M2177.0043 Introduction to Deep Learning

Lecture 8: Neural Networks[1]

Hyun Oh Song[1]

[1] Dept. of Computer Science and Engineering, Seoul National University

April 9, 2020

[1] Many slides and figures adapted Justin Johnson

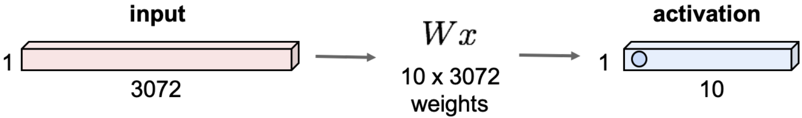
Last time

- Model fitting
- Computational Graphs
- Neural Networks

Outline

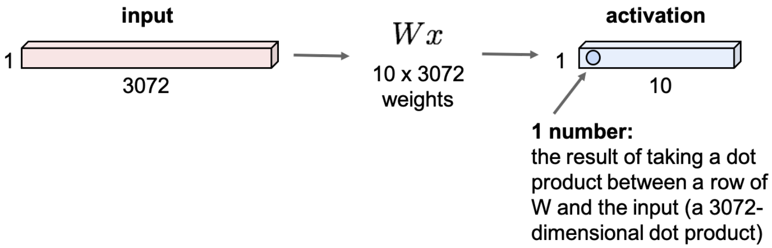
Fully connected layer

- $32 \times 32 \times 3$ image $\rightarrow$ stretch to 3072×1



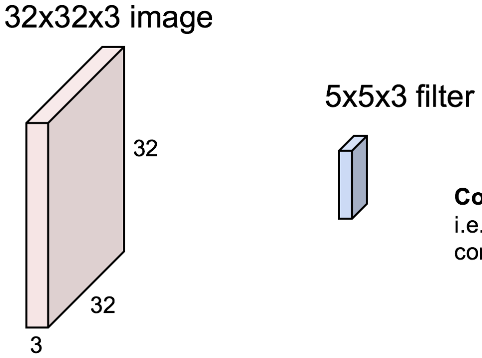
Fully connected layer

- $32 \times 32 \times 3$ image $\rightarrow$ stretch to 3072×1



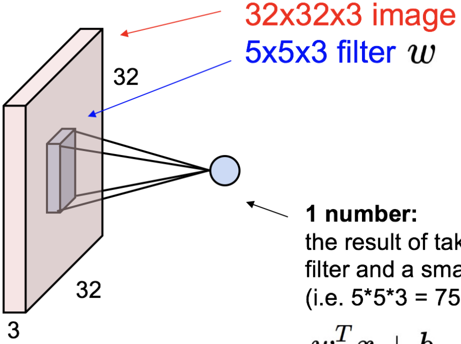
Convolution layer

- $32 \times 32 \times 3$ image $\rightarrow$ preserve spatial structure of input



Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

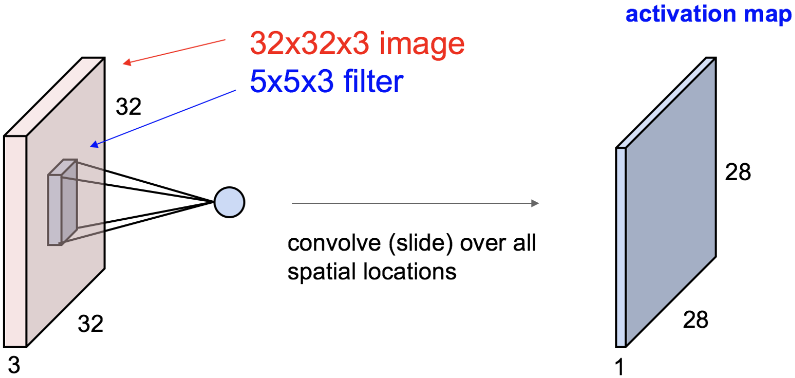
Convolution layer



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

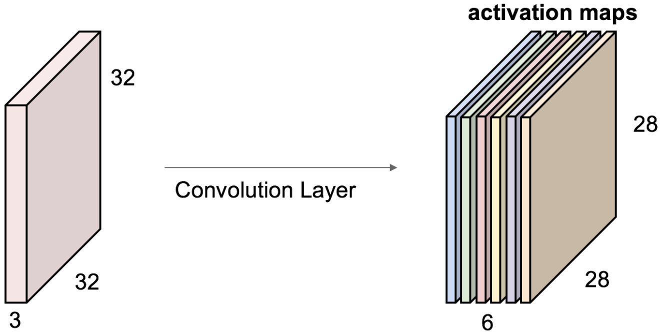
$$w^T x + b$$

Convolution layer



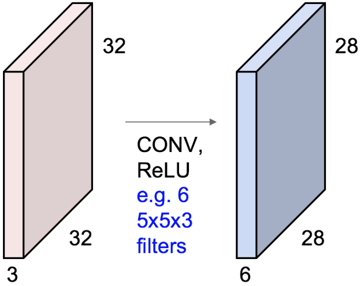
Convolution layer

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

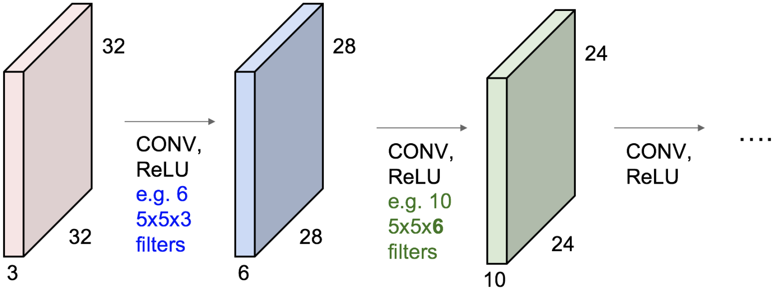


We stack these up to get a "new image" of size 28x28x6!

Convolution layer



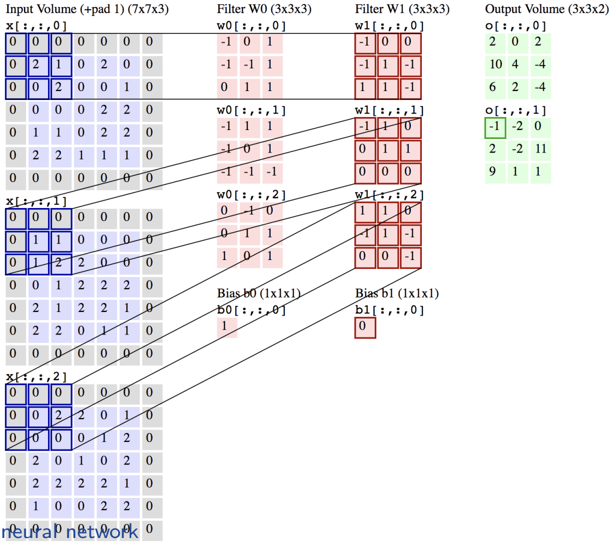
Convolution layer



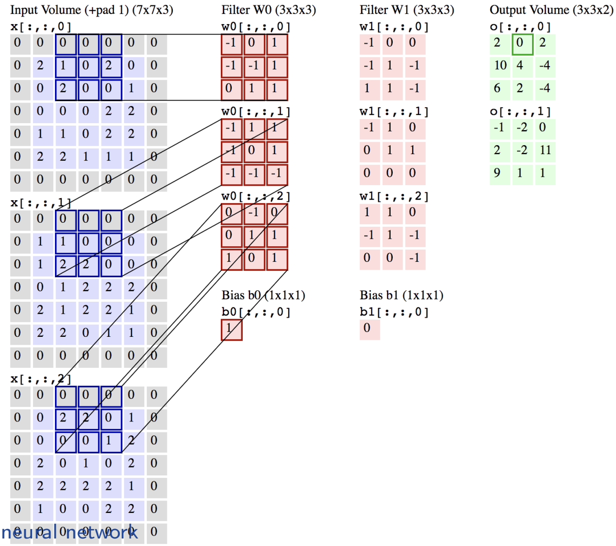
Convolution layer hyperparameters

- Number of filters. Depth of the output volume = number of activation maps.
- Stride. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.
- Zero-padding. The nice feature of zero padding is that it will allow us to control the spatial size of the output volumes

Conv layer w/ num_filters=2, stride=2, padding=1



Conv layer w/ num_filters=2, stride=2, padding=1



Conv layer w/ num_filters=2, stride=2, padding=1

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	2	1	0	2	0	0
0	0	2	0	0	1	0
0	0	0	0	2	2	0
0	1	1	0	2	2	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	2	2	0	0	0
0	0	1	2	2	2	0
0	2	1	2	2	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	2	2	0	1	0
0	0	0	0	1	2	0
0	2	0	1	0	2	0
0	2	2	2	2	1	0
0	1	0	0	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
-1	-1	1
0	1	1

w0[:,:,1]

-1	1	1
-1	0	1
-1	-1	-1

w0[:,:,2]

0	-1	0
0	1	1
1	0	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	0	0
-1	1	-1
1	1	-1

w1[:,:,1]

-1	1	0
0	1	1
0	0	0

w1[:,:,2]

1	1	0
-1	1	-1
0	0	-1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	0	2
10	4	-4
6	2	-4

o[:,:,1]

-1	-2	0
2	-2	11
9	1	1

Conv layer w/ num_filters=2, stride=2, padding=1

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	2	1	0	2	0	0
0	0	2	0	0	1	0
0	0	0	0	2	2	0
0	1	1	0	2	2	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	2	2	0	0	0
0	0	1	2	2	2	0
0	2	1	2	2	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	2	2	0	1	0
0	0	0	0	1	2	0
0	2	0	1	0	2	0
0	2	2	2	2	1	0
0	1	0	0	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
-1	-1	1
0	1	1

w0[:,:,1]

-1	1	1
-1	0	1
-1	-1	-1

w0[:,:,2]

0	-1	0
0	1	1
1	0	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	0	0
-1	1	-1
1	1	-1

w1[:,:,1]

-1	1	0
0	1	1
0	0	0

w1[:,:,2]

1	1	0
-1	1	-1
0	0	-1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

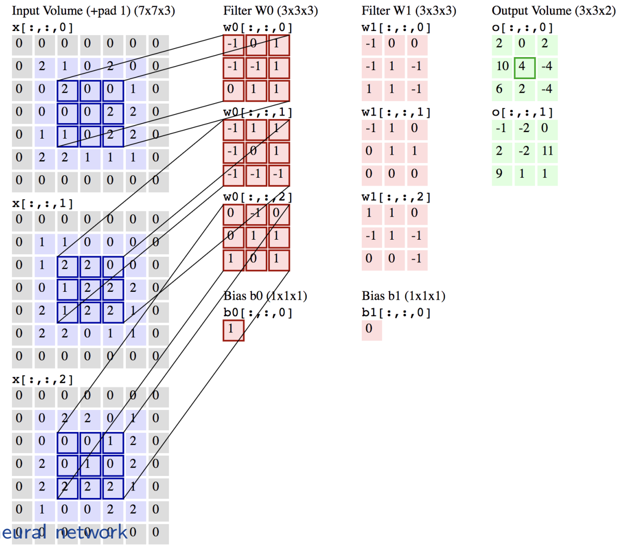
o[:,:,0]

2	0	2
10	4	-4
6	2	-4

o[:,:,1]

-1	-2	0
2	-2	11
9	1	1

Conv layer w/ num_filters=2, stride=2, padding=1



Conv layer w/ num_filters=2, stride=2, padding=1

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	2	1	0	2	0	0
0	0	2	0	0	1	0
0	0	0	0	2	2	0
0	1	1	0	2	2	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	2	2	0	0	0
0	0	1	2	2	2	0
0	2	1	2	2	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	2	2	0	1	0
0	0	0	0	1	2	0
0	2	0	1	0	2	0
0	2	2	2	2	1	0
0	1	0	0	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
-1	-1	1
0	1	1

w0[:,:,1]

-1	1	1
-1	0	1
-1	-1	-1

w0[:,:,2]

0	-1	0
0	1	1
1	0	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	0	0
-1	1	-1
1	1	-1

w1[:,:,1]

-1	1	0
0	1	1
0	0	0

w1[:,:,2]

1	1	0
-1	1	-1
0	0	-1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	0	2
10	4	-4
6	2	-4

o[:,:,1]

-1	-2	0
2	-2	11
9	1	1

Conv layer w/ num_filters=2, stride=2, padding=1

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	2	1	0	2	0	0
0	0	2	0	0	1	0
0	0	0	0	2	2	0
0	1	1	0	2	2	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	2	2	0	0	0
0	0	1	2	2	2	0
0	2	1	2	2	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	2	2	0	1	0
0	0	0	0	1	2	0
0	2	0	1	0	2	0
0	2	2	2	2	1	0
0	1	0	0	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
-1	-1	1
0	1	1

w0[:,:,1]

-1	1	1
-1	0	1
-1	-1	-1

w0[:,:,2]

0	-1	0
0	1	1
1	0	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	0	0
-1	1	-1
1	1	-1

w1[:,:,1]

-1	1	0
0	1	1
0	0	0

w1[:,:,2]

1	1	0
-1	1	-1
0	0	-1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	0	2
10	4	-4
6	2	-4

o[:,:,1]

-1	-2	0
2	-2	11
9	1	1

Conv layer w/ num_filters=2, stride=2, padding=1

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	2	1	0	2	0	0
0	0	2	0	0	1	0
0	0	0	0	2	2	0
0	1	1	0	2	2	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	2	2	0	0	0
0	0	1	2	2	2	0
0	2	1	2	2	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	2	2	0	1	0
0	0	0	0	1	2	0
0	2	0	1	0	2	0
0	2	2	2	2	1	0
0	1	0	0	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
-1	-1	1
0	1	1

w0[:,:,1]

-1	1	1
-1	0	1
-1	-1	-1

w0[:,:,2]

0	-1	0
0	1	1
1	0	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	0	0
-1	1	-1
1	1	-1

w1[:,:,1]

-1	1	0
0	1	1
0	0	0

w1[:,:,2]

1	1	0
-1	1	-1
0	0	-1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	0	2
10	4	-4
6	2	-4

o[:,:,1]

-1	-2	0
2	-2	11
9	1	1

Conv layer w/ num_filters=2, stride=2, padding=1

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	2	1	0	2	0	0
0	0	2	0	0	1	0
0	0	0	0	2	2	0
0	1	1	0	2	2	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	2	2	0	0	0
0	0	1	2	2	2	0
0	2	1	2	2	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	2	2	0	1	0
0	0	0	0	1	2	0
0	2	0	1	0	2	0
0	2	2	2	2	1	0
0	1	0	0	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
-1	-1	1
0	1	1

w0[:,:,1]

-1	1	1
-1	0	1
-1	-1	-1

w0[:,:,2]

0	-1	0
0	1	1
1	0	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	0	0
-1	1	-1
1	1	-1

w1[:,:,1]

-1	1	0
0	1	1
0	0	0

w1[:,:,2]

1	1	0
-1	1	-1
0	0	-1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

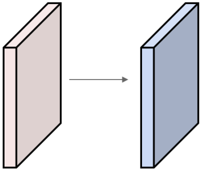
o[:,:,0]

2	0	2
10	4	-4
6	2	-4

o[:,:,1]

-1	-2	0
2	-2	11
9	1	1

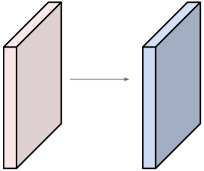
Example 1: Output activation size



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

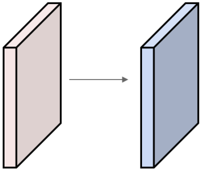
Example 1: Output activation size



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

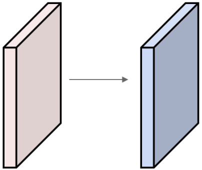
Example 2: Number of parameters



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

Example 2: Number of parameters



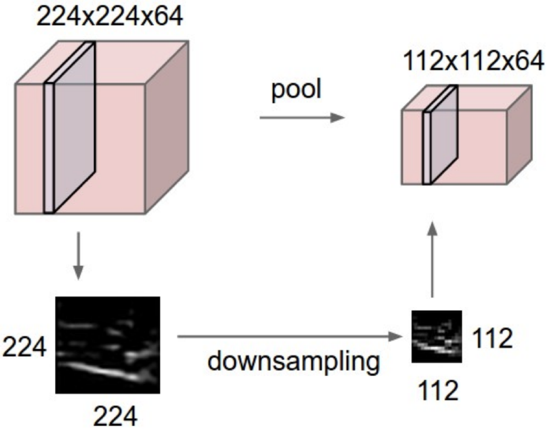
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

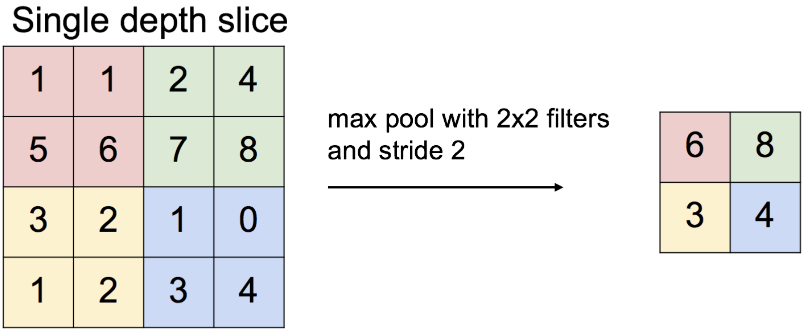
Outline

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently



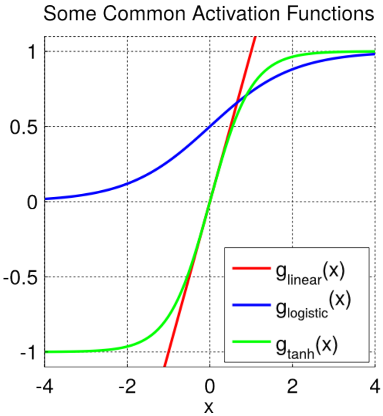
Max pooling, average pooling, etc.

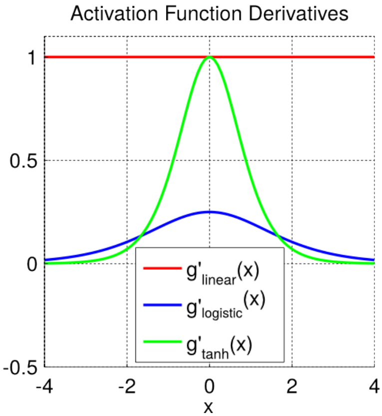


Outline

Activation functions[2]

- $g_{\text{logistic}} = 1/(1 + e^{-z})$
- $g_{\text{tanh}} = (e^z - e^{-z})/(e^z + e^{-z})$



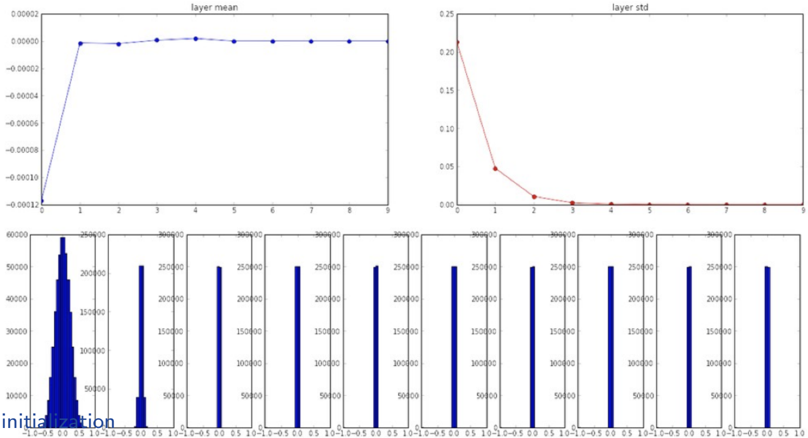


[2] https://theclevermachine.wordpress.com/tag/tanh-function/

Outline

10 Layer MLP w/ tanh: small weights

- `W = 0.01 * np.random.randn(n_in, n_out)`
- Activations get collapsed to zero. Why?

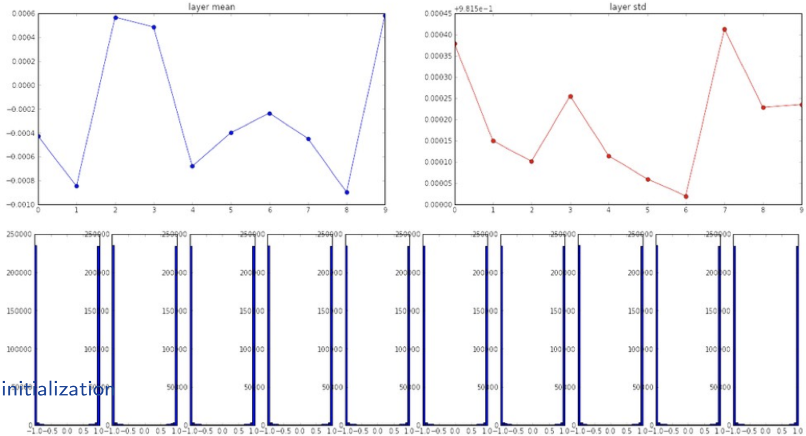


10 Layer MLP w/ tanh: small weights

- Activations get saturated at 0 and gradient doesn't flow. Why?
- Forward: Keep multiplying near zero matrices.
- Backward: Keep multiplying near zero matrices as well. Upstream gradient collapsing to zero as well.
- Activations collapse to all zeros.
- So the if the problem is because W is initialized with **small weights**, what if we initialize with **big weights instead**?

10 Layer MLP w/ tanh: large weights

- `W = 1.0 * np.random.randn(n_in, n_out)`
- Activations get saturated at -1 or 1 and gradient doesn't flow. Why?

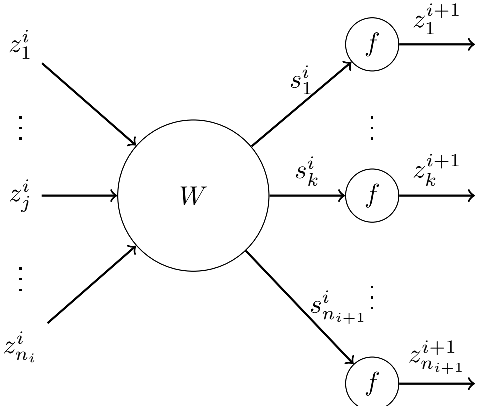


10 Layer MLP w/ tanh: large weights

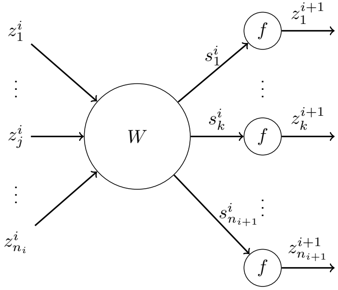
- Activations get saturated at -1 or 1 and gradient doesn't flow.
- Forward: Now, **tanh** saturates large activations to -1 or $+1$
- Backward: **tanh** has near zero gradient at -1 or $+1$ and the weights do not update

Xavier initialization

- Assume MLP network, weights and activations have zero mean, weights drawn i.i.d.
- $f(\cdot)$ is the non-linearity, centered & linear & has unit gradient around zero, *i.e.* tanh layer.



Xavier initialization - forward variance



$$z_k^{i+1} = f(s_k^{i+1}) = f(\sum_{j=1}^{n_i} W_{kj}^i z_j^i) \approx \sum_{j=1}^{n_i} W_{kj}^i z_j^i$$

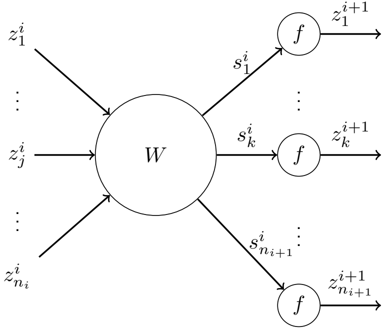
Note the property of variance and by independence ($X \perp\!\!\!\perp Y$), $Var(XY) = E[Y]^2 Var(X) + E[X]^2 Var(Y) + Var(X)Var(Y)$. Thus,

$$Var(z^{i+1}) \approx Var\left(\sum_{j=1}^{n_i} W_{kj}^i z_j^i\right)$$

$$\implies Var(z^{i+1}) = n_i Var(W^i) Var(z^i)$$

To achieve $Var(z^i) = Var(z^{i+1})$, set $Var(W^i) = \frac{1}{n_i}$

Xavier initialization - backward variance



$$\frac{\partial \ell}{\partial z_j^i} = \sum_{k=1}^{n_{i+1}} \overbrace{\frac{\partial \ell}{\partial z_k^{i+1}}}^{\text{upstream grad}} \overbrace{\frac{\partial z_k^{i+1}}{\partial z_j^i}}^{\text{local grad}}$$

$$= \sum_{k=1}^{n_{i+1}} \frac{\partial \ell}{\partial z_k^{i+1}} \underbrace{\frac{\partial z_k^{i+1}}{\partial s_k^{i+1}}}_{\approx 1} \frac{\partial s_k^{i+1}}{\partial z_j^i}$$

$$= \sum_{k=1}^{n_{i+1}} \frac{\partial \ell}{\partial z_k^{i+1}} W_{kj}^i$$

$$\implies Var\left(\frac{\partial \ell}{\partial z^i}\right) = n_{i+1} Var\left(\frac{\partial \ell}{\partial z^{i+1}}\right) Var(W^i)$$

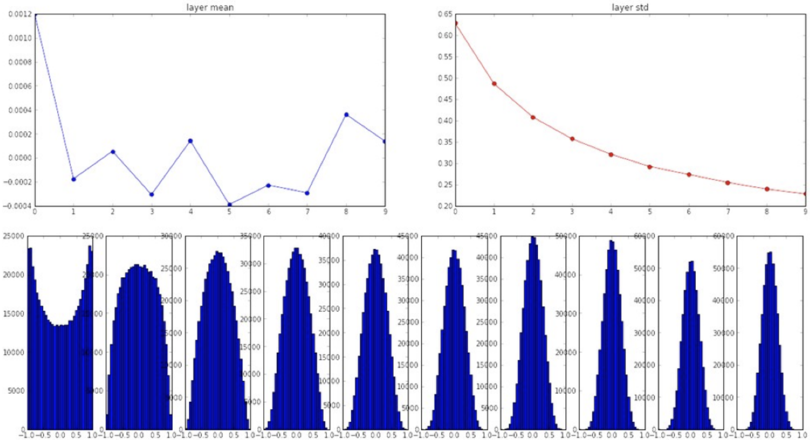
To achieve $Var(\frac{\partial \ell}{\partial z^i}) = Var(\frac{\partial \ell}{\partial z^{i+1}})$,
set $Var(W^i) = \frac{1}{n_{i+1}}$

Xavier initialization

- unless $n_i = n_{i+1}$, we have to compromise between these two conditions, and a reasonable choice is the harmonic mean, $Var(W^i) = \frac{2}{n_i + n_{i+1}}$
- Note, some implementations (Caffe) just use $Var(W^i) = \frac{1}{n_i}$
- if we sample from $N(0, 1)$, multiply the samples by $\sqrt{\frac{2}{n_i + n_{i+1}}}$.
- if we sample from $Uni(-a, a)$, take $a = \sqrt{\frac{6}{n_i + n_{i+1}}}$. Why?
- Read Glorot & Bengio 2010 for more details.

10 Layer MLP w/ tanh

- `W = np.random.randn(n_in, n_out) / np.sqrt(n_in)`



He initialization

- The assumptions do not hold for $f(\cdot)$ is ReLU activation.
- Roughly, ReLU zeroes out half the inputs.
- Set $Var(W^i) = \frac{2}{n_i}$.
- Read He *et al.* 2015 for more details.
- How to best initialize deep networks is an active research area.