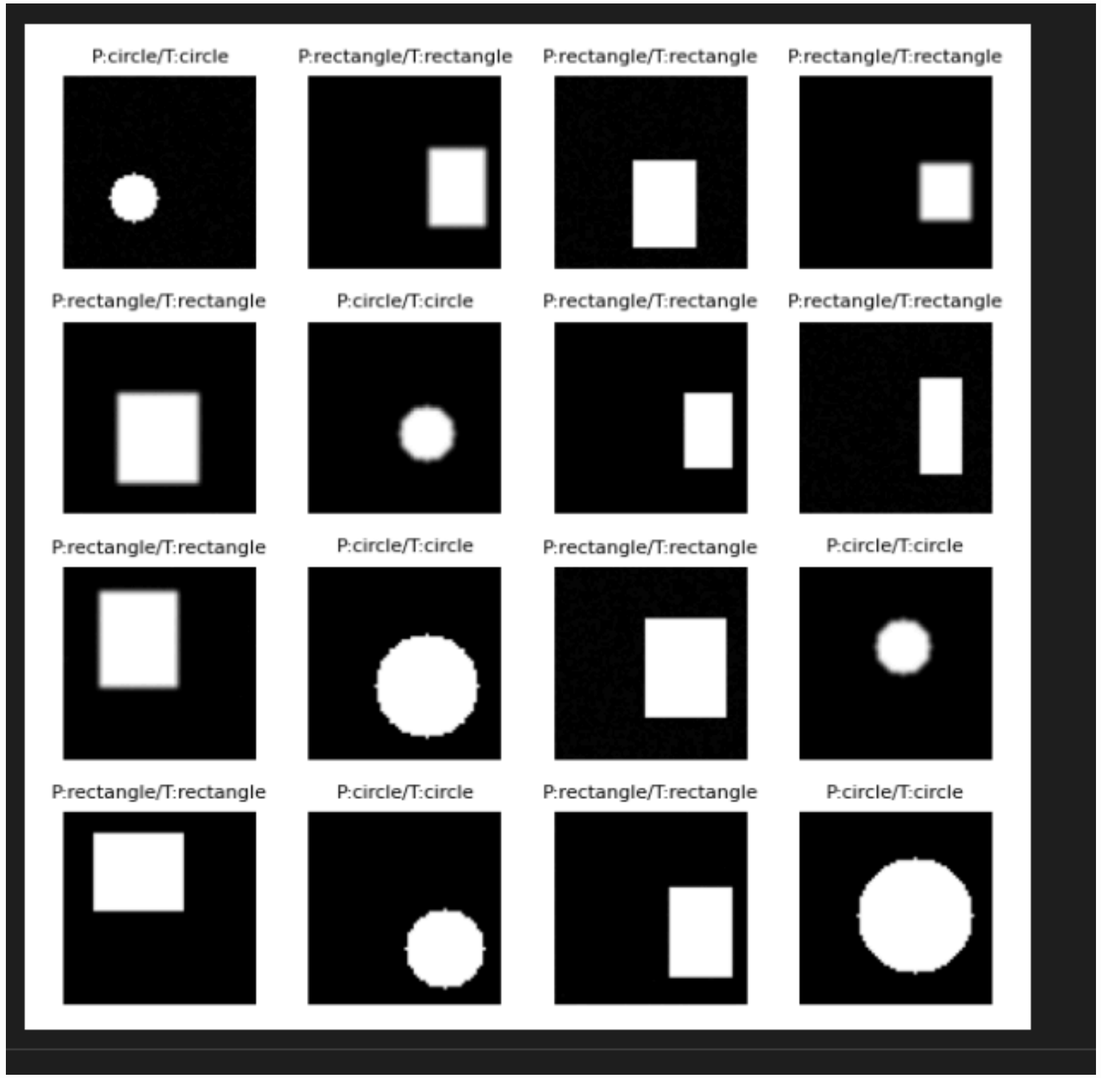

- Kích thước 28x28 pixel
- Định dạng ảnh xám, giá trị 0 - 255
- 10 lớp (0 - 9)

# Tập dữ liệu

Sử dụng 6.000 ảnh tổng cộng:

- Training: 4.800 ảnh
- Validation: 1.200 ảnh

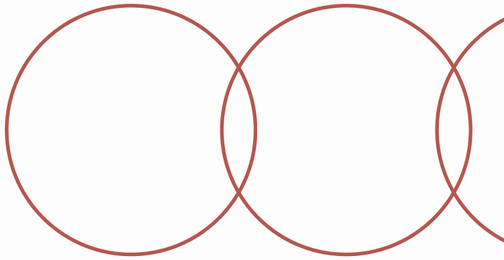
- Kích thước 64x64 pixel
- Định dạng ảnh xám, giá trị 0 - 255
- 2 lớp (circle, rectangle)



# Tại sao xử lý ảnh?

Tăng cường hiệu quả CNN

Việc xử lý ảnh trước khi áp dụng CNN rất quan trọng để giảm thiểu nhiễu, đồng nhất dữ liệu đầu vào, và giúp mô hình học tập đặc trưng chính xác hơn.



# Tiền xử lý dữ liệu

## Quá trình xử lý bao gồm 6 bước tuần tự:

1. Read Image (Đọc ảnh):Đọc ảnh từ file và chuyển sang grayscale bằng PIL. Kết quả là numpy array float32 với giá trị pixel 0–255.
2. Invert If Needed (Đảo màu):Nếu nền ảnh sáng (mean > 127), đảo màu để đối tượng nổi bật trên nền tối.
3. Edge Enhancement / Detection (Tăng cường cạnh):

- MNIST: Laplacian sharpening + Adaptive Histogram Equalization để tăng độ sắc nét và tương phản.
- Shapes: Sobel edge detection kết hợp với ảnh gốc để làm nổi bật biên.

4. Morphological Operations (Xử lý hình thái) Shapes: áp dụng Dilation để mở rộng vùng trắng, cải thiện kết nối và hình dạng đối tượng.
5. Resize (Đồng bộ kích thước):

- Resize ảnh về kích thước cố định (MNIST: 28×28
- Shapes: imgsz×imgsz) bằng bilinear interpolation để phù hợp input layer.

6. Normalize (Chuẩn hóa giá trị pixel):Chia pixel cho 255, đưa về [0,1] để dữ liệu sẵn sàng cho mô hình học sâu.

# Giới thiệu về CNN

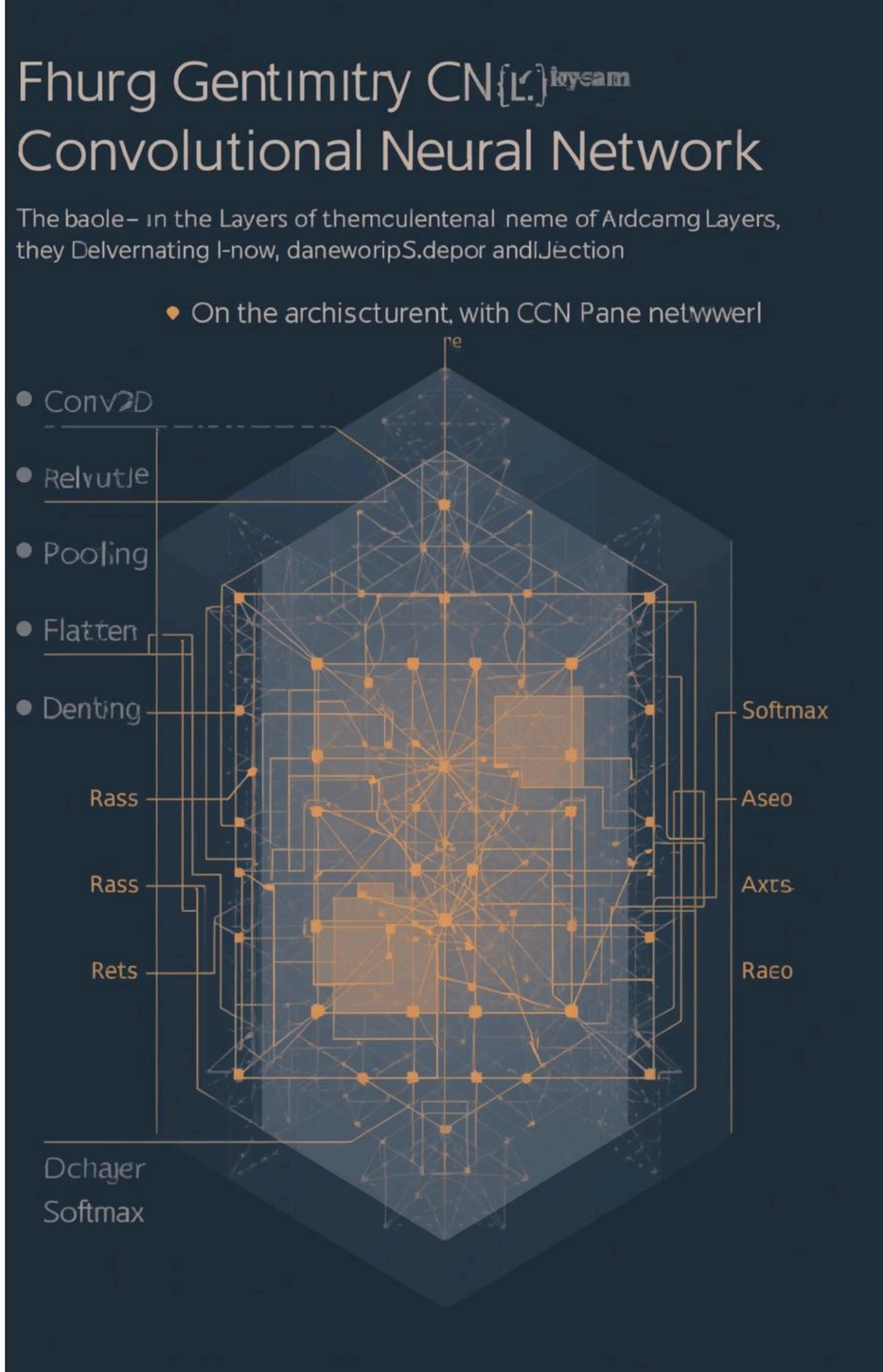
CNN, hay mạng tích chập, là một công nghệ mạnh mẽ trong nhận dạng hình ảnh, cho phép tự động học đặc trưng mà không cần phải trích chọn thủ công như trước đây.

Tổng quan kiến trúc: 2 giai đoạn chính

- Giai đoạn 1: Trích xuất đặc trưng (Feature Extraction): Sử dụng các lớp tích chập (Conv) và giảm chiều dữ liệu (Pooling).
- Giai đoạn 2: Phân loại (Classification): Sử dụng các lớp kết nối đầy đủ (Fully Connected/Linear) để đưa ra dự đoán.

Mô hình cho MNIST :

1. Input: Ảnh 28×28 pixel.
2. Block 1:
   - Conv2D (32 filters, kernel 3×3) -> ReLU.
   - MaxPooling (2×2).
3. Block 2:
   - Conv2D (64 filters, kernel 3×3) -> ReLU.
   - MaxPooling (2×2).
4. Classifier (Phân loại):
   - Flatten (Dát phẳng dữ liệu).
   - FC (128 neurons) → ReLU -> Dropout (0.25) (Chống overfitting).
   - Output Layer: FC (10 neurons) → Softmax (Xác suất 10 lớp).



# Kiến trúc mạng nơ-ron được đề xuất

MNIST model:

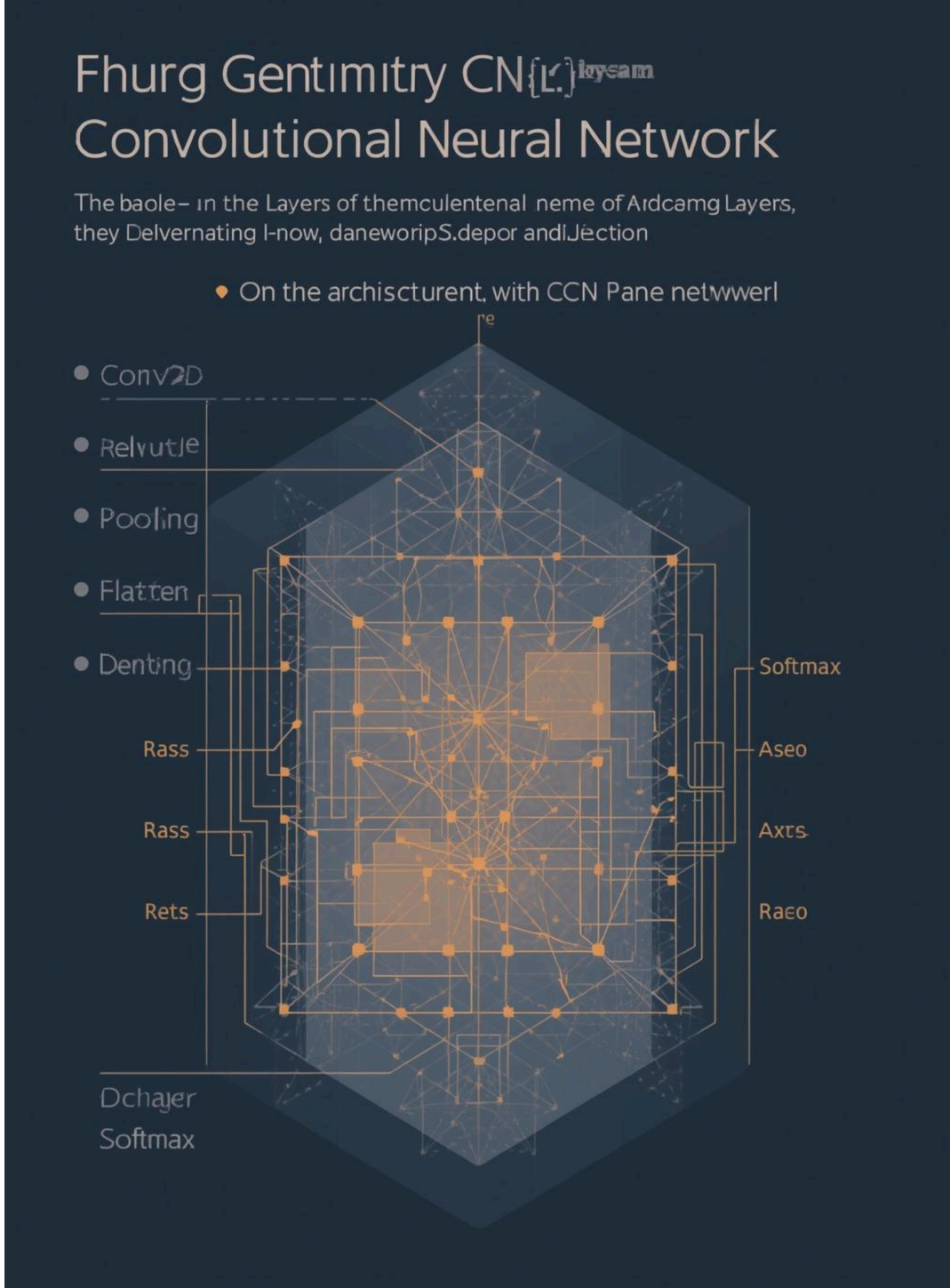
Conv(32,3×3) → ReLU → MaxPool(2×2) →

Conv(64,3×3) → ReLU → MaxPool → Flatten →

FC(128) → ReLU → Dropout(0.25) → FC(10).

Mô hình cho Shapes :

1. Cấu trúc: 3 lớp Conv (32 -> 64 ->128 filters).
2. Đặc biệt: Sử dụng AdaptiveAvgPool (1×1) trước khi vào lớp phân loại để xử lý kích thước linh hoạt hơn.
3. Output: FC (2 neurons) cho 2 lớp (Circle, Rectangle).



# Kiến trúc mạng nơ-ron được đề xuất

Shapes model:

Conv(32,3×3,pad=1) → ReLU → MaxPool → Conv(64,3×3) → ReLU → MaxPool → Conv(128,3×3) → ReLU → AdaptiveAvgPool(1×1) → Flatten → Dropout → FC(2).

# Lớp tuyến tính (Linear Layer)

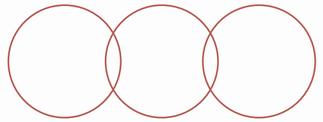
Công thức lan truyền thuận (Forward):

$$y = xW + b$$

- x: Vector đầu vào (Input)
- W: Ma trận trọng số (Weights) - Khởi tạo bằng phương pháp He Initialization
- b: Vector hệ số chệch (Bias)

Thực hiện phép biến đổi tuyến tính trên dữ liệu đầu vào

- Đầu vào đa chiều (Tensor 4D) sẽ được làm phẳng (Flatten) thành ma trận 2D trước khi nhân
- Gradient được tính toán cho cả W, b và đầu vào x trong quá trình lan truyền ngược để cập nhật trọng số



# HÀM KÍCH HOẠT (ACTIVATION FUNCTION) - ReLU

Hàm ReLU (Rectified Linear Unit):

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} 1 \text{ nếu } x > 0 \\ 0 \text{ nếu } x \leq 0 \end{cases}$$

Đưa tính phi tuyến (non-linearity) vào mạng, giúp mạng học được các mối quan hệ phức tạp

- Ưu điểm: Tính toán cực nhanh, giải quyết tốt vấn đề biến mất đạo hàm (Vanishing Gradient) trong mạng sâu.
- Nhược điểm: Có thể gặp hiện tượng "Dying ReLU" (nơ-ron chết) nếu đầu vào luôn âm

# HÀM SOFTMAX (LỚP ĐẦU RA)

Công thức: $Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_i}}$

Trong đó:

- $x_i$: giá trị đầu vào (logit) của lớp thứ $i$.
- $n$: tổng số lớp (classes).
- $e^{x_i}$: chuyển đổi giá trị $x_i$sang dạng mũ dương.
- Mẫu số $\sum_{j=1}^{n} e^{x_j}$: tổng của tất cả các giá trị mũ — giúp chuẩn hóa kết quả thành xác suất.

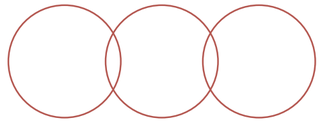
Softmax chuyển các giá trị đầu ra cuối cùng thành phân bố xác suất trên các lớp. Tầng này đảm bảo tổng xác suất bằng 1, giúp mô hình đưa ra dự đoán lớp chính xác và rõ ràng.

# HÀM MẤT MÁT (LOSS FUNCTION)

Hàm Cross-Entropy:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})$$

- Tiêu chuẩn cho bài toán phân loại đa lớp (Multi-class Classification).
- Đo lường sự khác biệt giữa phân phối xác suất dự đoán (^y) và nhãn thực tế (y)



# THUẬT TOÁN TỐI ƯU HÓA ADAM

**Quy trình tính toán**

- **Tính Moment (Ước lượng):**
  - Moment bậc 1 (Trung bình): $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  - Moment bậc 2 (Phương sai): $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
- **Bias Correction (Khắc phục thiên lệch về 0):**
  - $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ ; $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$
- **Cập nhật tham số:**
  - $\theta_{t+1} = \theta_t - \frac{\alpha \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$

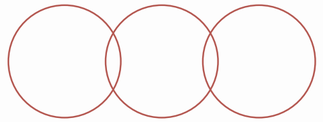
Giúp mạng nơ-ron học được các trọng số tối ưu một cách nhanh chóng (tiết kiệm thời gian) và ổn định (đạt độ chính xác cao >96%).

# LỚP FLATTEN (LÀM PHẲNG DỮ LIỆU)

Là cầu nối chuyển đổi dữ liệu đa chiều sang dạng 2D
để đưa vào lớp tuyến tính (Linear Layer).

Cơ chế thực hiện (Implementation)

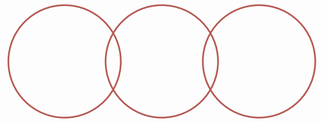
- Forward Pass (Lan truyền thuận):
  - Làm phẳng các chiều (trừ chiều batch).
  - Lưu lại kích thước gốc (Input Shape) vào cache để dùng cho bước sau.
- Backward Pass (Lan truyền ngược):
  - Nhận gradient đầu ra (dạng phẳng).
  - Sử dụng shape đã lưu trong cache để reshape (khôi phục hình dạng) gradient về kích thước ban đầu.
  - Đảm bảo lan truyền ngược hoạt động chính xác.



# MNIST Model

```
self.conv1 = nn.Conv2d(1, 32, 3, padding=1)
self.bn1 = nn.BatchNorm2d(32)

self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
self.bn2 = nn.BatchNorm2d(64)

self.pool1 = nn.MaxPool2d(2, 2)

self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
self.bn3 = nn.BatchNorm2d(128)

self.pool2 = nn.MaxPool2d(2, 2)

self.fc1 = nn.Linear(128*7*7, 256)
self.drop = nn.Dropout(0.4)
self.fc2 = nn.Linear(256, 10)
```

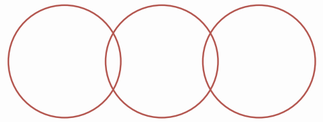
- Conv2d + BatchNorm: trích xuất đặc trưng từ ảnh, chuẩn hóa đầu ra để huấn luyện ổn định.
- MaxPool2d: giảm kích thước ảnh (downsampling), giữ đặc trưng quan trọng.
- Linear (fc1, fc2): fully connected, học đặc trưng cuối cùng để phân loại.
- Dropout: ngăn overfitting, random bỏ một số neuron trong huấn luyện.



# MNIST Model

```
x = F.relu(self.bn1(self.conv1(x)))
x = F.relu(self.bn2(self.conv2(x)))
x = self.pool1(x)

x = F.relu(self.bn3(self.conv3(x)))
x = self.pool2(x)

x = x.view(x.size(0), -1)
x = self.drop(F.relu(self.fc1(x)))
x = self.fc2(x)
return x
```

- Conv2d + BatchNorm + ReLU: trích xuất đặc trưng phi tuyến từ ảnh.
- Pooling: giảm độ phân giải ảnh, giữ thông tin quan trọng.
- Flatten: x.view(x.size(0), -1) → chuyển từ 2D feature map sang vector 1D để đưa vào Linear.
- Dropout + fc1 + ReLU: học đặc trưng ẩn với regularization.
- fc2: output logits 10 class (MNIST).

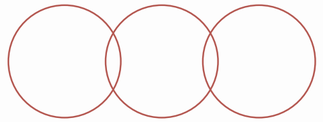


# Shapes Model

```
self.conv1 = nn.Conv2d(in_ch, 32, 3, padding=1)
self.pool = nn.MaxPool2d(2,2)
self.gap  = nn.AdaptiveAvgPool2d(1)
self.fc   = nn.Linear(128, num_classes)
```

```
x = self.pool(F.relu(self.conv1(x)))
x = self.pool(F.relu(self.conv2(x)))
x = F.relu(self.conv3(x))
x = self.gap(x).view(x.size(0), -1)
x = self.drop(x)
x = self.fc(x)
```

- Conv2d + ReLU: trích xuất đặc trưng hình học.
- MaxPool2d: giảm kích thước feature map.
- AdaptiveAvgPool2d(1): global average pooling → vector 128 chiều.
- Dropout: tránh overfitting.
- fc: fully connected → output 2 lớp (circle vs rectangle).

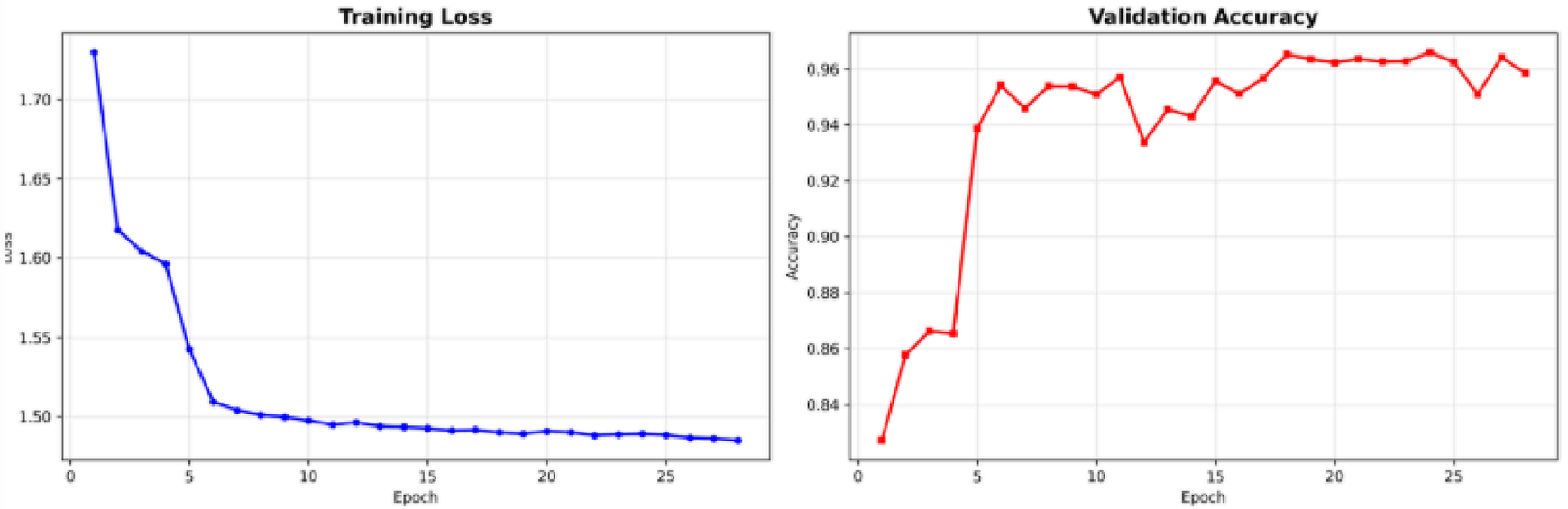


# Vì sao sử dụng mô hình CNN

- Tự động trích xuất đặc trưng từ ảnh (không cần thủ công như MLP, SVM).
- Hiệu quả cao với dữ liệu ảnh 2D nhờ lớp tích chập (Convolution).
- Giảm số tham số nhờ chia sẻ trọng số giữa các kernel.
- Pooling layer giúp mô hình bất biến với dịch chuyển và tỉ lệ ảnh.
- Huấn luyện nhanh, tránh overfitting nhờ Dropout và kiến trúc gọn nhẹ.
- Độ chính xác cao trên MNIST và Shapes, phù hợp cả bài toán nhỏ và trung bình.

# Biểu đồ huấn luyện

*Biểu đồ 1: Biểu đồ lịch sử huấn luyện*



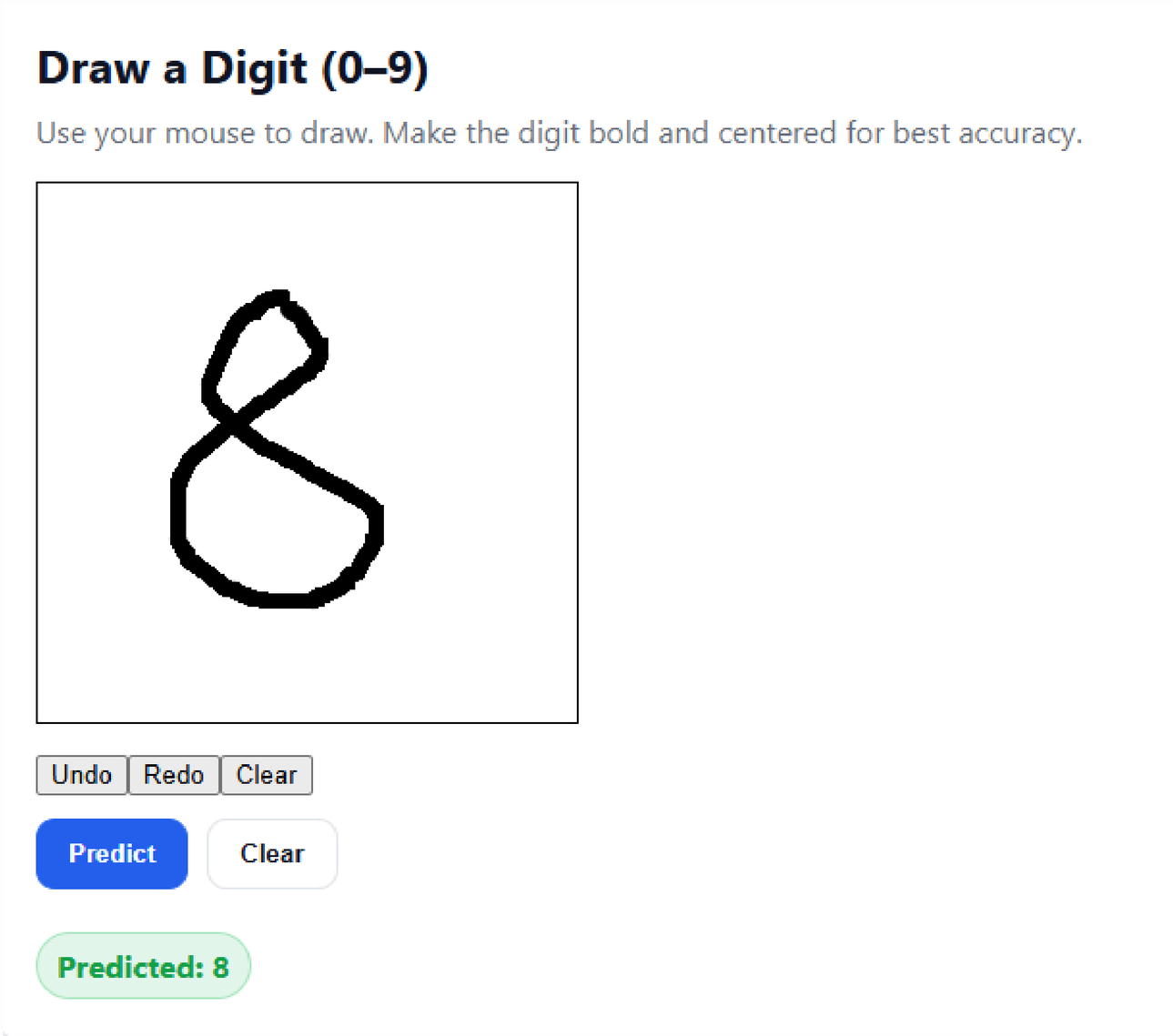

Quá trình huấn luyện hoàn tất nhanh chóng trong 4.7 phút (28 epoch) nhờ cơ chế Early Stopping, dừng sớm khi không còn cải thiện sau 10 epoch liên tiếp.

Mô hình đạt độ chính xác ấn tượng 96.53% trên tập validation và 96.30% trên tập test. Mức chênh lệch rất nhỏ (0.23%) khẳng định khả năng tổng quát hóa tốt và không xảy ra hiện tượng overfitting.

Về tốc độ hội tụ, mô hình học rất nhanh trong 5 epoch đầu (đạt 93.88%) trước khi đi vào ổn định và đạt đỉnh tại epoch 18, với hàm mất mát giảm sâu xuống 1.49

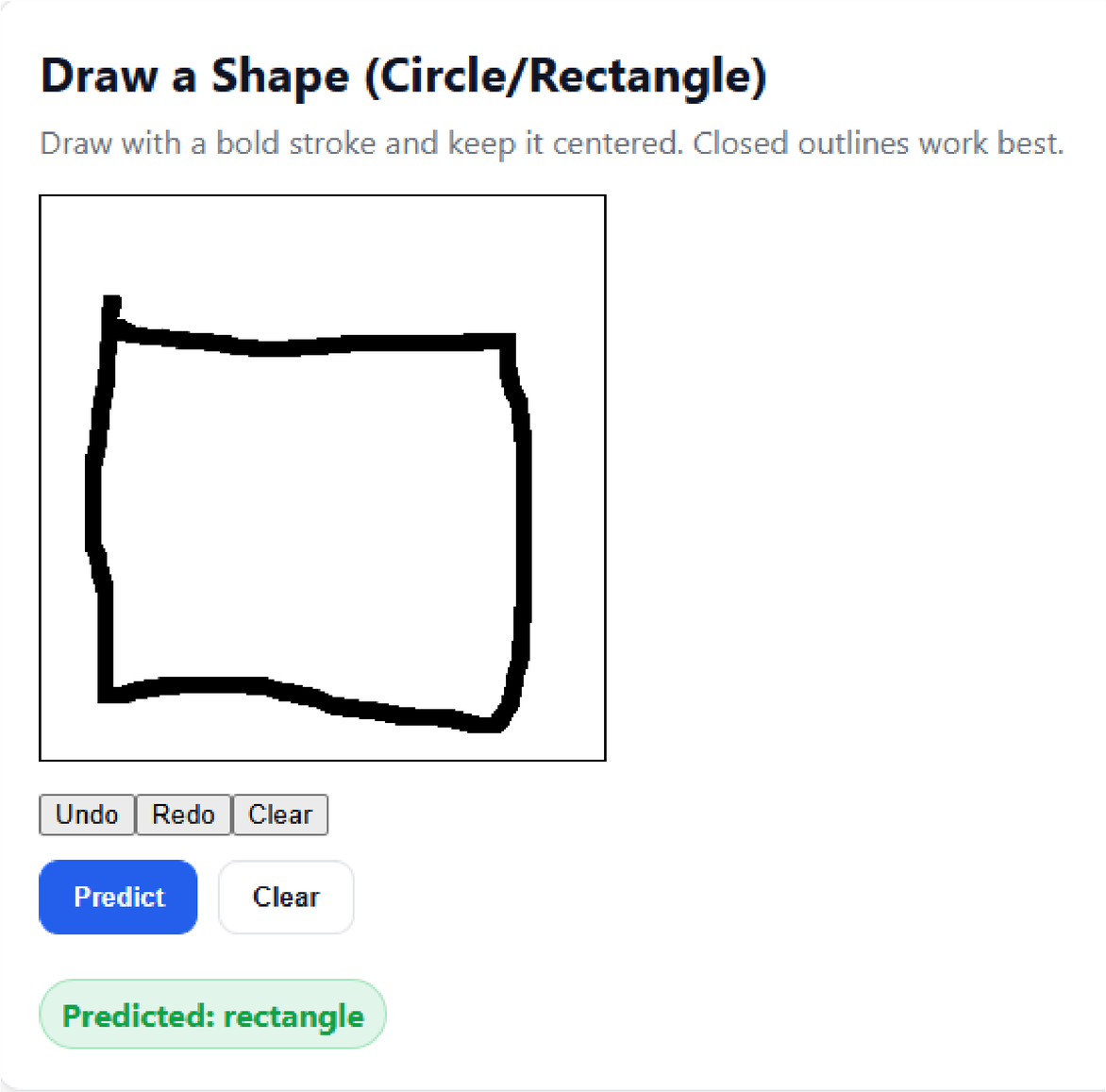
# Kết quả dự đoán

Hình ảnh dự đoán MNIST



# Kết quả dự đoán

Hình ảnh dự đoán Shapes



# Phân tích kết quả

## Đánh giá hiệu suất mô hình

Kết quả cho thấy mô hình đã học được các đặc trưng hình dạng hiệu quả, nhưng vẫn còn sai số do ảnh mờ và nhiễu, xử lý ảnh tốt giúp tăng độ chính xác đáng kể.

# Hướng phát triển

## Nhận dạng hình

Việc mở rộng khả năng nhận dạng nhiều loại hình học khác nhau sẽ giúp mô hình cải thiện độ chính xác và khả năng ứng dụng trong thực tiễn.

## OCR nâng cao

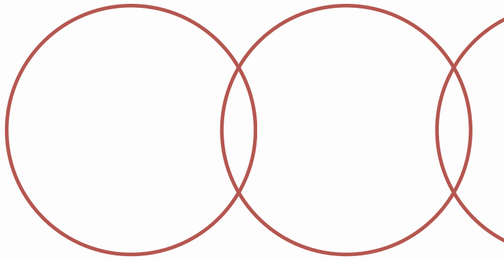
Tăng cường công nghệ nhận dạng chữ viết tay và in ấn sẽ mở ra cơ hội mới cho nhiều ứng dụng, từ giáo dục đến giám sát an ninh.

## ResNet và Transfer Learning

Sử dụng ResNet và Transfer Learning sẽ giúp tối ưu hóa quá trình huấn luyện, giảm thời gian và tài nguyên cần thiết cho các mô hình phức tạp hơn.

# Kết luận

Triển khai thành công một mạng nơ-ron từ đầu, tự xây dựng tất cả các thành phần như lớp tuyến tính, Backpropagation, ReLU, Softmax và thuật toán Adam, giúp nắm vững bản chất toán học của mô hình. Mô hình đạt hiệu suất ấn tượng với độ chính xác 96,3% trên tập kiểm tra và thời gian huấn luyện nhanh chỉ khoảng 4,7 phút.



# Cảm ơn thầy và các bạn đã lắng nghe