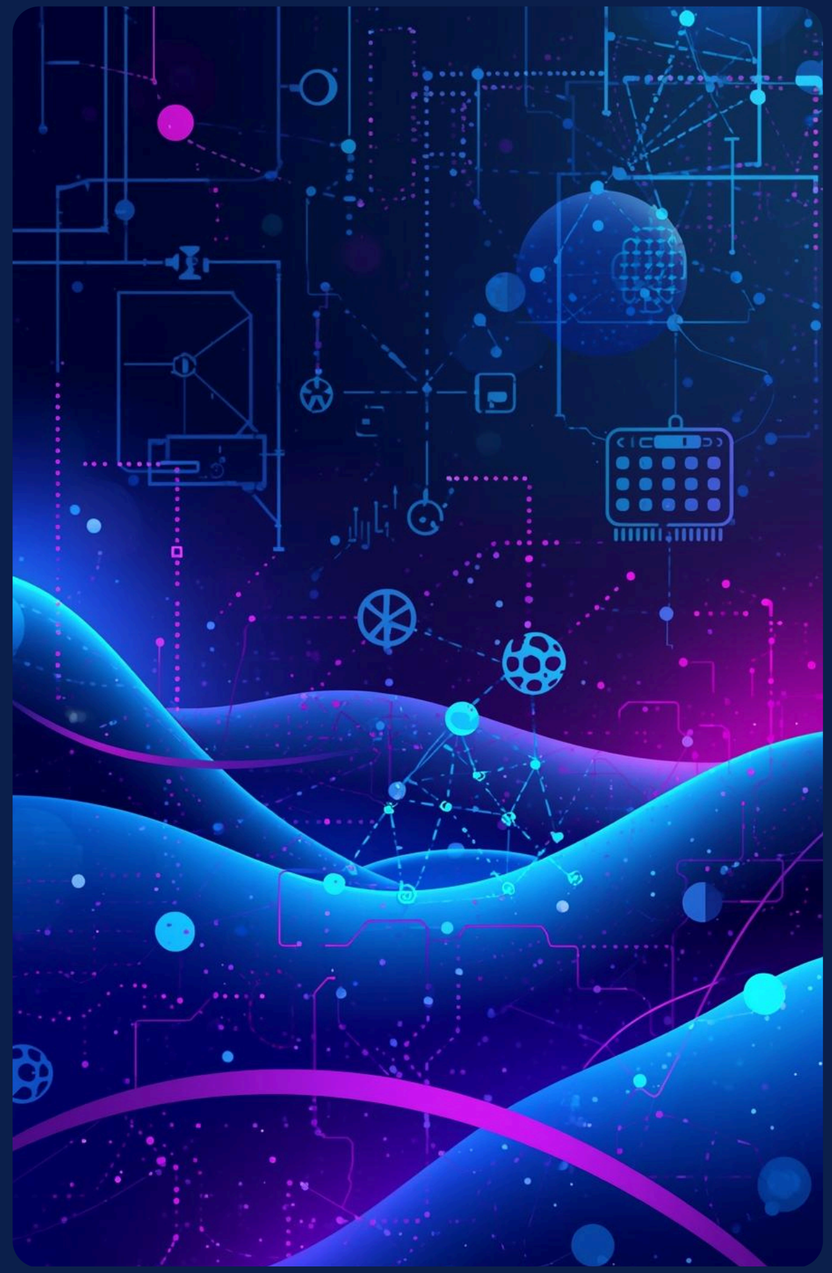
Nhận dạng chữ số viết tay và hình học cơ bản

Nguyễn Tất Thắng



Mục tiêu dự án

Tự xây dựng CNN

Mục tiêu đầu tiên là tự xây dựng một mô hình CNN bằng Python, mang lại hiểu biết sâu sắc về cấu trúc và hoạt động của mạng nơ-ron tích chập.

Nhận dạng MNIST

Dự án sẽ bao gồm việc nhận dạng các chữ số từ bộ dữ liệu MNIST, giúp cải thiện khả năng phân loại hình ảnh và hiểu biết về kỹ thuật học máy.

Nhận dạng hình dạng

Bên cạnh đó, dự án còn tập trung vào việc nhận dạng các hình tròn và chữ nhật, mở rộng ứng dụng của CNN trong việc phát hiện hình dạng cơ bản.

Một số kiến thức

01 **Convolution**

Chức năng chính của lớp này là trích xuất đặc trưng.

02 **Pooling**

Giảm kích thước đầu vào và giữ lại thông tin quan trọng.

03 **Flatten**

Chuyển đổi đầu ra của lớp trước thành vector một chiều.

04 **Activation**

Hàm kích hoạt giúp tăng cường phi tuyến tính cho mô hình.

05 **Dense**

Kết nối tất cả các neuron trong lớp trước với lớp này.

Tại sao CNN hiệu quả

01 **Học đặc trưng**

CNN học được các đặc trưng cục bộ một cách hiệu quả nhờ cấu trúc phân cấp, cho phép chúng nhận diện các mẫu ở nhiều mức độ phức tạp khác nhau, từ đó cải thiện độ chính xác tổng thể trong các bài toán phân loại.

02 **Bất biến dịch chuyển**

Tính bất biến theo dịch chuyển của CNN cho phép chúng nhận diện các đối tượng trong ảnh bất kể vị trí của chúng, giúp mô hình trở nên mạnh mẽ trong nhiều ứng dụng thực tế và cải thiện hiệu suất.

Kiến trúc tổng quan mô hình

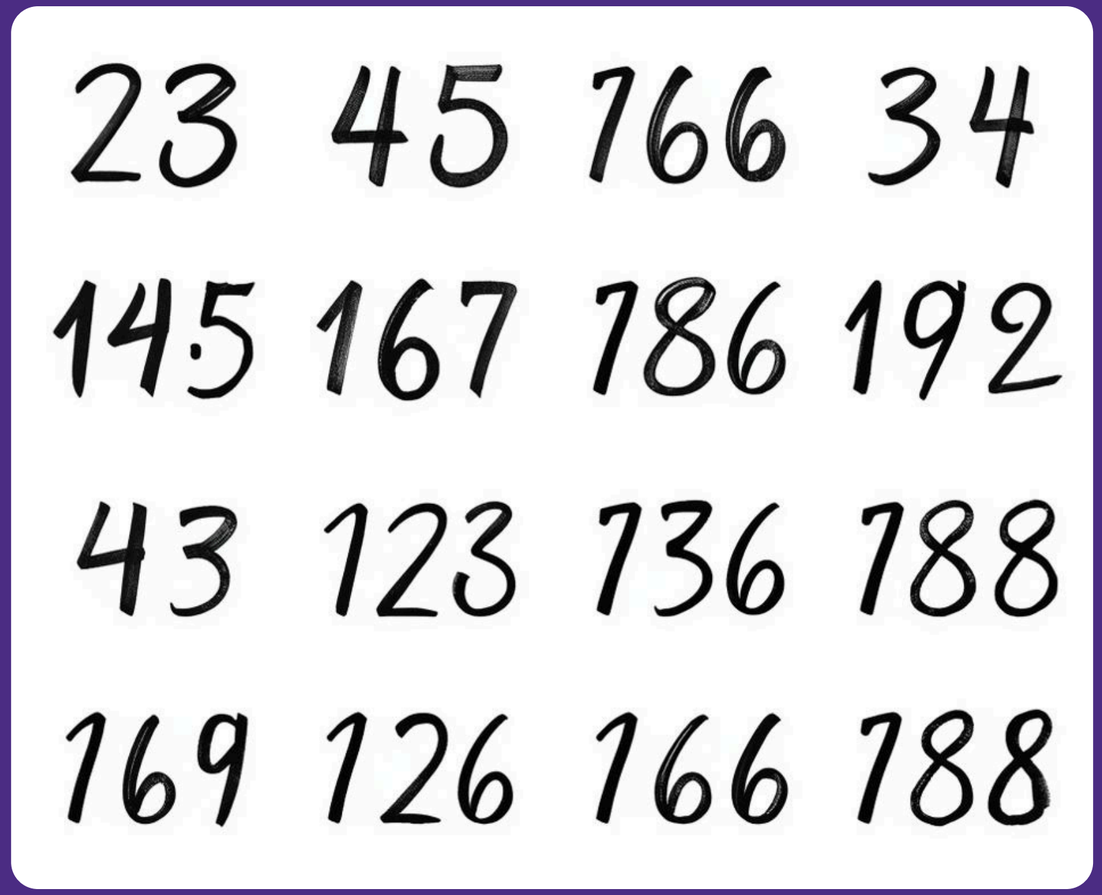
```
layers=[
    Conv2D(1,8,3,padding=1),
    ReLU(),
    MaxPool2D(2,2),

    Conv2D(8,16,3,padding=1),
    ReLU(),
    MaxPool2D(2,2),

    Flatten(),
    Dense(16*7*7,64),
    ReLU(),
    Dense(64,10)
]
```

Nhận dạng chữ số viết tay

```
layers=[
    Conv2D(1,8,3,padding=1),
    ReLU(),
    MaxPool2D(2,2),

    Conv2D(8,16,3,padding=1),
    ReLU(),
    MaxPool2D(2,2),

    Flatten(),
    Dense(16*8*8,32),
    ReLU(),
    Dense(32,2)
]
```

Nhận dạng hình học cơ bản

Dữ liệu MNIST

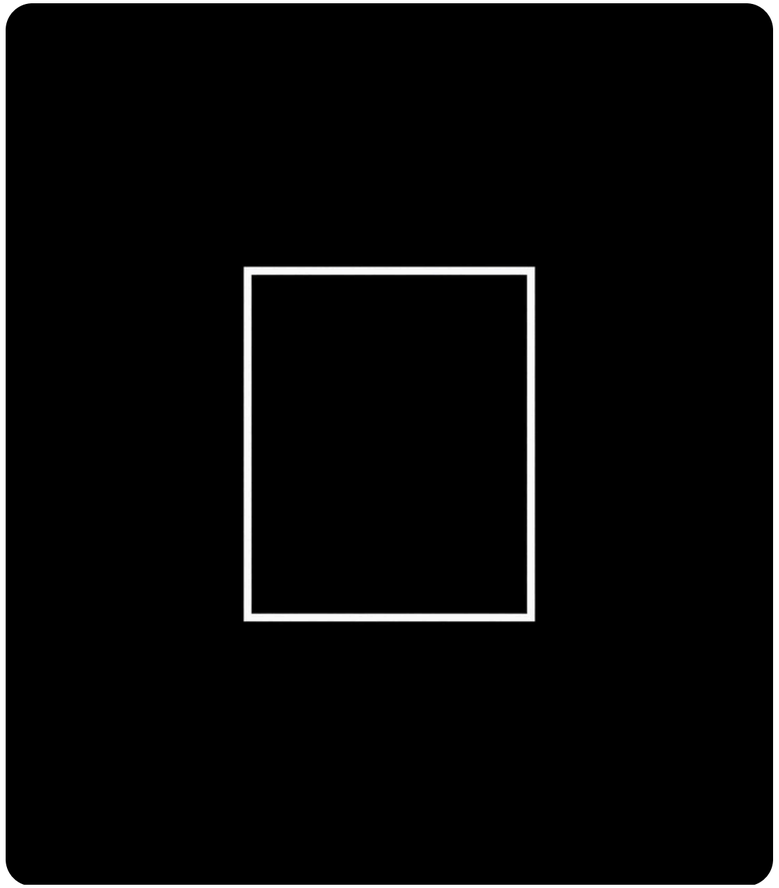



```python
def load_mnist(root="data/mnist"):
    # Xác định đường dẫn các file MNIST chuẩn IDX
    paths = {
        "train_images": os.path.join(root, "train-images.idx3-ubyte"),
        "train_labels": os.path.join(root, "train-labels.idx1-ubyte"),
        "test_images":  os.path.join(root, "t10k-images.idx3-ubyte"),
        "test_labels":  os.path.join(root, "t10k-labels.idx1-ubyte"),
    }

    # Đọc dữ liệu train/test
    X_train = _read_idx_images(paths["train_images"])
    y_train = _read_idx_labels(paths["train_labels"])
    X_test  = _read_idx_images(paths["test_images"])
    y_test  = _read_idx_labels(paths["test_labels"])

    # Trả về tuple giống kiểu MNIST trong PyTorch
    return (X_train, y_train), (X_test, y_test)
```

Dữ liệu hình học cơ bản

```python
def gen_shapes_dataset(n_samples=6000, img_size=32, noise=0.05, seed=123, bbox=False):
    """
    Generate images containing either rectangles or circles.
    Output:
      X: (N,1,H,W) ảnh grayscale đã chuẩn hóa
      y: nhãn (0=rectangle, 1=circle)
      B: bounding box chuẩn hóa (tùy chọn)
    """
```

```python
X, y, _ = gen_shapes_dataset(n_samples=args.samples, seed=args.seed)
n = int(len(X) * (1 - args.val_split))
X_tr, y_tr = X[:n], y[:n]
X_val, y_val = X[n:], y[n:]
X_te2, y_te2 = X[n:], y[n:]
```



Chi tiết cài đặt các lớp neural

```python
# Lớp base cho mọi layer
class Layer:
    def forward(self, x, train=True): raise NotImplementedError
    def backward(self, grad): raise NotImplementedError
    def params_and_grads(self): return []
```

Convolution Layer (Conv2D)

```python
def __init__(self, in_channels, out_channels, kernel_size, s
    self.in_c = in_channels
    self.out_c = out_channels

    # kernel_size có thể là int hoặc tuple
    if isinstance(kernel_size, int):
        kh, kw = kernel_size, kernel_size
    else:
        kh, kw = kernel_size
    self.kh, self.kw = kh, kw

    self.stride = stride
    self.pad = padding

    # Khởi tạo trọng số bằng He-init
    self.W = he_init((out_channels, in_channels, kh, kw))
    self.b = np.zeros((out_channels,), dtype=np.float32)
    self.cache = None
```

```python
def params_and_grads(self):
    return [(self.W, getattr(self, "dW", None)),
            (self.b, getattr(self, "db", None))]
```

Convolution Layer (Conv2D)

```python
def forward(self, x, train=True):
    self.x_shape = x.shape

    # im2col: chuyển ảnh thành ma trận để tích
    chập thành phép nhân ma trận
    cols, out_h, out_w = im2col(x, self.kh, self.
    kw, self.stride, self.pad)
    self.cols = cols

    # Trải kernel thành 2D để nhân với cols
    W_col = self.W.reshape(self.out_c, -1)

    # Tính output = cols @ W + bias
    out = cols @ W_col.T + self.b

    # Chuyển shape về (N, C, H, W)
    out = out.reshape(x.shape[0], out_h, out_w,
    self.out_c).transpose(0, 3, 1, 2)
    self.out_shape = (out_h, out_w)
    return out
```

```python
def backward(self, grad):
    N = grad.shape[0]

    # Chuyển grad về dạng 2D khớp với cols
    grad_reshaped = grad.transpose(0, 2, 3, 1).
    reshape(-1, self.out_c)

    # Gradient trọng số và bias
    dW = grad_reshaped.T @ self.cols
    db = grad_reshaped.sum(axis=0)

    # Tính gradient đầu vào
    W_col = self.W.reshape(self.out_c, -1)
    dcols = grad_reshaped @ W_col
    dx = col2im(dcols, self.x_shape, self.kh, self.
    kw,
                self.out_shape[0], self.out_shape
                [1],
                self.stride, self.pad)

    self.dW = dW.reshape(self.W.shape)
    self.db = db
    return dx
```

Layer Pooling (MaxPool2D)

```python
def forward(self, x, train=True):
    N, C, H, W = x.shape
    out_h = (H - self.kh) // self.stride + 1
    out_w = (W - self.kw) // self.stride + 1

    # reshape để dùng im2col
    x_reshaped = x.reshape(N * C, 1, H, W)
    cols, _, _ = im2col(x_reshaped, self.kh, self.
    kw, self.stride, 0)
    self.cols = cols

    # Lấy vị trí max index (quan trọng cho
    backward)
    self.arg_max = np.argmax(cols, axis=1)

    out = cols[np.arange(cols.shape[0]), self.
    arg_max].reshape(N, C, out_h, out_w)
    self.x_shape = x.shape
    self.out_hw = (out_h, out_w)
    return out
```

```python
def backward(self, grad):
    N, C, out_h, out_w = grad.shape

    # Khởi tạo gradient theo cols
    dcols = np.zeros_like(self.cols)

    # Gán gradient đúng vị trí max
    dcols[np.arange(dcols.shape[0]), self.
    arg_max] = grad.reshape(-1)

    # Chuyển ngược từ col về ảnh
    dx = col2im(dcols,
                (N * C, 1, self.x_shape[2], self.
                x_shape[3]),
                self.kh, self.kw,
                out_h, out_w, self.stride, 0)
    return dx.reshape(self.x_shape)
```

Dense Layer (Dense)

```python
class Dense(Layer):
    def __init__(self, in_features, out_features):
        self.W = he_init((in_features, out_features))
        self.b = np.zeros((out_features,), dtype=np.
        float32)

    def forward(self, x, train=True):
        self.x = x
        return x @ self.W + self.b

    def backward(self, grad):
        # Gradient của W và b
        self.dW = self.x.T @ grad
        self.db = grad.sum(axis=0)

        # Truy gradient về input
        return grad @ self.W.T

    def params_and_grads(self):
        return [(self.W, getattr(self, "dW", None)),
                (self.b, getattr(self, "db", None))]
```

Thuật toán huấn luyện

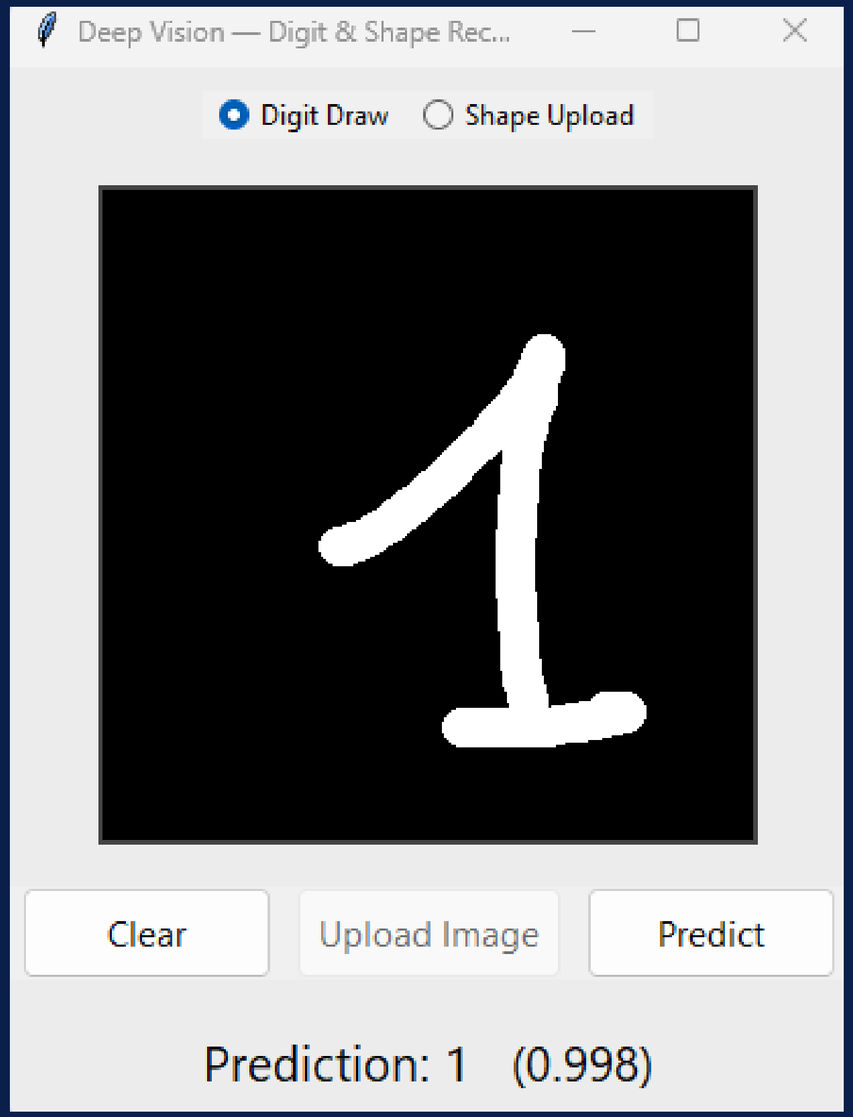
Hệ thống học sâu sử dụng **softmax cross-entropy loss** để tối ưu hóa hiệu suất, kết hợp với **gradient descent** nhằm giảm thiểu sai số qua từng bước huấn luyện, tạo ra mô hình chính xác hơn.

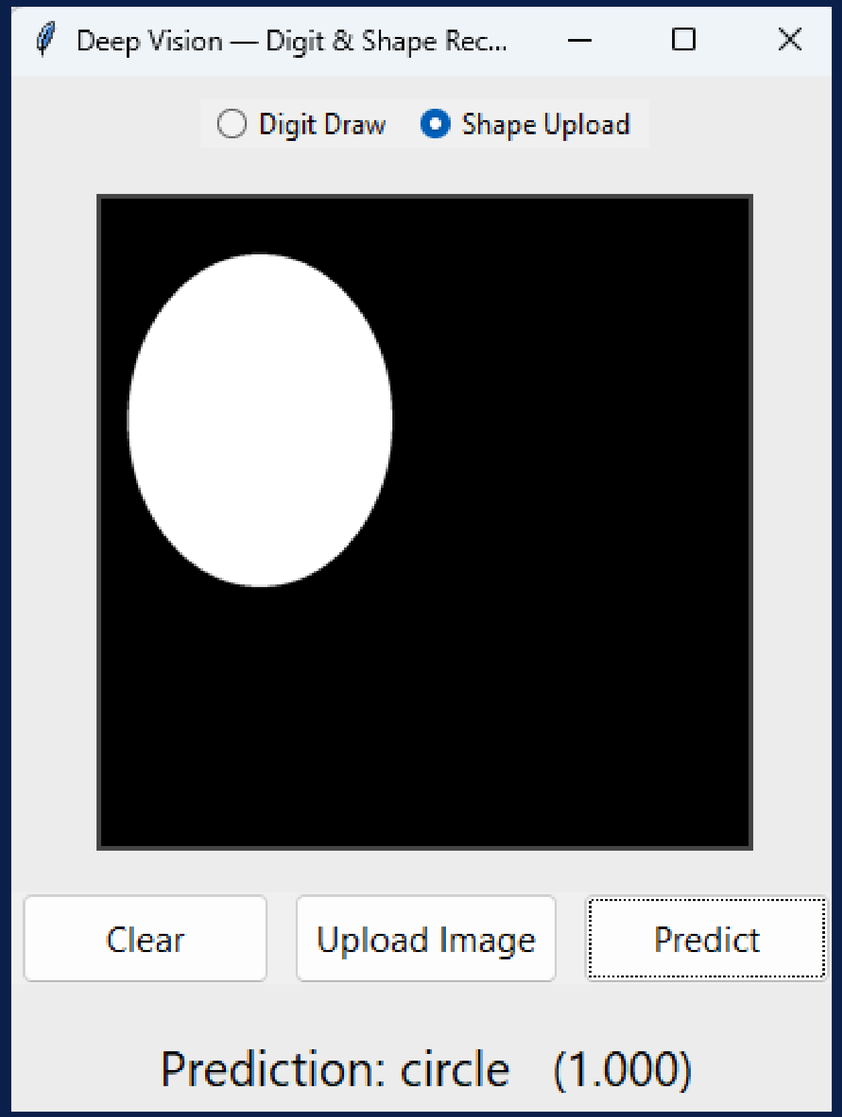
```python
class SoftmaxCrossEntropy:
    def __init__(self):
        self.probs = None
        self.y = None

    def forward(self, logits, y):
        # Trừ max để tránh overflow khi exp
        logits = logits - logits.max(axis=1,
        keepdims=True)
        # Softmax
        exp = np.exp(logits)
        probs = exp / exp.sum(axis=1, keepdims=True)
        self.probs = probs

        # Lưu nhãn thật
        self.y = y.astype(int)

        # Tính loss function
        N = y.shape[0]
        loss = -np.log(probs[np.arange(N), self.y] +
        1e-12).mean()
        return loss

    def backward(self):
        N = self.y.shape[0]

        # grad = softmax - one_hot
        grad = self.probs.copy()
        grad[np.arange(N), self.y] -= 1.0
        grad /= N  # chia batch size
        return grad
```

Kết quả huấn luyện mô hình

```
PS D:\nguyentatthang\NttProjects\BTL\XuLyAnh\B23DCKH107> python train.py --task shapes --epochs 5
Epoch 1: loss=0.5604 | train_acc=0.6990 | val_acc=0.8917
Saved best to checkpoints/shapes_best.npz
Epoch 2: loss=0.1046 | train_acc=0.9706 | val_acc=0.9983
Saved best to checkpoints/shapes_best.npz
Epoch 3: loss=0.0093 | train_acc=0.9998 | val_acc=0.9983
Epoch 4: loss=0.0037 | train_acc=1.0000 | val_acc=1.0000
Saved best to checkpoints/shapes_best.npz
Epoch 5: loss=0.0021 | train_acc=1.0000 | val_acc=1.0000
Test accuracy: 1.0000
```

Giao diện tool





Kết luận

Tóm tắt và đánh giá dự án

Dự án đã hoàn thành mục tiêu đề ra, cho thấy CNN hoạt động hiệu quả trong việc nhận dạng hình ảnh. GUI chạy ổn định, mang lại trải nghiệm người dùng tốt.

Q&A

Cảm ơn đã lắng nghe!