Homework 2

Justin Hughes

February 2020

1 SIFT

The SIFT algorithm works by initially assuming that the local extrema or high value points are in the same general location after rotation. A descriptor of vectors depicts the high value points and uses this vector to compare high value points between these images. A histogram of gradient directions of key features around the high value points is calculated to give us the high value point location, scale and orientation. We then compare gradient vectors in the rotated images and those that meet a set threshold by the user help determine accurate orientation between key points in these images.

2 Negative Log Likelihood Loss

$$L = -log(p_{gt})$$

$$p_{gt} = \frac{exp(y_{gt})}{\sum_{i=1}^{n} exp(y_i)}$$

$gt \in I$

$$\frac{dp_{gt}}{dy_{gt}} = \frac{exp(y_{gt}) \sum_{i=1}^{n} exp(y_i) - exp(y_{gt})exp(y_i)}{(\sum_{i=1}^{n} exp(y_i))^2}$$

$$= \frac{exp(y_{gt})}{\sum_{i=1}^{n} exp(y_i)} - \frac{exp(y_{gt})exp(y_i)}{(\sum_{i=1}^{n} exp(y_i))^2}$$

$$= p_{gt} - p_{gt}p_i = p_{gt}(1 - p_i)$$

$$\frac{dL}{dp_{gt}} = -\frac{1}{p_{gt}}$$

$$\frac{dL}{dy_{gt}} = \frac{dL}{dp_{gt}} \frac{dp_{gt}}{y_{gt}} = -\frac{1}{p_{gt}} p_{gt}(1 - p_i) = p_i - 1$$

$gt \notin I$

$$\frac{dp_{gt}}{dy_i} = -\frac{exp(y_{gt})exp(y_i)}{(\sum_{i=1}^{n} exp(y_i))^2}$$

$$= -p_{gt}p_i$$

$$\frac{dL}{y_i} = \frac{dL}{dp_{gt}}\frac{dp_{gt}}{y_i} = -\frac{1}{p_{gt}}(-p_{gt}p_i) = p_i$$

3 Back-propagation for Fully-Connected Layer

$$y = Wx + b$$

[1] is the vector of 1 with length k

$$\frac{dL}{db} = \frac{dL}{dy}\frac{dy}{db} = \frac{dL}{dy} \cdot [1] = \frac{dL}{dy}$$

$$\frac{dL}{dW} = \frac{dL}{dy}\frac{dy}{dW} = \frac{dL}{dy} \cdot x$$

$$\frac{dL}{dx} = \frac{dL}{dy}\frac{dy}{dx} = \frac{dL}{dy} \cdot W$$

These derivations are straight forward, not much explanation is required. Parameters must match so we may have to take the transpose of matrices when appropriate.

4 Back-propagation for Convolutional Layer

$$y(k,i,j) = \sum_{t=0}^{T-1}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x(t, i(s)+m, j(s)+n)W(t,m,n) + b_k$$

[1] is the vector of 1 with length k

$$\frac{dL}{db} = \frac{dL}{dy} \cdot \frac{dy}{db} = \frac{dL}{dy_{kij}} \cdot [1]_k = \sum_{i=0}^{I}\sum_{j=0}^{J}\frac{dL}{dy_{kij}}$$

$$\frac{dL}{dW} = \frac{dL}{dy}\frac{dy}{dW} = \sum_{t=0}^{T-1}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\frac{dL}{dy} \cdot x(t, i(s)+m, j(s)+n)$$

$$\frac{dL}{dx} = \frac{dL}{dy}\frac{dy}{dx} = \sum_{\frac{m'-M+1}{s}}^{\frac{m'}{s}}\sum_{\frac{n'-N+1}{s}}^{\frac{n'}{s}}\frac{dL}{dy} \cdot W(t, m'-i(s), n'-j(s))$$

when

$$m' = i(s) + m, m = m' - i(s)$$

$$n' = j(s) + n, n = n' - j(s)$$

We simply solve for the new bounds using the old bounds. That is,

$$m = 0, i = \frac{m'}{s}$$

$$m = M - 1, i = \frac{m' - M + 1}{s}$$

$$n = 0, j = \frac{n'}{s}$$

$$n = N - 1, j = \frac{n' - N + 1}{s}$$

5 Back-propagation for Max Pooling Layer

$$y(c, i, j) = Max_{m=0..M-1} Max_{n=0...N-1} x(c, i(s) + m, j(s) + n)$$

Similar to the convolutional layer we will create m' and n'.
Take i' = i(s) + m and j' = j(s) + n where the max value is found in y.

$$\frac{dL}{dx} = \frac{dL}{dy}\frac{dy}{dx} = \frac{dL}{dy} \cdot [1]_{c,i',j'}$$

where $[1]_{c,i',j'}$ is 1 in the position of the max value in the window and 0 in the rest of the positions. This is because only the max value of each window is the only factor that contributed to the outcome. An example is provided below.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

x(1, 1, 1) = $Max_{m=0..M-1} Max_{n=0...N-1} x(c, i(s) + m, j(s) + n)$
so,

$$\begin{bmatrix} c & c & c \\ c & x & c \\ c & c & c \end{bmatrix}$$

and $\frac{dy}{dx}$ is

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$