Apr 7, 2020

Deep Learning

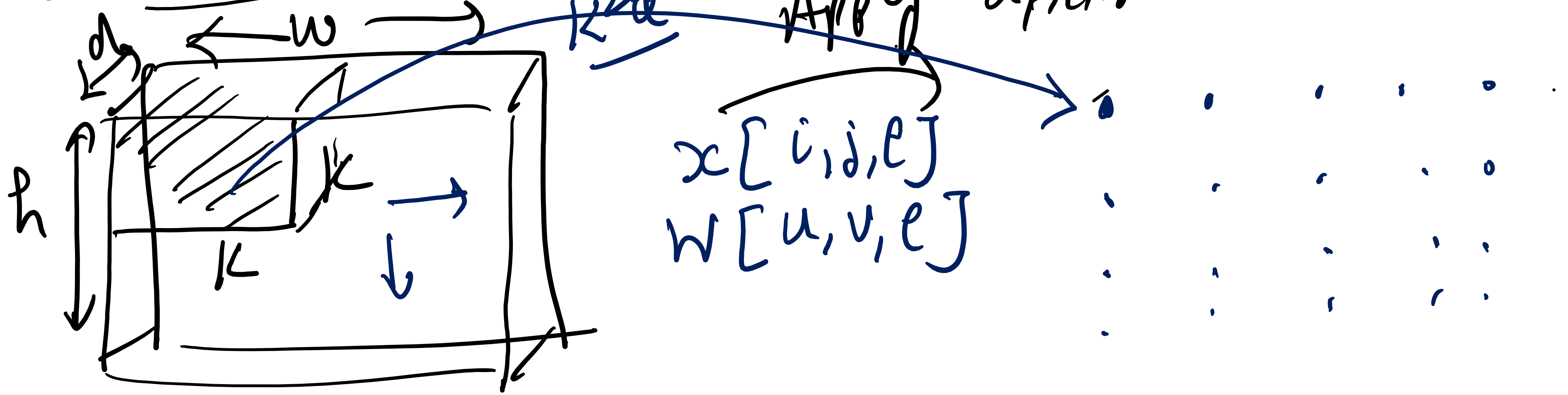
Last class:-

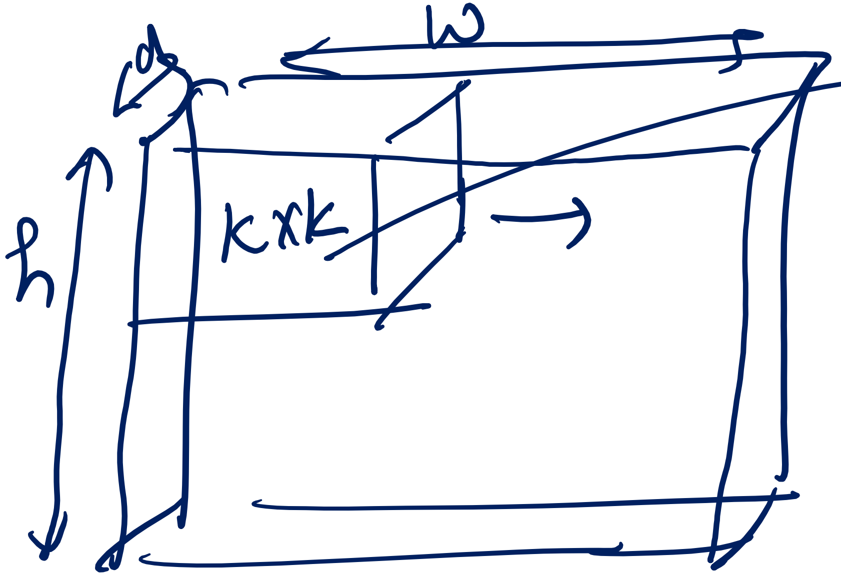
1. Motivation for deep learning { scalability / compositional
2. (a) CNN:- Convolutional Neural Networks
   (b) RNN:- Recurrent Neural Networks
   (c) GAN:- Generative Adversarial Networks

] { - ~~Self~~ Attention
  - Dropouts

Convolutional Neural Networks:-

Apply a filter





Input :- 3-Dim

$Z[i,j]$ output feature map

→ A set of operations

filter

$$z[i,j] = \sum_{l=0}^{d-1}\sum_{u=0}^{k-1}\sum_{v=0}^{k-1} x[i+u,\, j+v,\, l]\cdot \underbrace{w[u,v,l]}_{\text{tied across } [i,j]} + b$$

Assuming stride $s=1$

Resulting feature map

$$0 \le i \le \frac{(w-k)}{s} + 1$$

$$0 \le j \le \frac{(h-k)}{s} + 1$$

Zero padding

Apply non-linearity to get the final output

$$o[i,j] = g(z[i,j]) \quad \forall i,j$$

↳ sigmoid (activation fn.)

↳ Some intuition about how to backpropagate gradients in this network:-

→ think about it

(A) $\dfrac{\partial J}{\partial W[u,v,e]}$ → error metric

$$= \underbrace{\sum_{i,j}}_{} \left[\frac{\partial J}{\partial z[i,j]}\right] \cdot \left[\frac{\partial z[i,j]}{\partial W[u,v,e]}\right]$$

$\equiv \left[\dfrac{\partial J}{\partial \theta_j}\right]$ neural networks

↳ generalized chain rule

$0 \le u, v \le k$

$0 \le e \le d$

$[f(y_1, \ldots, y_k)$

$y_\ell(x_1, \ldots, x_n)$

$$\frac{\partial f}{\partial x_t} = \sum_\ell \frac{\partial f}{\partial y_\ell} \cdot \frac{\partial y_\ell}{\partial x_t}$$

(B) chain rule

$$\left[\frac{\partial J}{\partial Z_t[i,j]} = \sum_{i',j'} \frac{\partial J}{\partial Z_{t+1}[i',j']} \cdot \frac{\partial Z_{t+1}[i',j']}{\partial Z_t[i,j]}\right]$$

$\Downarrow$

$\dfrac{\partial net_\ell}{\partial net_j}$

$\ell \in downNbr(j)$

$Z_t[i,j]$ index

$Z_{t+1}[i',j']$ layer $\underline{t+1}$

At layer t

(C) $$\left[\frac{\partial Z_{t+1}[i',j']}{\partial Z_t[i,j]} \quad \forall\, i,j,i',j'\right]$$

(D) $\dfrac{\partial O_t[i,j]}{\partial Z_t[i,j]} = \dfrac{\partial g(Z_t[i,j])}{\partial Z_t[i,j]} = O_t[i,j](1-O_t[i,j])$

Sigmoid

In summary:–

$Z_t[i,j]$ $Z_{t+1}[i',j']$

(A) $\dfrac{\partial J}{\partial W_t[u,v,l]}$ (expressed in terms of $\dfrac{\partial J}{\partial Z_t[i,j]}$

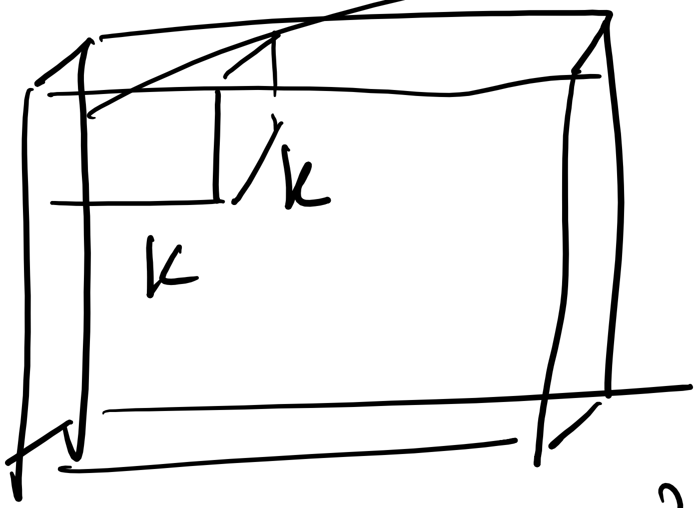
(B) $\dfrac{\partial J}{\partial Z_t[i,j]}$ expressed in terms of $\partial J / \partial Z_{t+1}[i',j']$

(C) $\dfrac{\partial Z_{t+1}[i',j']}{\partial Z_t[i,j]}$ (req. in (B))

(D) $\dfrac{\partial O_t[i,j]}{\partial Z_t[i,j]}$

"depth"

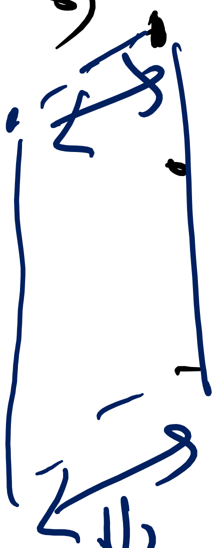
$x[i,j,\ell]$ (Linear operation) $z[i,j]$



$0 \le i < w$

$0 \le j \le h$

$0 \le \ell < d$



$k^2 d$ sized kernel (filter)

$W[u,v,\ell]$

# of kernels (Feature map)

$g(z[i,j])$

↳ activation function

⇒ Applying multiple filters (kernels)

$W_t^s[u,v,\ell]$

$s \in \{1 \dots d_{t+1}\}$

32, 64, 128

$$\frac{\partial J}{\partial Z_t[i,j,l]}$$] uniform treatment

t

Avg / Sum / max

Pooling Operation :- depth 1

Op → $g(Z_t[i,j])$

$g(Z_t[i+1,j])$

$g(Z_t[i,j])$

Deleting a part

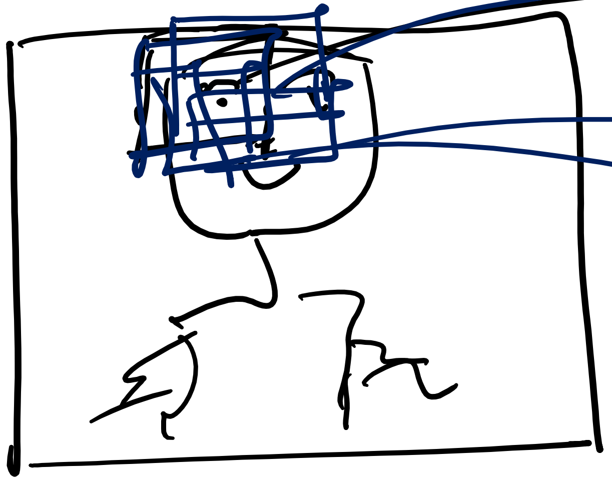
$\boxed{p=2}$

$Z_t[i', j']$

$0 \le i' < w'$

$0 \le j' < h'$

$w' = w/p, \; h' = h/p$



convolution $g(Z[i,j])$ $g(Z[i+1,j])$

$g(Z[i,j+1])$ $g(Z[i+1,j+1])$

Sum / Avg / Max

$Op(g(Z[i,j]),$
$g(Z[i+1,j]),$
$g(Z[i,j+1]),$
$g(Z[i+1,j+1]))$

✓ → Robust → less overfitting] → handle noise

✓ → scalability — $[w, h, d] \rightarrow [w/p, h/p, d]$

Reduction in size by p^2

AlexNet:- (2012)] convolutional Neural Networks

Imagenet] ~ 1 million(s) image

↳ Layers of [Convolution/pooling] followed Fully connected.