CNN - MNIST Dataset

Overview of General CNN's

a) Forward Pass: Convolution

- Set of learnable filters (Kernels) ~ 3x3 or 5x5
- Slide convolutional operations
- Single input + stride of 1 (no padding) ⇒ 2D convolutional: slide @ position (i,j):

$$Y[i,j] = \sum_{p=0}^{F-1}\sum_{q=0}^{F-1}\left(X[i+p, j+q]\cdot W[p,q]\right)+b$$

X: input (for MNIST, it's a 28x28 matrix)
W: Filter

Multiple channels (Deeper layers means colored images):

each kernel has shape $(F, F, C_{in}) \Rightarrow \{C_{in}\}$ # input channels for multiple feature maps.

$$Y_k[i,j] = \sum_{c=1}^{C_{in}}\sum_{p=0}^{F-1}\sum_{q=0}^{F-1}\left(X_c[i+p, j+q]\cdot W_{p,q,c,k}\right)+b_k$$

b) Backpropagation via Convolution: In training, we filter weights through PDE's (partial diff. eqs.) of loss fn. $\mathcal{L}$ wrt for each weight:

$$\frac{\partial \mathcal{L}}{\partial W_{p,q,c,k}} = \sum_{i,j}\frac{\partial \mathcal{L}}{\partial Y_k[i,j]}\cdot\frac{\partial Y_k[i,j]}{\partial W_{\{p,q,c,k\}}}$$

where $X_c[i+p, j+q]$

$$\frac{\partial Y_k[i,j]}{\partial W_{\{p,q,c,k\}}} = X_c[i+p, j+q]$$

Bias term:

$$\frac{\partial \mathcal{L}}{\partial b_k} = \sum_{i,j}\frac{\partial \mathcal{L}}{\partial Y_k[i,j]}$$

c) Activation Functions: Usually ReLU (Rectified Linear Unit)

$$\text{ReLU}(z) = \max(0, z)$$

↳ Each element $Y[i,j,k]$ from convolution is passed through ReLU to obtain

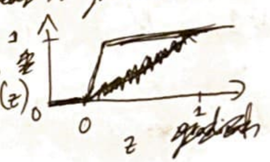
$$A_k[i,j] = \max(0, Y_k[i,j])$$

Backpropagation

$$\frac{\partial \mathcal{L}}{\partial Y_k[i,j]} = \begin{cases}\frac{\partial \mathcal{L}}{\partial A_k[i,j]} & \text{if } Y_k[i,j] \geq 0\\ 0 & \text{if } Y_k[i,j] < 0\end{cases}$$

ReLU(z) = piecewise linear ∴ gradient = 1 for $z > 0$ and 0 for $z < 0$:

(similar to fourier xform.)



d) Pooling Layers [Max Pooling]: W/ convolution + ReLU, CNNs use pooling layers to reduce spatial dimensions ∴ the most common form is a 2x2 pool size which is a moving box "striding" or square box which formulates a single output in the pooled map. This decreases variance when translating data.

$$\frac{\partial \mathcal{L}}{\partial A[i', j']} = \begin{cases}\frac{\partial \mathcal{L}}{\partial P[i,j]} & \text{if } A[i', j'] \\ 0 & \end{cases}$$

Equation of pooling: $P[i,j] = \max_{(p,q)}$ ...

Loss Functions

$$P[i,j] = \max_{(p,q)\in\Omega} \left(A[i\cdot 2+p,\ j\cdot 2+q] \right)$$

Ω = local 2×2 region in activation map A

Backpropagation through max pooling:

$$\frac{d\mathcal{L}}{dA[i',j']} = \begin{cases} \frac{d\mathcal{L}}{dP[i,j]} & \text{, if } A[i',j'] \text{ is max in 2×2 region} \\ 0 & \text{, if else} \end{cases}$$

d) Fully Connected (Dense) Layers

$$z_m = \sum_{d=1}^{D} w_{m,d} h_d + b_m, \quad \text{for } m = 1, \ldots, M$$

With a non-linear activation (ReLU) element wise:

$$a_m = \sigma(z_m)$$

For multiple fully connected layers, the output of one becomes input to the next:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}, \quad a^{(l)} = \sigma(z^{(l)})$$

e) Output Layer + Softmax:

$$\hat{y}_k = \frac{e^{z_k}}{\sum_{j=0}^{9} e^{z_k}}, \quad \text{for } k = 0, \ldots, 9$$

[MNIST digits 0 to 9]

f) Loss Function (Cross-Entropy):

$$\mathcal{L}(\hat{y}, y) = -\sum_{j=0}^{9} y_j \cdot \ln(\hat{y}_j)$$

g) Weight Updates (Gradient Descent):

$$\theta \leftarrow \theta - \eta \frac{d\mathcal{L}}{d\theta}$$

η = learning rate and $\frac{d\mathcal{L}}{d\theta}$ → computed via backpropagation