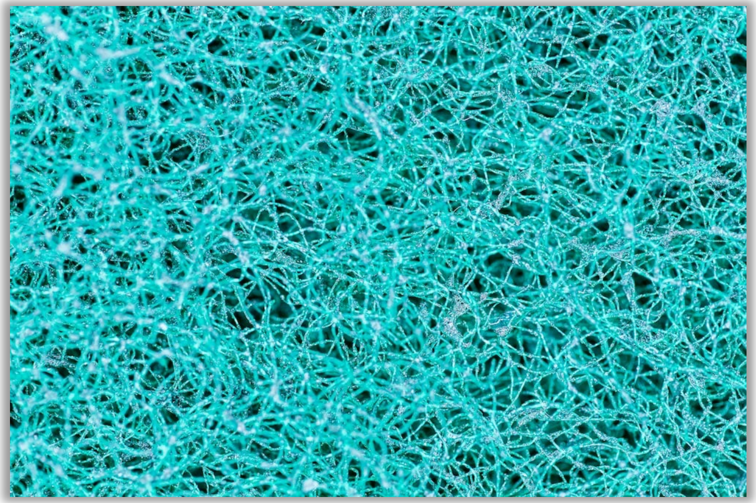
Introduction to Neural Networks

MLPs, CNNs, Backpropagation, Learned Imaging Processing



CSC2529

David Lindell

University of Toronto

cs.toronto.edu/~lindell/teaching/2529

*slides adapted from CS231n at Stanford

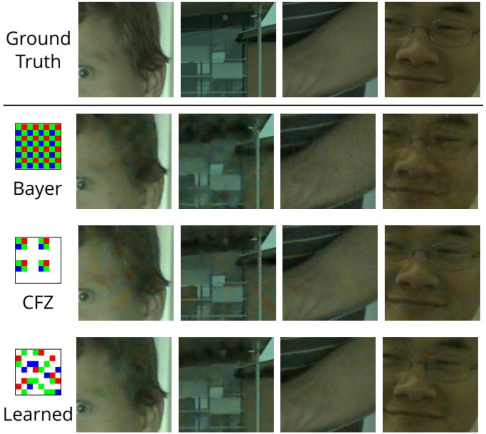
Announcements

- HW4 due Wednesday 26/10
- HW5 is out

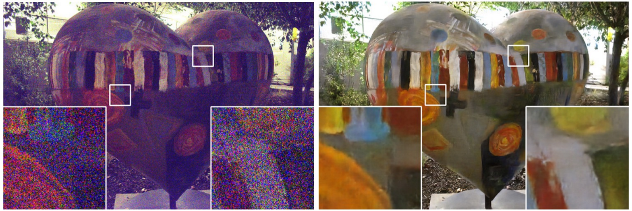
- See website for all office hours/problem session dates

Neural Networks in Computational Imaging

- Now: learned pipelines for computational imaging



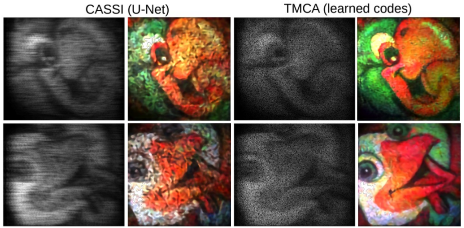
Learning CFAs



(b) Raw data via traditional pipeline

(c) Our result

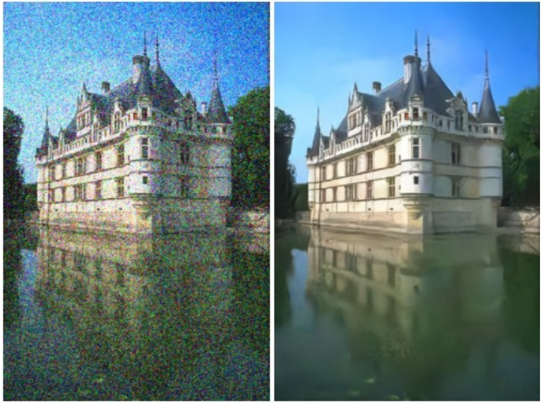
Learning ISPs



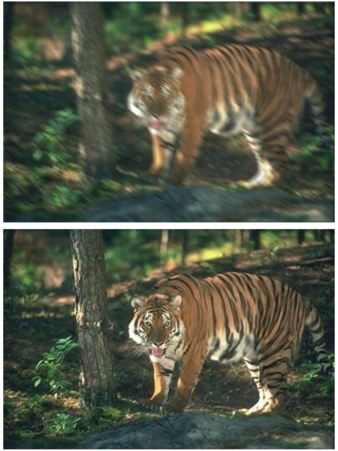
Learning coded apertures

Neural Networks in Computational Imaging

- Now: learned pipelines for computational imaging



Learned denoising



Learned deblurring

HDR Imaging

Today

- What is a neural network?
- Training/optimizing neural nets
- Why “neural”?
- Convolutional neural networks
- Applications & inverse problems

What is a neural network?

- Image classification example

Image Classification

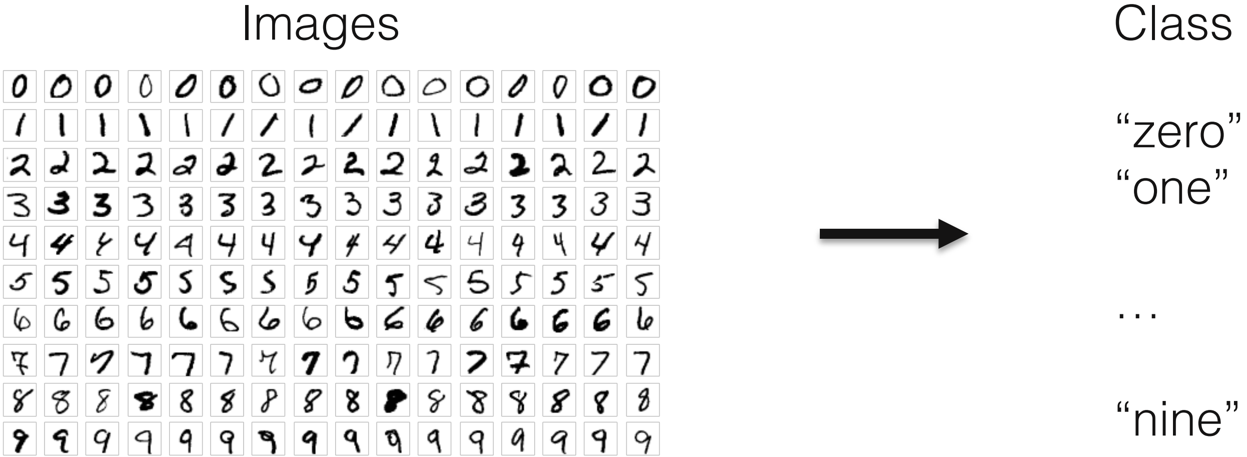
- Image classification example

Images

MNIST Dataset

Image Classification

- Image classification example



Images

Class

“zero”

“one”

…

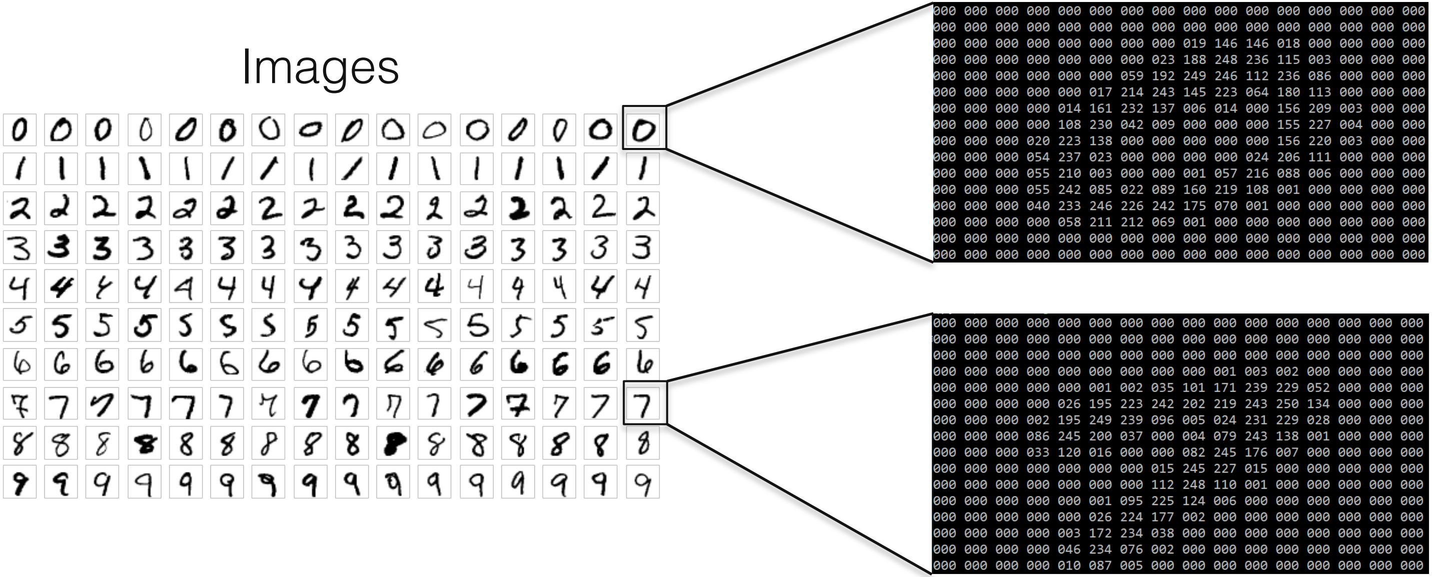
“nine”

Image Classification

- Image classification example

What the computer “sees”

Images



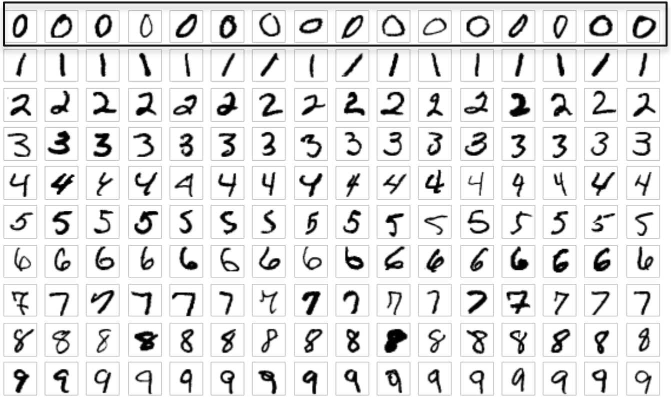
```
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000 019 146 146 018 000 000 000 000
000 000 000 000 000 000 000 023 188 248 236 115 003 000 000 000
000 000 000 000 000 000 059 192 249 246 112 236 086 000 000 000
000 000 000 000 000 017 214 243 145 223 064 180 113 000 000 000
000 000 000 000 014 161 232 137 006 014 000 156 209 003 000 000
000 000 000 000 108 230 042 009 000 000 000 155 227 004 000 000
000 000 000 020 223 138 000 000 000 000 000 156 220 003 000 000
000 000 000 054 237 023 000 000 000 000 024 206 111 000 000 000
000 000 000 055 210 003 000 000 001 057 216 088 006 000 000 000
000 000 000 055 242 085 022 089 160 219 108 001 000 000 000 000
000 000 000 040 233 246 226 242 175 070 001 000 000 000 000 000
000 000 000 000 058 211 212 069 001 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
```

```
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000 000 001 003 002 000 000 000 000
000 000 000 000 000 001 002 035 101 171 239 229 052 000 000 000
000 000 000 000 026 195 223 242 202 219 243 250 134 000 000 000
000 000 000 002 195 249 239 096 005 024 231 229 028 000 000 000
000 000 000 086 245 200 037 000 004 079 243 138 001 000 000 000
000 000 000 033 120 016 000 000 082 245 176 007 000 000 000 000
000 000 000 000 000 000 000 015 245 227 015 000 000 000 000 000
000 000 000 000 000 000 000 112 248 110 001 000 000 000 000 000
000 000 000 000 000 001 095 225 124 006 000 000 000 000 000 000
000 000 000 000 000 026 224 177 002 000 000 000 000 000 000 000
000 000 000 000 003 172 234 038 000 000 000 000 000 000 000 000
000 000 000 000 046 234 076 002 000 000 000 000 000 000 000 000
000 000 000 000 010 087 005 000 000 000 000 000 000 000 000 000
```

Image Classification

- Image classification example

Images



Challenges

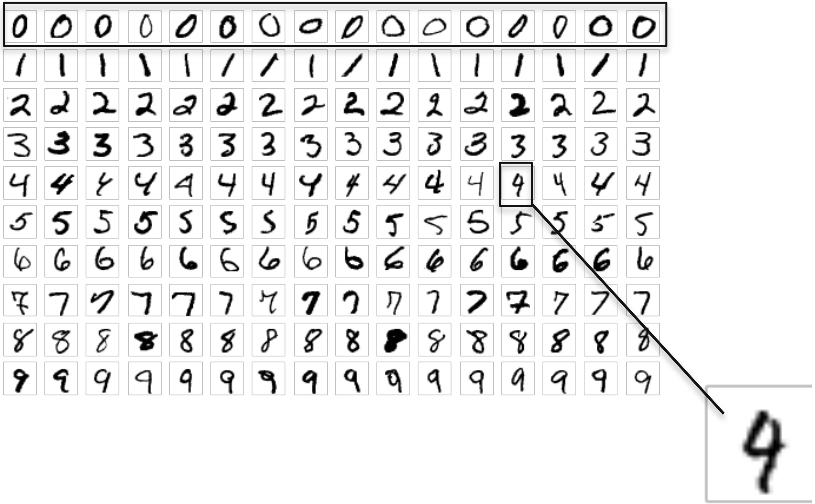
Intra-class variation

- stroke widths
- alignment
- writing styles

Image Classification

- Image classification example

Images



Challenges

Intra-class variation

- stroke widths
- alignment
- writing styles

Inter-class similarities

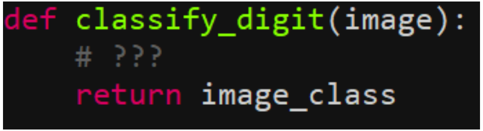
- “four” or “nine”?

Image Classification

- Image classification example

Images

Implementation?



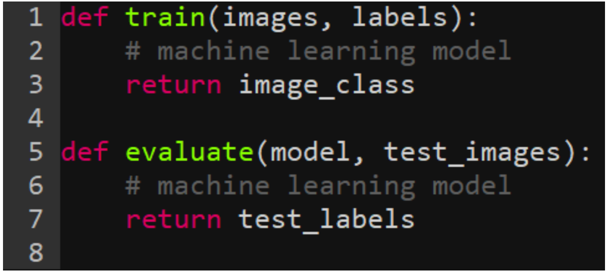
```
def classify_digit(image):
    # ???
    return image_class
```

Can’t hardcode solution!

Image Classification

- Data-driven approach
    - Collect training images and labels
    - Train a classifier using machine learning
    - Evaluate the classifier on unseen images

Implementation?



```
def train(images, labels):
    # machine learning model
    return image_class

def evaluate(model, test_images):
    # machine learning model
    return test_labels
```

Image Classification

- Linear Model

$$f(x, W) = Wx$$

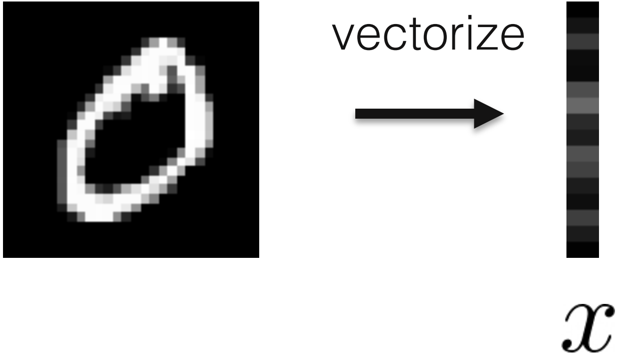


Image Classification

- Linear Model $f(x, W) = Wx$

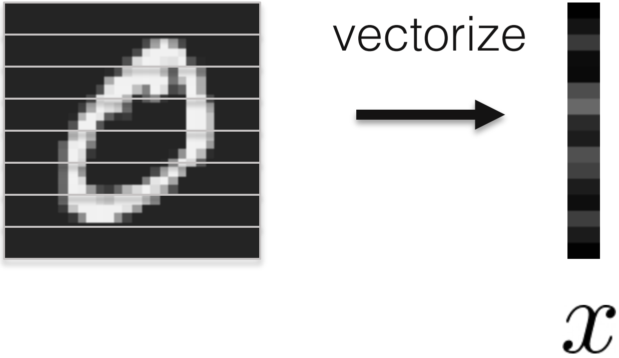
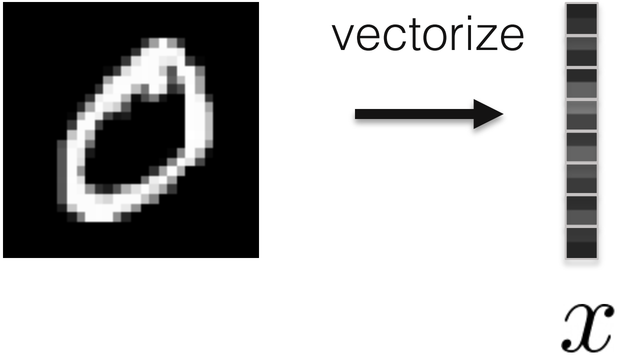


Image Classification

- Linear Model $f(x, W) = Wx$



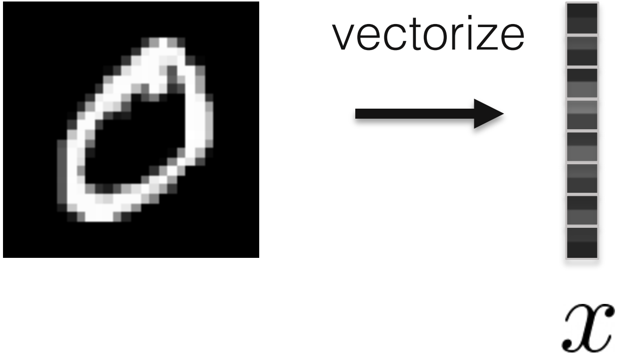
vectorize

x

Image Classification

- Linear Model

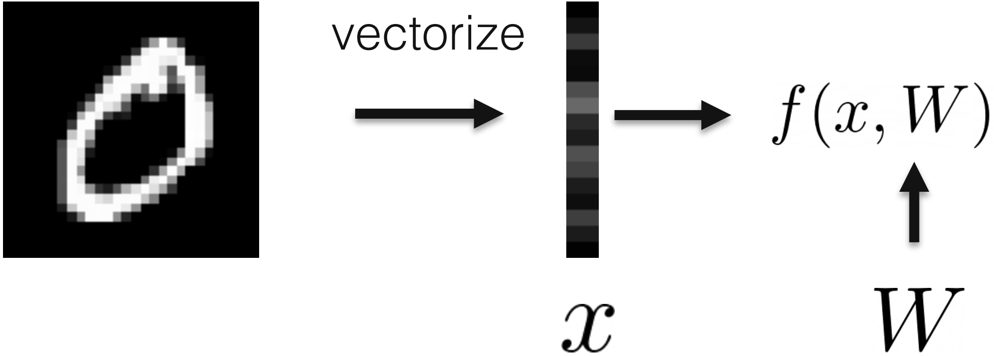
$$f(x, W) = Wx$$



Length of this vector is the “dimensionality” of our problem!

Image Classification

- Linear Model $f(x, W) = Wx$



vectorize

x

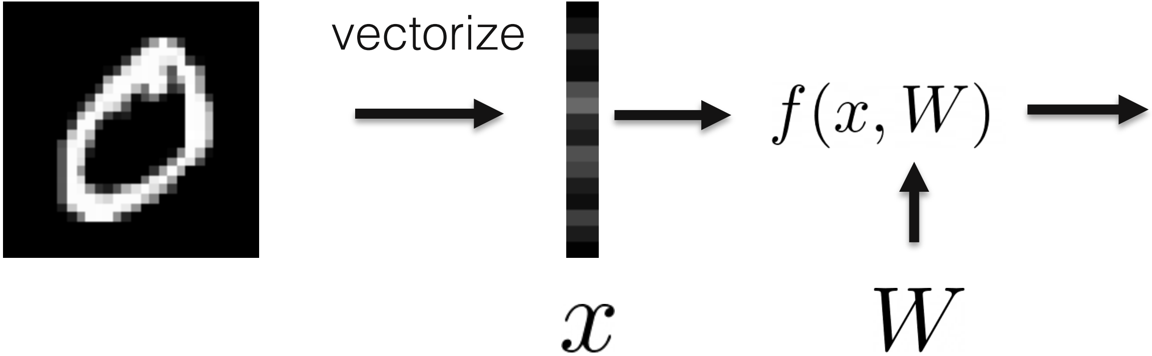
$f(x, W)$

W

$\boxed{\text{In general: } Wx + b}$

Image Classification

- Linear Model $f(x, W) = Wx$



vectorize

x

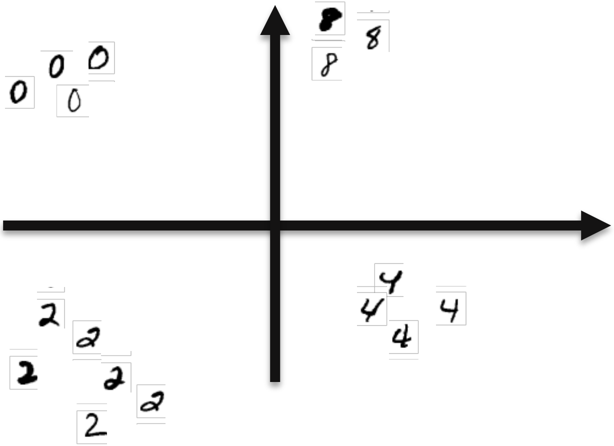
$f(x, W)$

W

10 numbers with class scores

Image Classification

- Linear model: geometric intrepretation

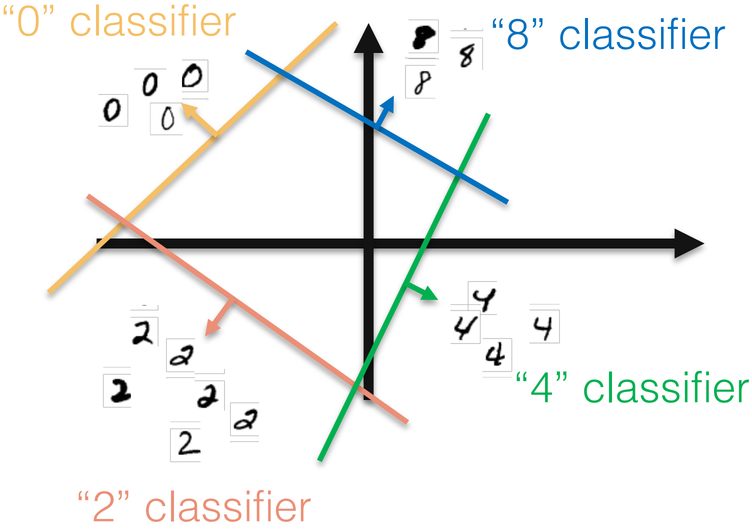


Each image is a point in an N-dimensional space

- N is the number of pixels

Image Classification

- Linear model: geometric interpretation



$$f(x, W) = Wx$$

Computes inner product between rows of W and x!

- Each row of W is a hyperplane
- Sign of inner product tells you which side of the hyperplane
- "separates" the digits

Image Classification

- Linear model (visual interpretation)

Learned filters (rows of W)

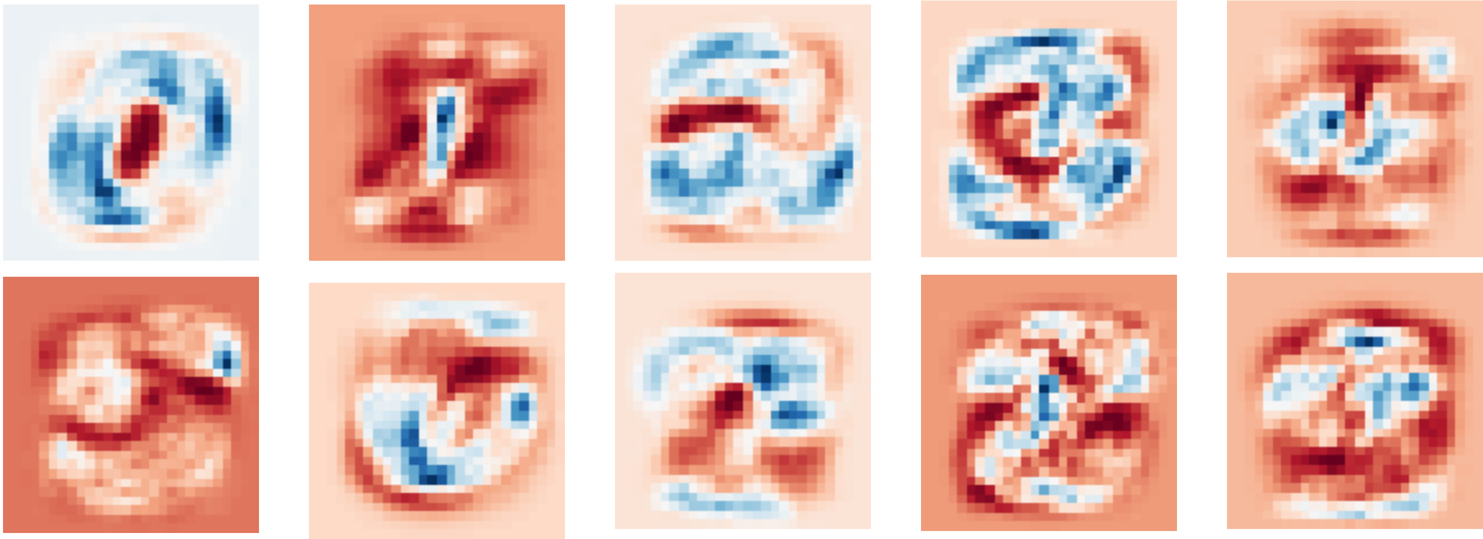
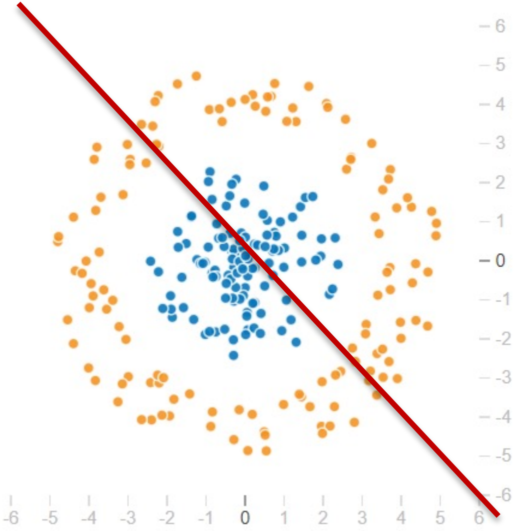


Image Classification

- Limits of linear classifiers

Linear classifiers learn linear decision planes

What if dataset is not linearly separable?



Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$
- 3-layer MLP $f = W_3 \max(0, W_2 \max(0, W_1 x))$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$
- 3-layer MLP $f = W_3 \max(0, W_2 \max(0, W_1 x))$

Non-linearity/activation function between linear layers

Multilayer Perceptrons (MLPs)

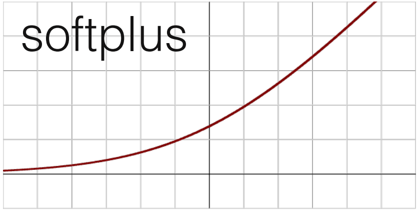
- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$
- 3-layer MLP $f = W_3 \max(0, W_2 \max(0, W_1 x))$

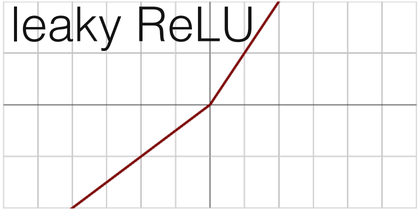
Otherwise we have:

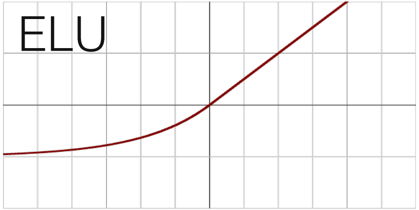
$$f = W_3 W_2 W_1 x$$

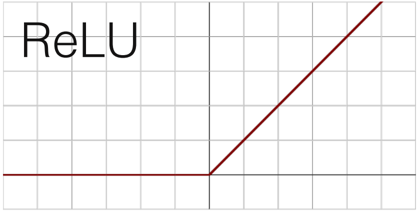
Activation Functions

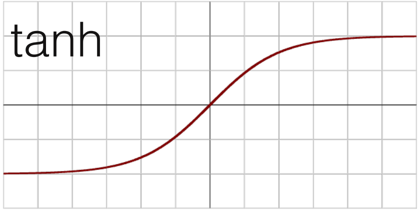
…many to choose from

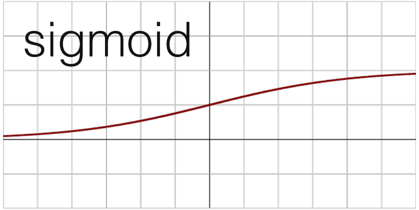










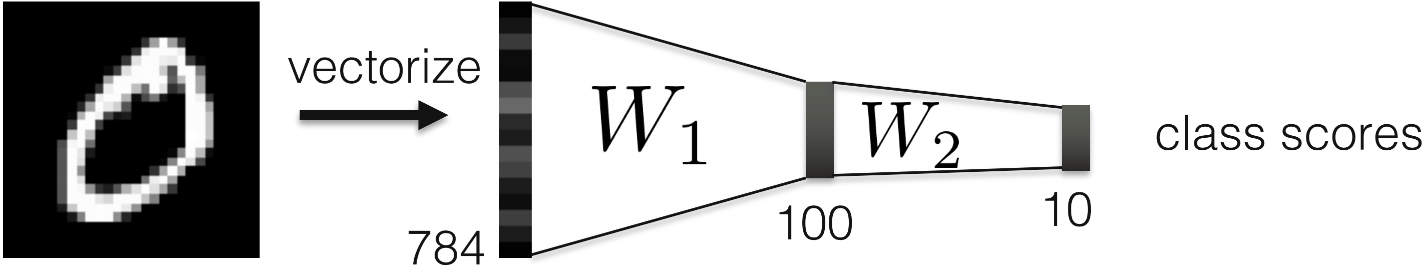


… ReLU is a good general-purpose choice: ReLU(x) = max(0, x)

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

Back to our classification example…

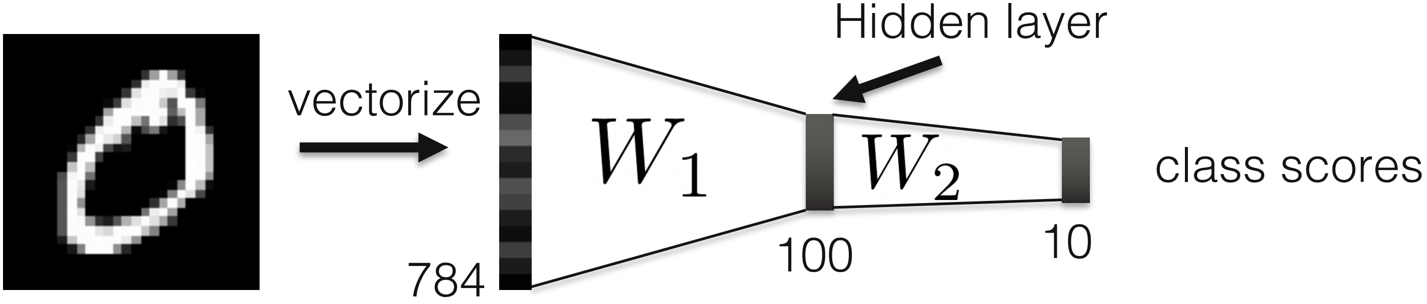


$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

Back to our classification example…

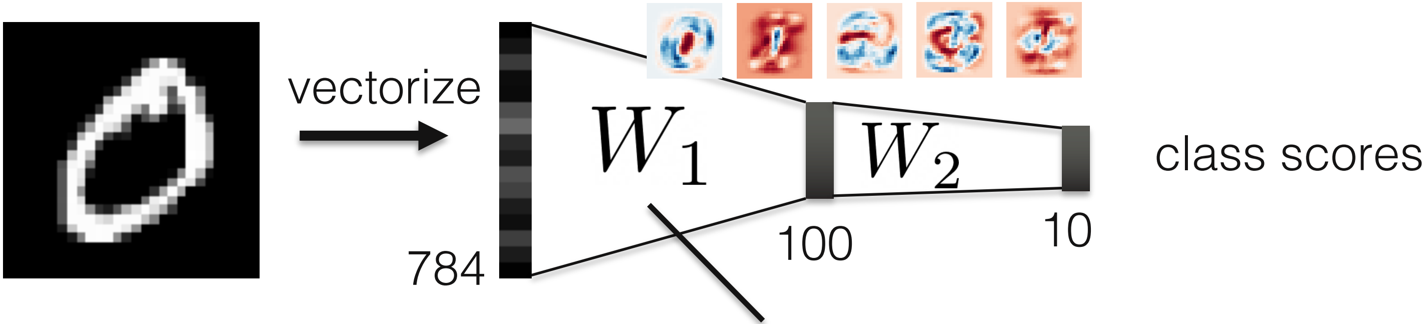


$$x \in \mathbb{R}^D,\, W_1 \in \mathbb{R}^{H \times D},\, W_2 \in \mathbb{R}^{C \times H}$$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

Back to our classification example…



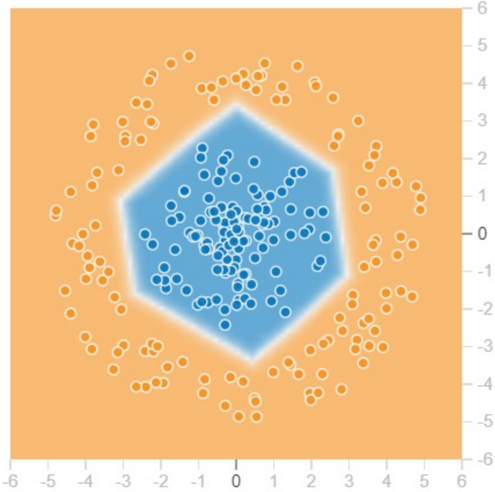
Now we have 100 shape templates, shared between classes

Multilayer Perceptrons (MLPs)

- Overcomes limits of linear classifiers

  - Can learn non-linear decision boundaries

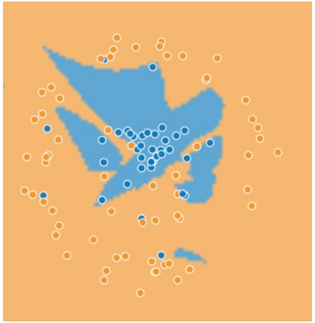
  - Complexity scales with the number of neurons/hidden layers



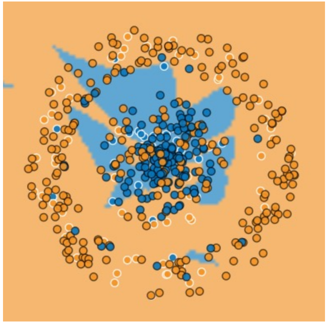
Multilayer Perceptrons (MLPs)

- More parameters is not always better!
  - Can lead to overfitting the training data
  - Performance on test data is worse

train

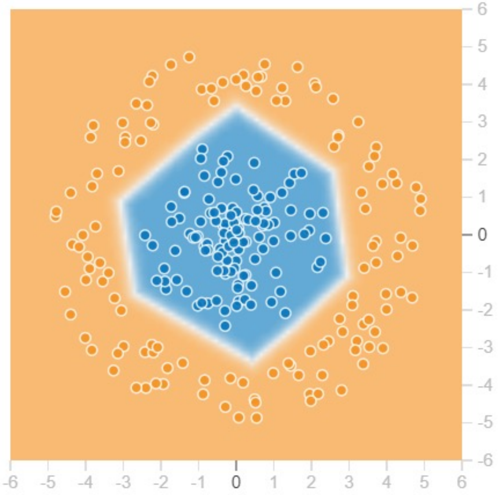


test



Multilayer Perceptrons (MLPs)

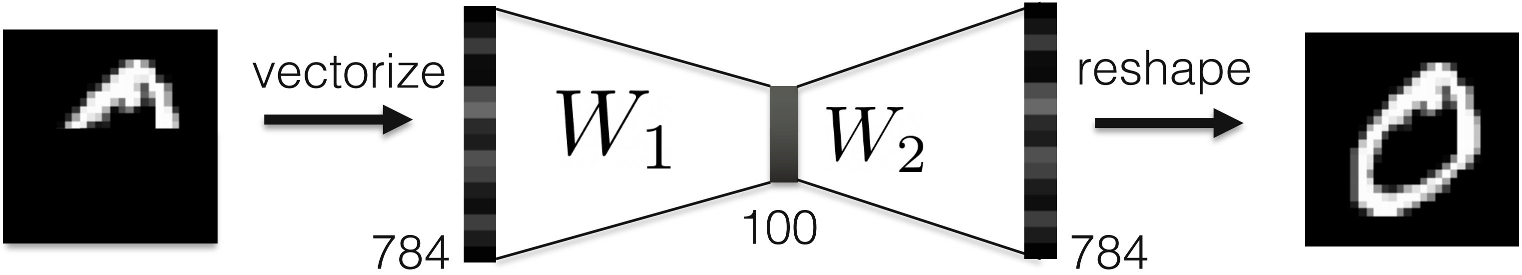
- More on classification…
  - https://cs231n.github.io/linear-classify/
  - https://csc413-uoft.github.io/



Today

- What is a neural network?
- Training/optimizing neural nets
- Why “neural”?
- Convolutional neural networks
- Applications & inverse problems

Image Inpainting



vectorize

W_1

W_2

reshape

784

100

784

masked input

predicted output

Training the MLP

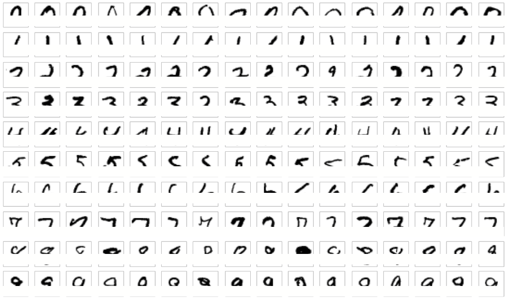
Image inpainting example

Training dataset:

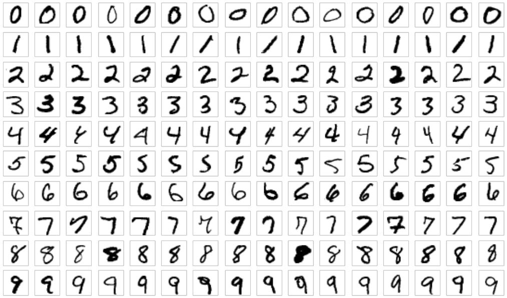
- masked and complete image pairs

- train network to predict the complete image

masked images



ground truth



Training the MLP

Train the network to minimize the loss function

$$\mathcal{L}_\theta = \frac{1}{2}||y - \hat{y}||_2^2$$

network parameters

$\theta = \{W_1, W_2\}$

Training the MLP

Train the network to minimize the loss function

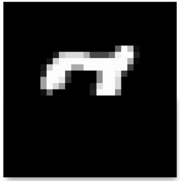
$$\mathcal{L}_\theta = \frac{1}{2}||y - \hat{y}||_2^2$$

network parameters

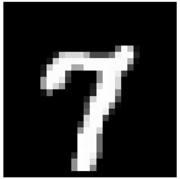
$\theta = \{W_1, W_2\}$

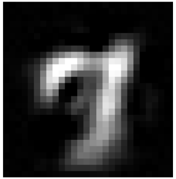
ground truth image

network prediction



input





Training the MLP

How do we figure out θ ?

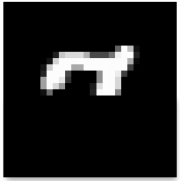
$$\mathcal{L}_\theta = \frac{1}{2}||y - \hat{y}||_2^2$$

network parameters

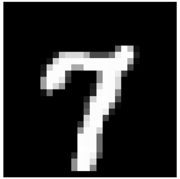
$\theta = \{W_1, W_2\}$

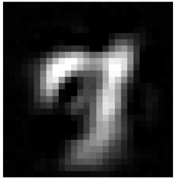
ground truth image

network prediction



input

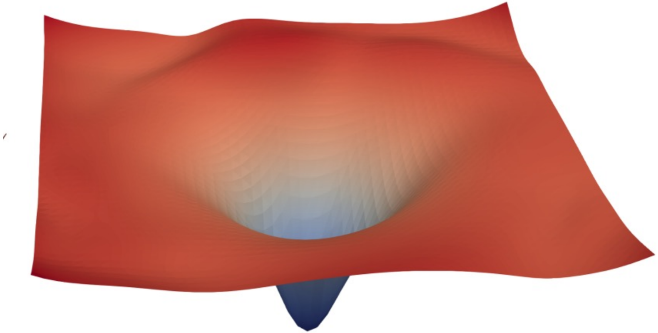




Training the MLP

Gradient-based optimization

$$\nabla_\theta \mathcal{L}_\theta$$



Loss Landscape

[Li et al. ‘18]

Training the MLP

$$\frac{\partial}{\partial W_1}\mathcal{L}_\theta = \frac{\partial}{\partial W_1}\frac{1}{2}\|y - \hat{y}\|_2^2$$

$$\frac{\partial}{\partial W_2}\mathcal{L}_\theta = \frac{\partial}{\partial W_2}\frac{1}{2}\|y - \hat{y}\|_2^2$$

Need to calculate the partial derivative with respect to each parameter

Training the MLP

Generally there are 3 options

1. Numerical differentiation
2. Symbolic differentiation
3. “Automatic” differentiation

Numerical Differentiation

$$\frac{\partial f(x)}{\partial x} \approx \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Not very accurate, computationally expensive

Easy to implement! Can be used to check your analytical answers..

Symbolic Differentiation

$$\begin{aligned}\frac{\partial \mathcal{L}_\theta}{\partial W_1} &= \frac{\partial}{\partial W_1}\frac{1}{2}\|y-\hat{y}\|_2^2 \\ &= \frac{\partial}{\partial W_1}\frac{1}{2}\left(W_2\sigma(W_1x)\right)^T\left(W_2\sigma(W_1x)\right) \\ &= \frac{\partial}{\partial W_1}\frac{1}{2}\sigma(W_1x)^T W_2^T W_2\sigma(W_1x) \\ &= \ldots \quad \text{chain rule, product rule…}\end{aligned}$$

Accurate, but must be manually calculated for each term
Tedious!

Automatic Differentiation

Think about the problem as a “computational graph”

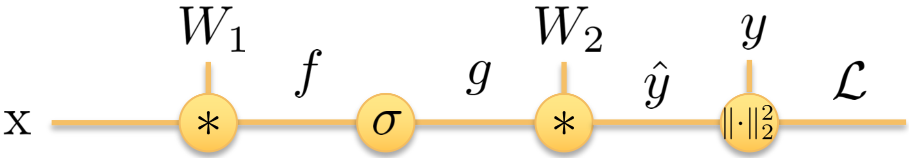
Divide and conquer using the chain rule

Enables “backpropagation” – an efficient way to take derivatives of all parameters in a computational graph

Automatic Differentiation

Think about the problem as a “computational graph”

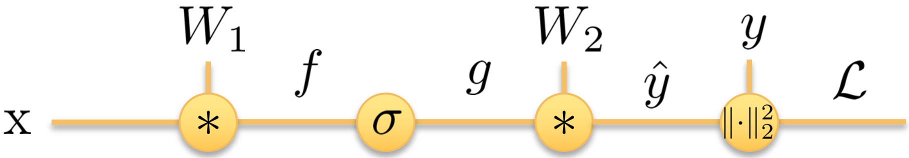
Divide and conquer using the chain rule



Automatic Differentiation

Think about the problem as a "computational graph"

Divide and conquer using the chain rule

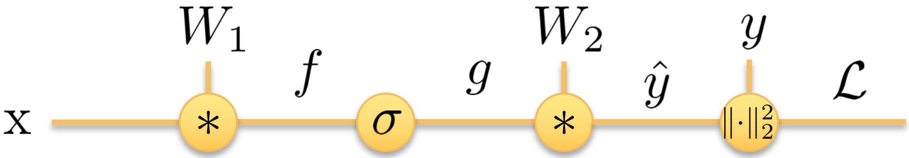


$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial \hat{y}}{\partial W_2} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Automatic Differentiation

Think about the problem as a "computational graph"

Divide and conquer using the chain rule

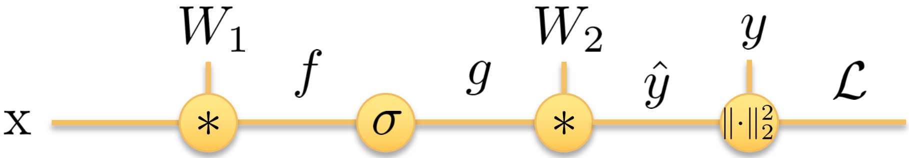


$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial f}{\partial W_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Automatic Differentiation

Think about the problem as a "computational graph"

Divide and conquer using the chain rule

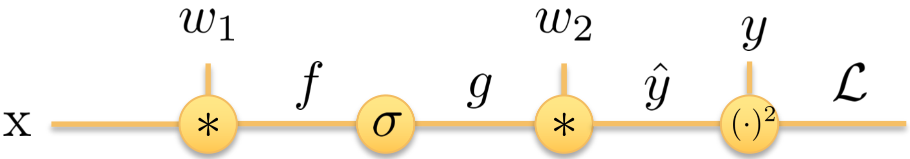


$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial f}{\partial W_1}\frac{\partial g}{\partial f}\frac{\partial \hat{y}}{\partial g}\frac{\partial \mathcal{L}}{\partial \hat{y}}$$

We can calculate analytical expressions for each of these terms and then plug in our values

Autodiff Example

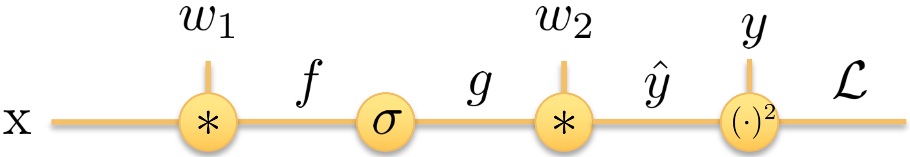
(assume scalar values for now)



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1}\frac{\partial g}{\partial f}\frac{\partial \hat{y}}{\partial g}\boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

Autodiff Example

(assume scalar values for now)

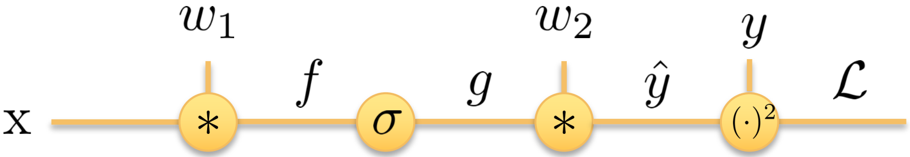


$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1}\frac{\partial g}{\partial f}\frac{\partial \hat{y}}{\partial g}\boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\frac{1}{2}(\hat{y} - y)^2 = \hat{y} - y$$

Autodiff Example

(assume scalar values for now)

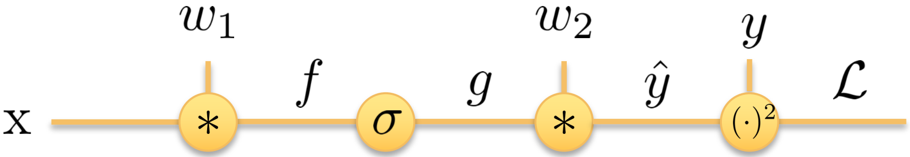


$$\frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1}\frac{\partial g}{\partial f}\boxed{\frac{\partial \hat{y}}{\partial g}}\frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

(assume scalar values for now)



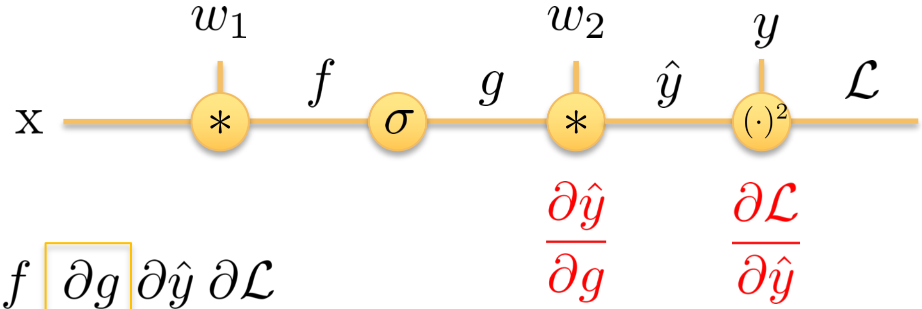
$$\frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \frac{\partial g}{\partial f} \boxed{\frac{\partial \hat{y}}{\partial g}} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} w_2 \cdot g = w_2$$

Autodiff Example

(assume scalar values for now)

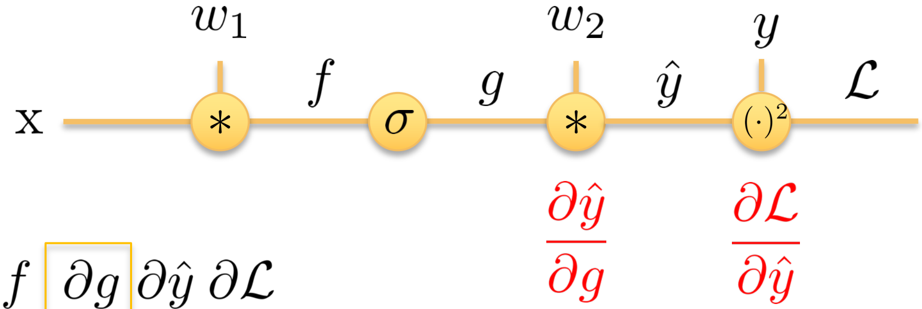


w_1
w_2
y
x
$*$
f
σ
g
$*$
$\hat{y}$
$(\cdot)^2$
$\mathcal{L}$
$\frac{\partial \hat{y}}{\partial g}$
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \boxed{\frac{\partial g}{\partial f}} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

(assume scalar values for now)



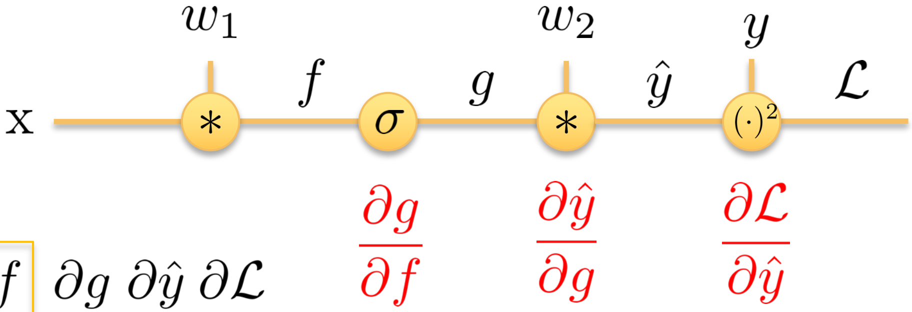
w_1
w_2
y
x
$*$
f
σ
g
$*$
$\hat{y}$
$(\cdot)^2$
$\mathcal{L}$
$\frac{\partial \hat{y}}{\partial g}$
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \boxed{\frac{\partial g}{\partial f}} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial g}{\partial f} = \frac{\partial}{\partial f}\sigma(f) = \frac{\partial}{\partial f}\max(0, f) = \begin{cases} 0, & f < 0 \\ 1 & \text{else} \end{cases}$$

Autodiff Example

(assume scalar values for now)

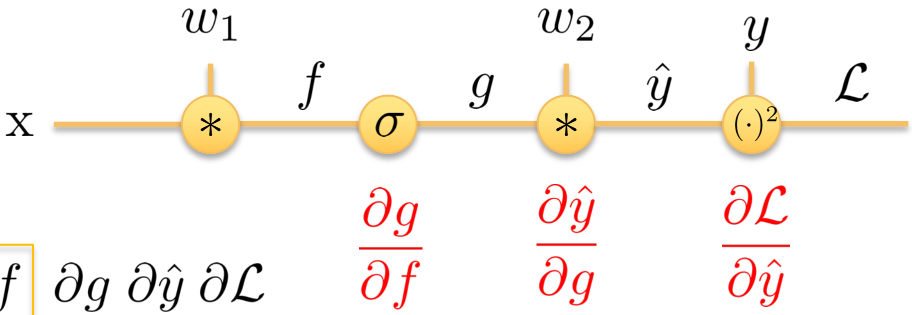


w_1
w_2
y
f
g
$\hat{y}$
$\mathcal{L}$
x
$*$
σ
$*$
$(\cdot)^2$
$\frac{\partial g}{\partial f}$
$\frac{\partial \hat{y}}{\partial g}$
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial f}{\partial w_1}} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

(assume scalar values for now)

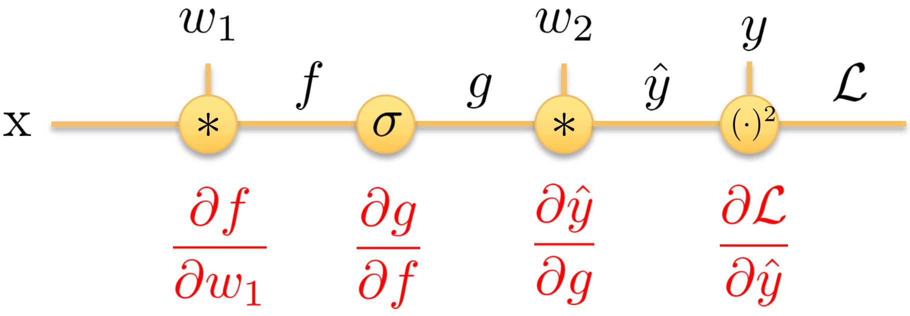

x
w_1
$*$
f
σ
g
w_2
$*$
$\hat{y}$
y
$(\cdot)^2$
$\mathcal{L}$
$\frac{\partial g}{\partial f}$
$\frac{\partial \hat{y}}{\partial g}$
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial f}{\partial w_1}} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial f}{\partial w_1} = \frac{\partial}{\partial w_1} w_1 \cdot x = x$$

Autodiff Example

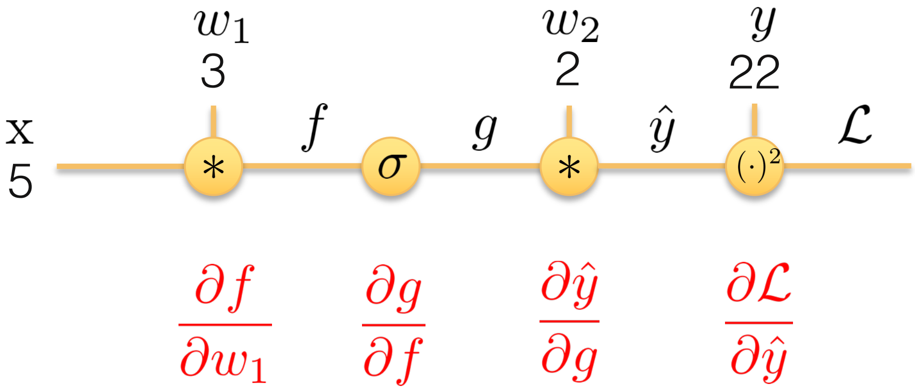
(assume scalar values for now)



w_1
w_2
y
x
$*$
f
σ
g
$*$
$\hat{y}$
$(\cdot)^2$
$\mathcal{L}$
$\frac{\partial f}{\partial w_1}$
$\frac{\partial g}{\partial f}$
$\frac{\partial \hat{y}}{\partial g}$
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$

Autodiff Example

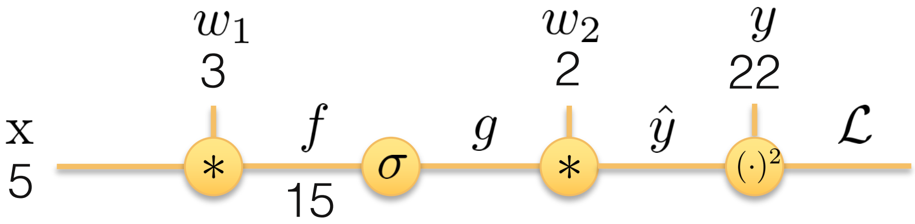
Let's plug in the values now…



w_1
3
w_2
2
y
22
x
5
f
g
$\hat{y}$
$\mathcal{L}$
$*$
σ
$*$
$(\cdot)^2$
$\frac{\partial f}{\partial w_1}$
$\frac{\partial g}{\partial f}$
$\frac{\partial \hat{y}}{\partial g}$
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$

Autodiff Example

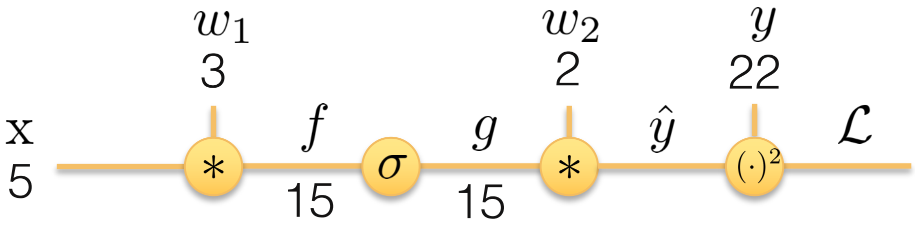
Let’s plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

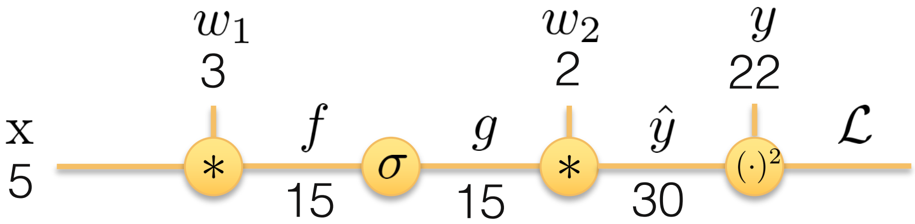
Let’s plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

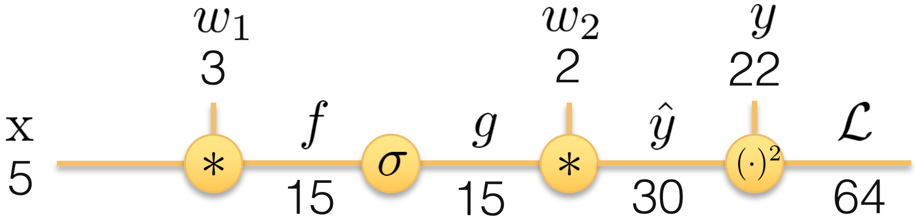
Let’s plug in the values now…



$$\frac{\partial f}{\partial w_1} \quad \frac{\partial g}{\partial f} \quad \frac{\partial \hat{y}}{\partial g} \quad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

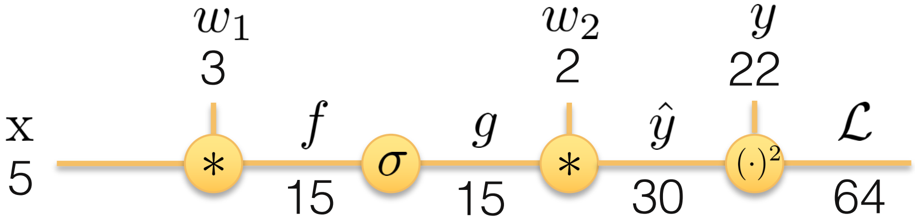
Let’s plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

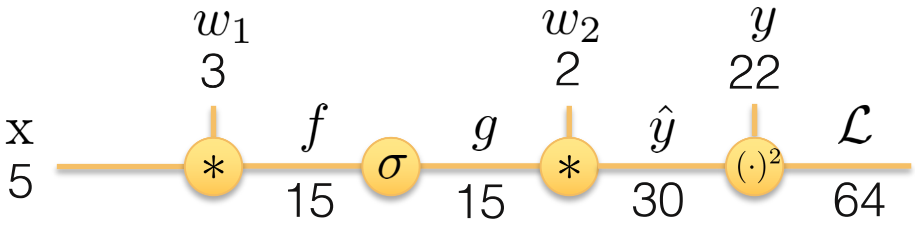
Let’s plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

Autodiff Example

Let’s plug in the values now…

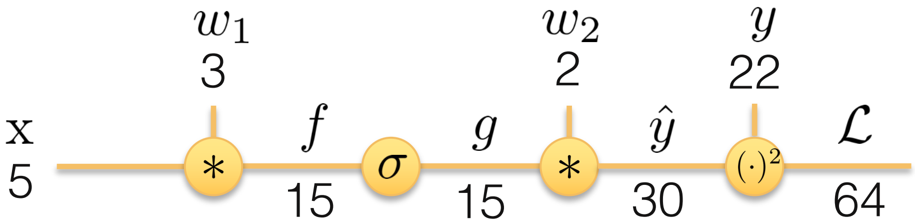


$$\frac{\partial f}{\partial w_1} \quad \frac{\partial g}{\partial f} \quad \frac{\partial \hat{y}}{\partial g} \quad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

8

Autodiff Example

Let’s plug in the values now…



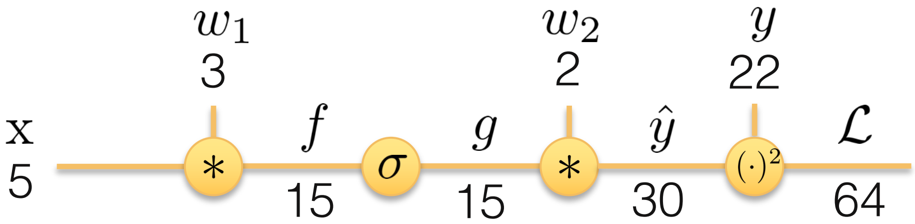
$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

2

8

Autodiff Example

Let’s plug in the values now…



$$\frac{\partial f}{\partial w_1}$$

$$\frac{\partial g}{\partial f}$$

1

$$\frac{\partial \hat{y}}{\partial g}$$

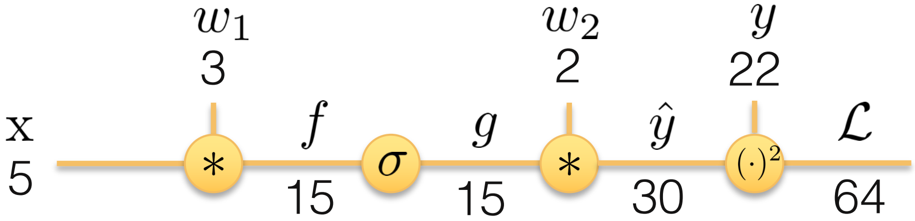
2

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

8

Autodiff Example

Let’s plug in the values now…

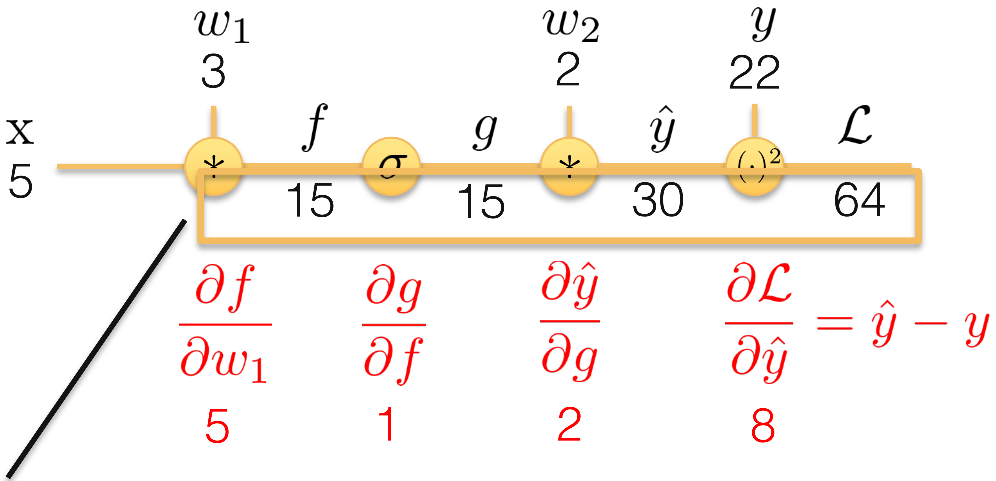


$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

5 1 2 8

Autodiff Example

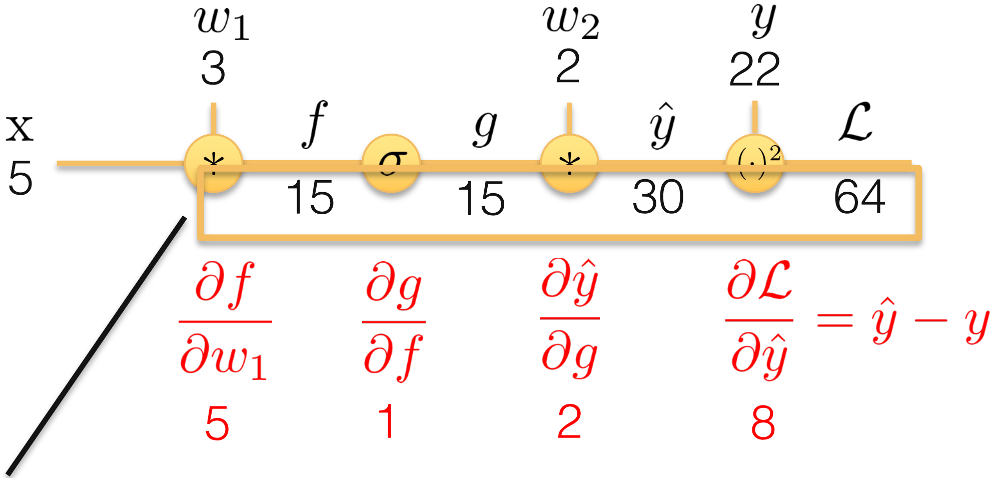
What is backpropagation?


w_1
3
w_2
2
y
22
x
5
f
g
$\hat{y}$
$\mathcal{L}$
$*$
σ
$*$
$(\cdot)^2$
15
15
30
64
$\frac{\partial f}{\partial w_1}$
5
$\frac{\partial g}{\partial f}$
1
$\frac{\partial \hat{y}}{\partial g}$
2
$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$
8

Save these intermediate values during forward computation

Autodiff Example

What is backpropagation?

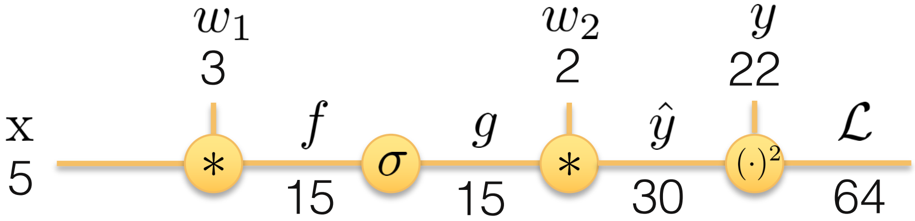


w_1
3
w_2
2
y
22
x
5
f
g
$\hat{y}$
$\mathcal{L}$
*
σ
*
$(\cdot)^2$
15
15
30
64
$\frac{\partial f}{\partial w_1}$
$\frac{\partial g}{\partial f}$
$\frac{\partial \hat{y}}{\partial g}$
$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$
5
1
2
8

Then we perform a “backward pass”

Autodiff Example

What is backpropagation?



$\frac{\partial f}{\partial w_1}$ 5

$\frac{\partial g}{\partial f}$ 1

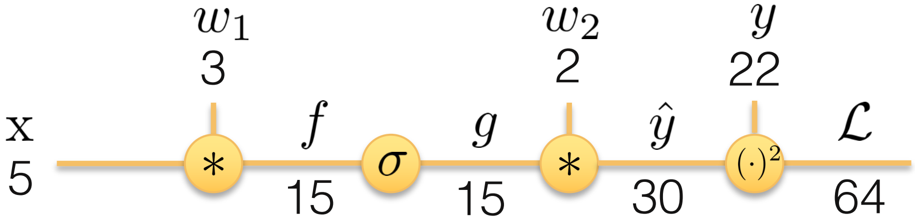
$\frac{\partial \hat{y}}{\partial g}$ 2

$\frac{\partial \mathcal{L}}{\partial g}$

$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$ 8

Autodiff Example

What is backpropagation?



$$\frac{\partial f}{\partial w_1}$$

5

$$\frac{\partial g}{\partial f}$$

1

$$\frac{\partial \hat{y}}{\partial g}$$

2

$$\frac{\partial \mathcal{L}}{\partial g}$$

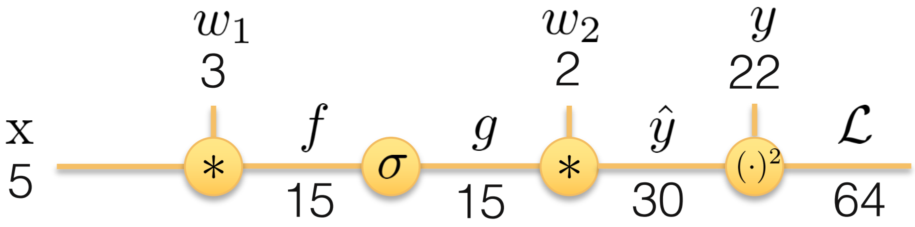
16

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

8

Autodiff Example

What is backpropagation?



$\frac{\partial f}{\partial w_1}$ $\frac{\partial g}{\partial f}$ $\frac{\partial \hat{y}}{\partial g}$ $\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$

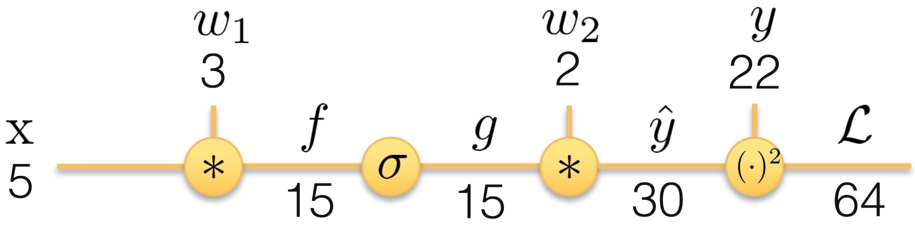
5 1 2 8

$\frac{\partial \mathcal{L}}{\partial f}$ $\frac{\partial \mathcal{L}}{\partial g}$

16

Autodiff Example

What is backpropagation?



$$\frac{\partial f}{\partial w_1}$$

5

$$\frac{\partial g}{\partial f}$$

1

$$\frac{\partial \mathcal{L}}{\partial f}$$

16

$$\frac{\partial \hat{y}}{\partial g}$$

2

$$\frac{\partial \mathcal{L}}{\partial g}$$

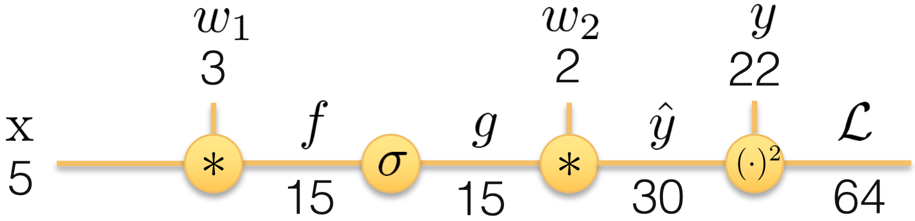
16

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

8

Autodiff Example

What is backpropagation?



$$\frac{\partial f}{\partial w_1}$$
5
$$\frac{\partial \mathcal{L}}{\partial w_1}$$

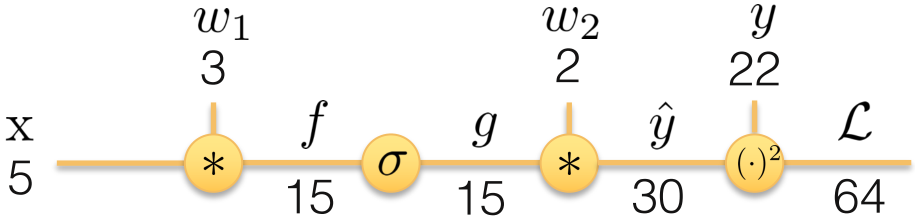
$$\frac{\partial g}{\partial f}$$
1
$$\frac{\partial \mathcal{L}}{\partial f}$$
16

$$\frac{\partial \hat{y}}{\partial g}$$
2
$$\frac{\partial \mathcal{L}}{\partial g}$$
16

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$
8

Autodiff Example

What is backpropagation?



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

5 1 2 8

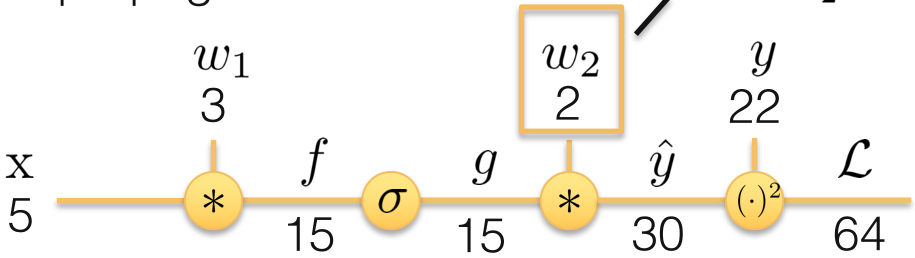
$$\frac{\partial \mathcal{L}}{\partial w_1} \qquad \frac{\partial \mathcal{L}}{\partial f} \qquad \frac{\partial \mathcal{L}}{\partial g}$$

90 16 16

Autodiff Example

What is backpropagation?

What about $\frac{\partial \mathcal{L}}{\partial w_2}$?



$$\frac{\partial f}{\partial w_1}$$

5

$$\frac{\partial \mathcal{L}}{\partial w_1}$$

90

$$\frac{\partial g}{\partial f}$$

1

$$\frac{\partial \mathcal{L}}{\partial f}$$

16

$$\frac{\partial \hat{y}}{\partial g}$$

2

$$\frac{\partial \mathcal{L}}{\partial g}$$

16

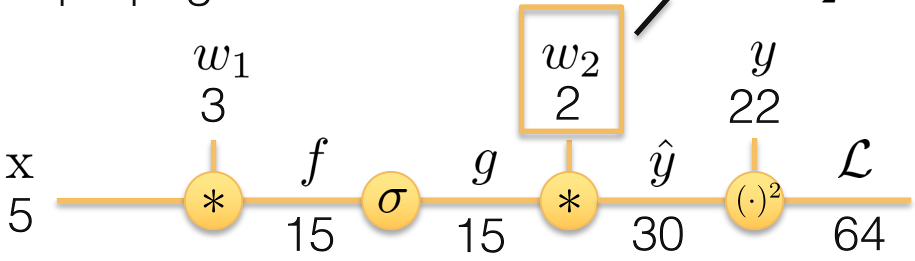
$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

8

Autodiff Example

What is backpropagation?

What about $\frac{\partial \mathcal{L}}{\partial w_2}$?



$$\frac{\partial f}{\partial w_1}$$
5
$$\frac{\partial \mathcal{L}}{\partial w_1}$$
90

$$\frac{\partial g}{\partial f}$$
1
$$\frac{\partial \mathcal{L}}{\partial f}$$
16

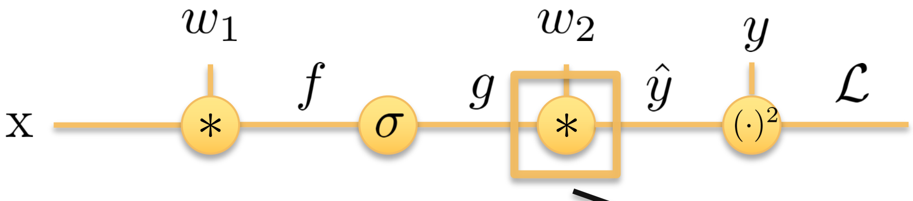
$$\frac{\partial \hat{y}}{\partial g}$$
2
$$\frac{\partial \mathcal{L}}{\partial g}$$
16

$$\boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}} = \hat{y} - y$$

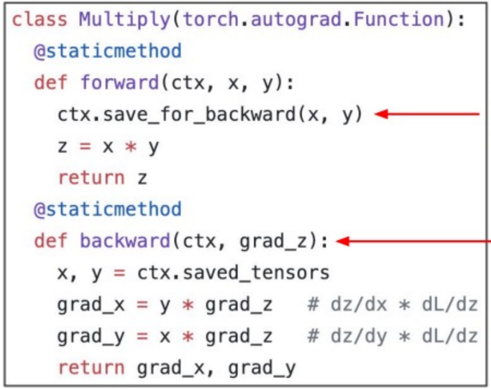
We can re-use computation!

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \hat{y}}{\partial w_2} \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

Autodiff Example



PyTorch Code:



```
class Multiply(torch.autograd.Function):
  @staticmethod
  def forward(ctx, x, y):
    ctx.save_for_backward(x, y)
    z = x * y
    return z
  @staticmethod
  def backward(ctx, grad_z):
    x, y = ctx.saved_tensors
    grad_x = y * grad_z    # dz/dx * dL/dz
    grad_y = x * grad_z    # dz/dy * dL/dz
    return grad_x, grad_y
```

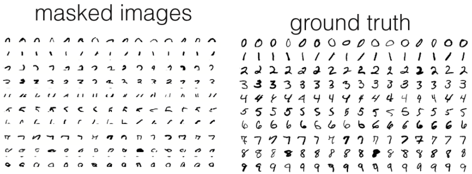
Need to stash some values for use in backward

Upstream gradient

Multiply upstream and local gradients

Image Inpainting Training Loop

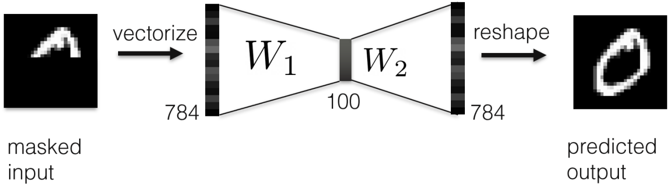
1. Sample batch of images from dataset



masked images

ground truth

2. Run forward pass to calculate network output for each image



3. Run backward pass to calculate gradients with backpropagation

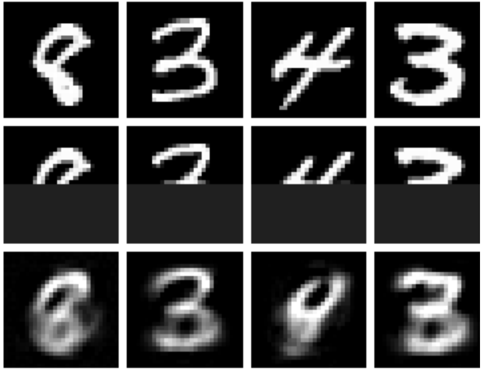
4. Update parameters with stochastic gradient descent

4. Update parameters with stochastic gradient descent

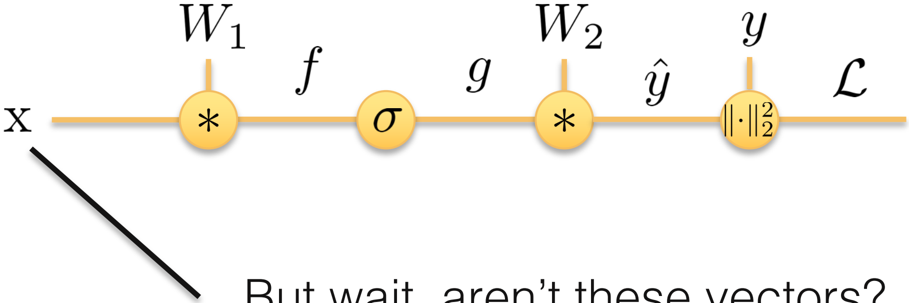
$$\mathcal{L}_\theta = \|y - \hat{y}\|_2^2$$

$$W_2^{(k+1)} = W_2^{(k)} - \alpha \frac{\partial \mathcal{L}}{\partial W_2}$$

$$W_1^{(k+1)} = W_1^{(k)} - \alpha \frac{\partial \mathcal{L}}{\partial W_1}$$

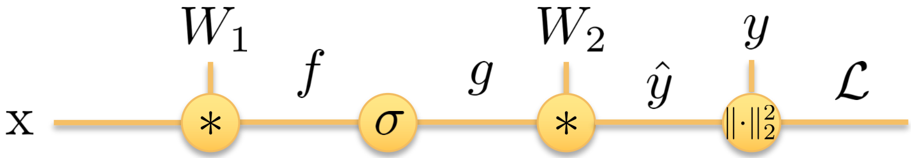


Vector Differentiation



But wait, aren't these vectors?

Vector Differentiation



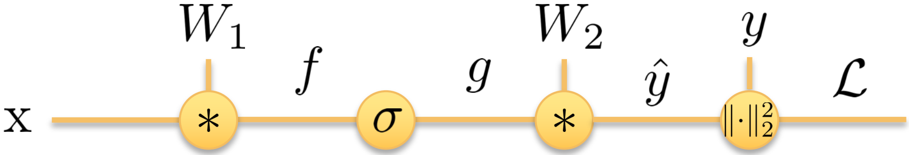
Recap: vector differentiation

Scalar by Scalar

$$x, y \in \mathbb{R}$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

Vector Differentiation



Recap: vector differentiation

Scalar by Scalar

$$x, y \in \mathbb{R}$$

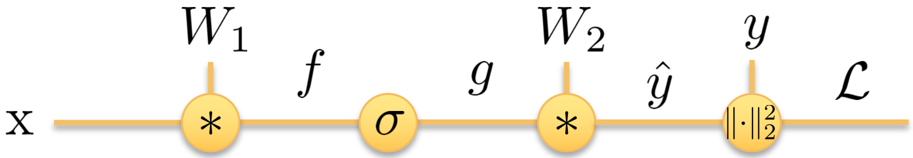
$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

Scalar by Vector

$$x \in \mathbb{R}^N, y \in \mathbb{R}$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N$$

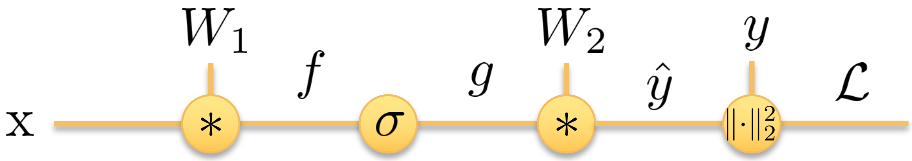
Vector Differentiation



Recap: vector differentiation

Scalar by Scalar	Scalar by Vector	Vector by Vector
$x, y \in \mathbb{R}$	$x \in \mathbb{R}^N, y \in \mathbb{R}$	$x \in \mathbb{R}^N, y \in \mathbb{R}^M$
$\frac{\partial y}{\partial x} \in \mathbb{R}$	$\frac{\partial y}{\partial x} \in \mathbb{R}^N$	$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M}$

Recap: vector differentiation



Example 1: matrix multiply

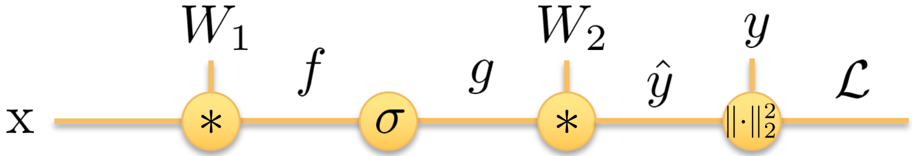
$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$W_2 \in \mathbb{R}^{M \times N}$$

$$g \in \mathbb{R}^{N}$$

$$\frac{\partial \hat{y}}{\partial g} \in \mathbb{R}^{N \times M}$$

Recap: vector differentiation



Example 1: matrix multiply

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$W_2 \in \mathbb{R}^{M \times N}$$

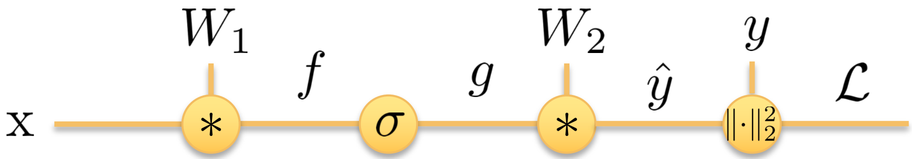
$$g \in \mathbb{R}^N$$

$$\frac{\partial \hat{y}}{\partial g} \in \mathbb{R}^{N \times M}$$

$$\frac{\partial}{\partial g} W_2 g$$

$$= \frac{\partial}{\partial g} \begin{bmatrix} w_{11}g_1 + \cdots + w_{1n}g_n \\ \vdots \quad \ddots \quad \vdots \\ w_{m1}g_1 + \cdots + w_{mn}g_n \end{bmatrix}$$

Recap: vector differentiation



$$\frac{\partial \hat{y}}{\partial g} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial g_1} & \cdots & \frac{\partial \hat{y}_m}{\partial g_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_1}{\partial g_n} & \cdots & \frac{\partial \hat{y}_m}{\partial g_n} \end{bmatrix}$$

Example 1: matrix multiply

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$W_2 \in \mathbb{R}^{M \times N}$$

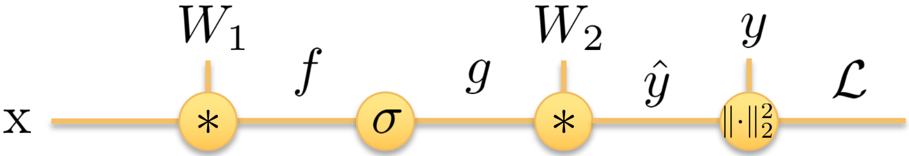
$$g \in \mathbb{R}^{N}$$

$$\frac{\partial \hat{y}}{\partial g} \in \mathbb{R}^{N \times M}$$

$$\frac{\partial}{\partial g} W_2 g$$

$$= \frac{\partial}{\partial g} \begin{bmatrix} w_{11}g_1 + \cdots + w_{1n}g_n \\ \vdots \quad \ddots \quad \vdots \\ w_{m1}g_1 + \cdots + w_{mn}g_n \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{m1} \\ \vdots & \ddots & \vdots \\ w_{1n} & \cdots & w_{mn} \end{bmatrix}$$

Recap: vector differentiation



$$\boxed{\frac{\partial \hat{y}}{\partial g} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial g_1} & \cdots & \frac{\partial \hat{y}_n}{\partial g_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_1}{\partial g_n} & \cdots & \frac{\partial \hat{y}_n}{\partial g_n} \end{bmatrix}}$$

Example 1: matrix multiply

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$W_2 \in \mathbb{R}^{M \times N}$$

$$g \in \mathbb{R}^{N}$$

$$\frac{\partial \hat{y}}{\partial g} \in \mathbb{R}^{N \times M}$$

$$\frac{\partial}{\partial g} W_2 g$$

$$= \frac{\partial}{\partial g} \begin{bmatrix} w_{11} g_1 + \cdots + w_{1n} g_n \\ \vdots \quad \ddots \quad \vdots \\ w_{n1} g_1 + \cdots + w_{nn} g_n \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{n1} \\ \vdots & \ddots & \vdots \\ w_{1n} & \cdots & w_{nn} \end{bmatrix}$$

$$= \boxed{W_2^T}$$

Recap: vector differentiation

Example 2: elementwise functions

$$h = f \odot g$$

$$f \in \mathbb{R}^N$$

$$g \in \mathbb{R}^N$$

$$\frac{\partial h}{\partial f} \in \mathbb{R}^{N \times N}$$

Recap: vector differentiation

Example 2: elementwise functions

$$\frac{\partial h}{\partial f} = \begin{bmatrix} \frac{\partial h_1}{\partial f_1} & \cdots & \frac{\partial h_n}{\partial f_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_1}{\partial f_n} & \cdots & \frac{\partial h_n}{\partial f_n} \end{bmatrix}$$

$$h = f \odot g$$

$$f \in \mathbb{R}^N$$

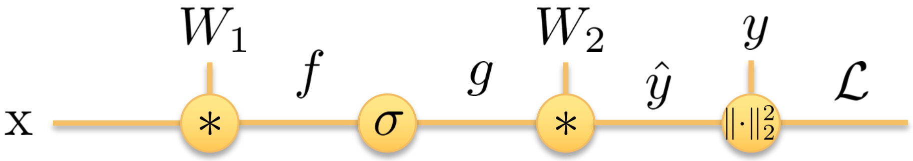
$$g \in \mathbb{R}^N$$

$$\frac{\partial h}{\partial f} \in \mathbb{R}^{N \times N}$$

$$\frac{\partial h}{\partial f} = \begin{bmatrix} g_1 & & 0 \\ & \ddots & \\ 0 & & g_n \end{bmatrix} = \mathrm{diag}(g)$$

Recap: vector differentiation

Final hint: dimensions should always match up!



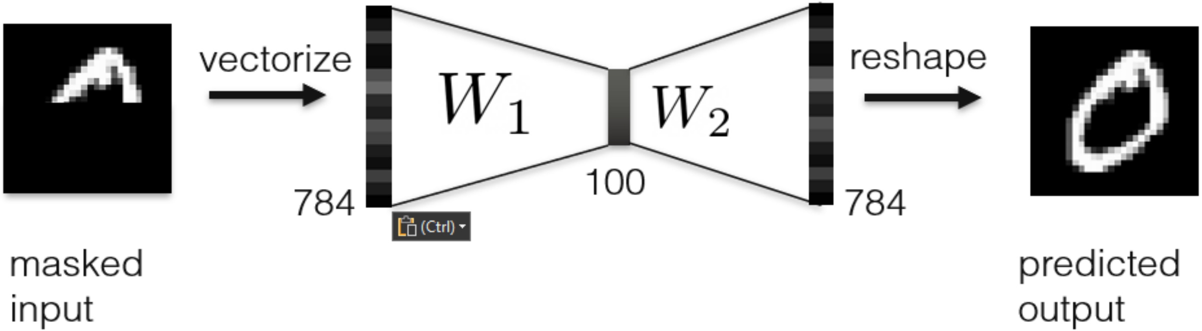
$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial f}{\partial W_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

You should be able to calculate derivatives of each of these terms and then perform matrix multiplications without issues

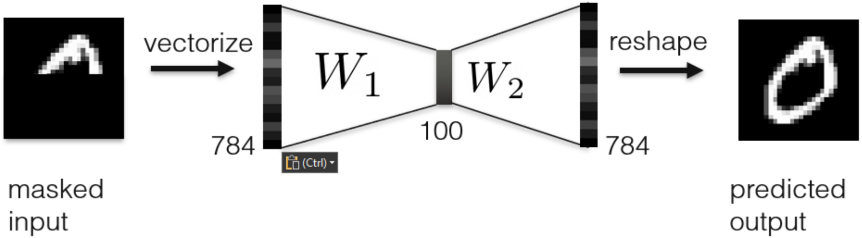
Today

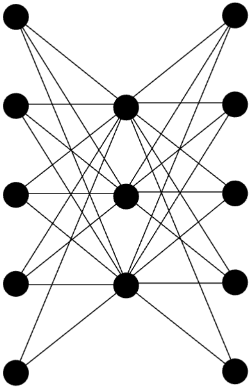
- What is a neural network?
- Training/optimizing neural nets
- Why “neural”?
- Convolutional neural networks
- Applications & inverse problems

Why "neural" network?

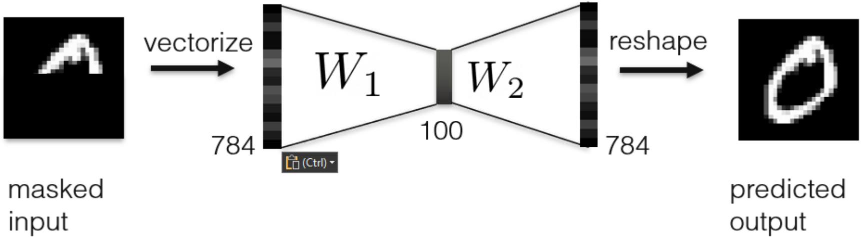


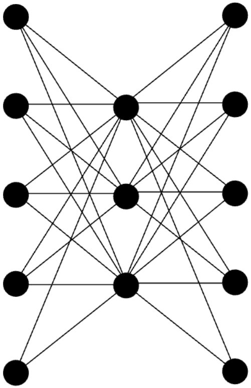
Why “neural” network?





Why “neural” network?



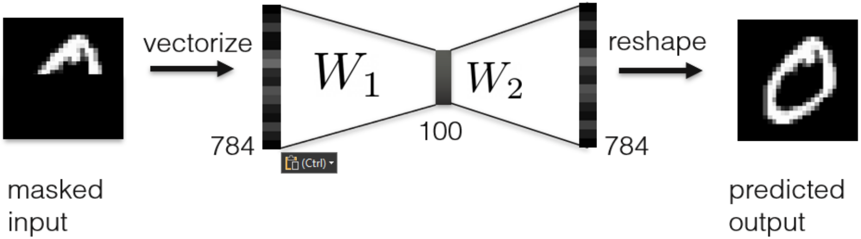


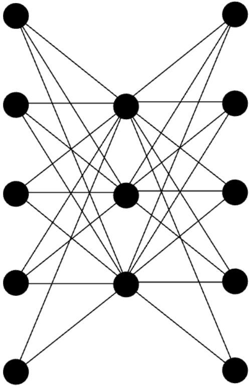
Input Layer

Output Layer

Hidden Layer

Why “neural” network?





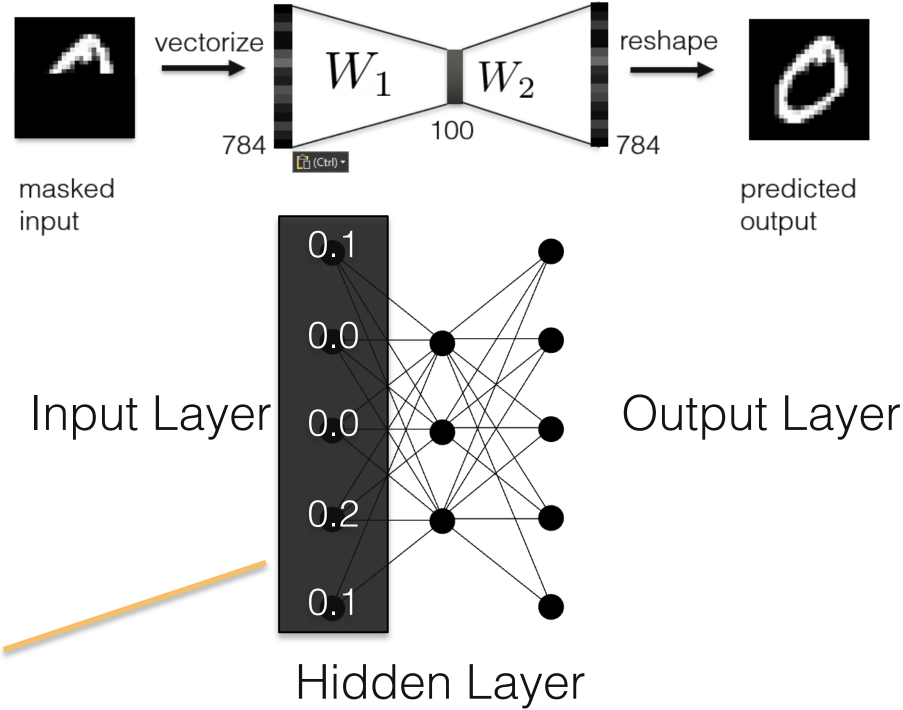
Input Layer

Output Layer

“Neuron”

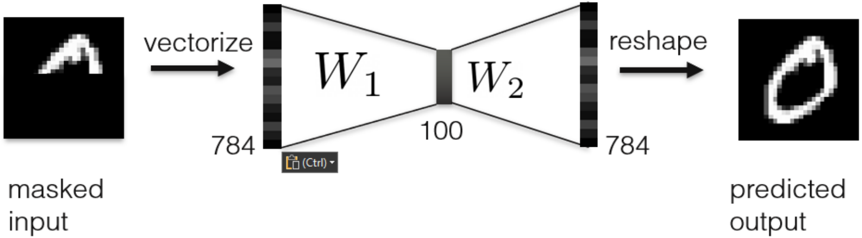
Hidden Layer

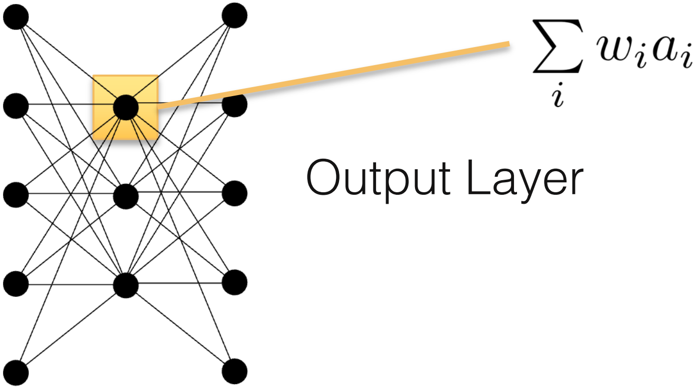
Why “neural” network?



“Activations”

Why “neural” network?





$$\sum_i w_i a_i$$

Input Layer

Output Layer

Hidden Layer

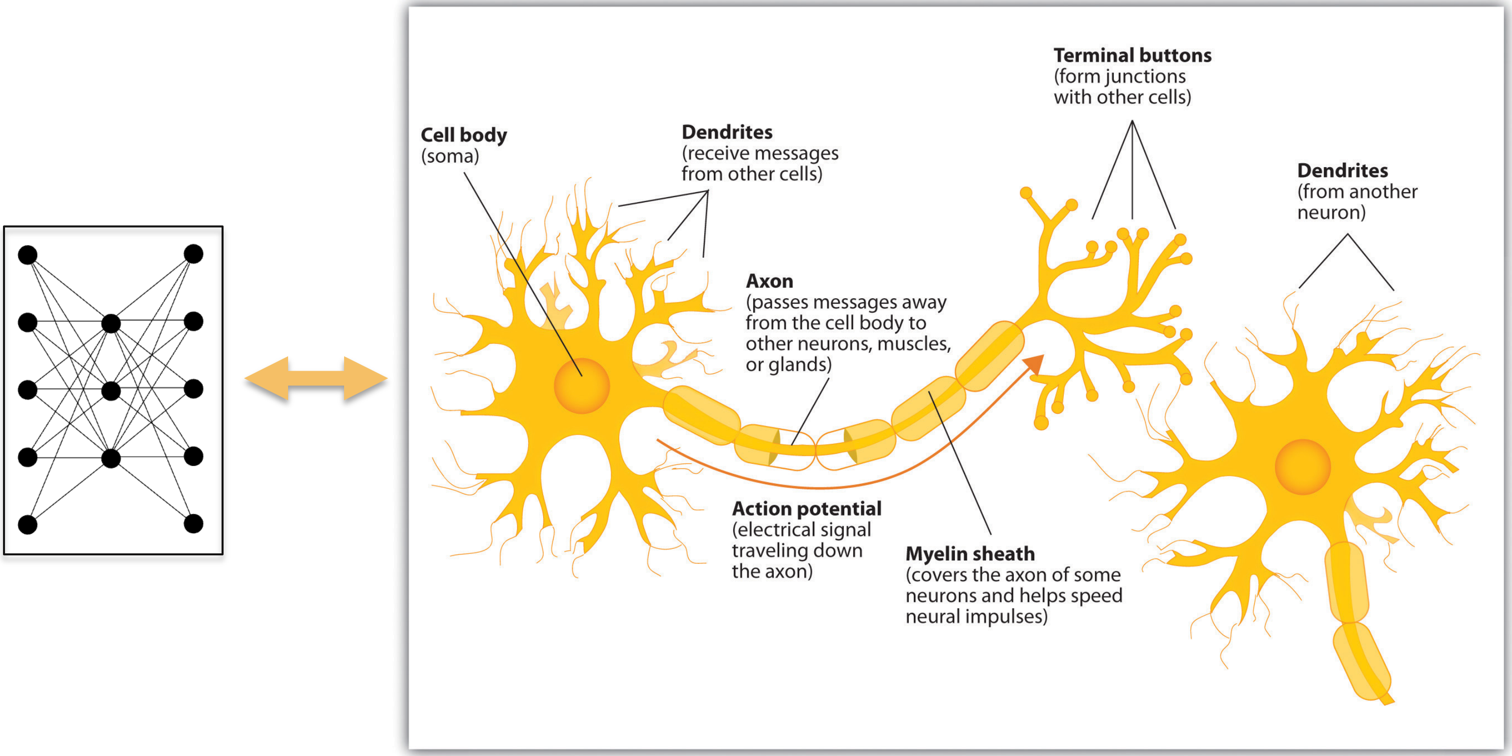
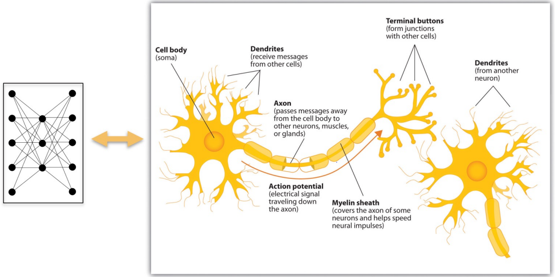

Cell body
(soma)
Dendrites
(receive messages from other cells)
Axon
(passes messages away from the cell body to other neurons, muscles, or glands)
Terminal buttons
(form junctions with other cells)
Dendrites
(from another neuron)
Action potential
(electrical signal traveling down the axon)
Myelin sheath
(covers the axon of some neurons and helps speed neural impulses)

Image: CC BY-SA Jennifer Walinga

Loose analogy!

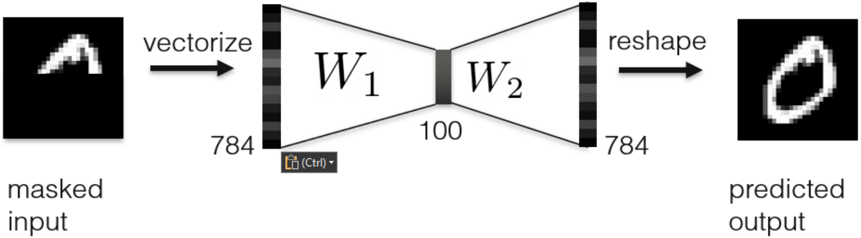
- Neurons have activation potentials, all-or-none firing behavior
- Interconnectivity between actual neurons is dense and complicated
- Connection between neurons is complex non-linear dynamical system



Terminal buttons (form junctions with other cells)
Cell body (soma)
Dendrites (receive messages from other cells)
Dendrites (from another neuron)
Axon (passes messages away from the cell body to other neurons, muscles, or glands)
Action potential (electrical signal traveling down the axon)
Myelin sheath (covers the axon of some neurons and helps speed neural impulses)

Image: CC BY-SA Jennifer Walinga

Drawbacks of fully-connected networks



- spatial structure is destroyed
- fully-connected weights do not scale

Today

- What is a neural network?
- Training/optimizing neural nets
- Why “neural”?
- Convolutional neural networks
- Applications & inverse problems

Convolutional Neural Networks

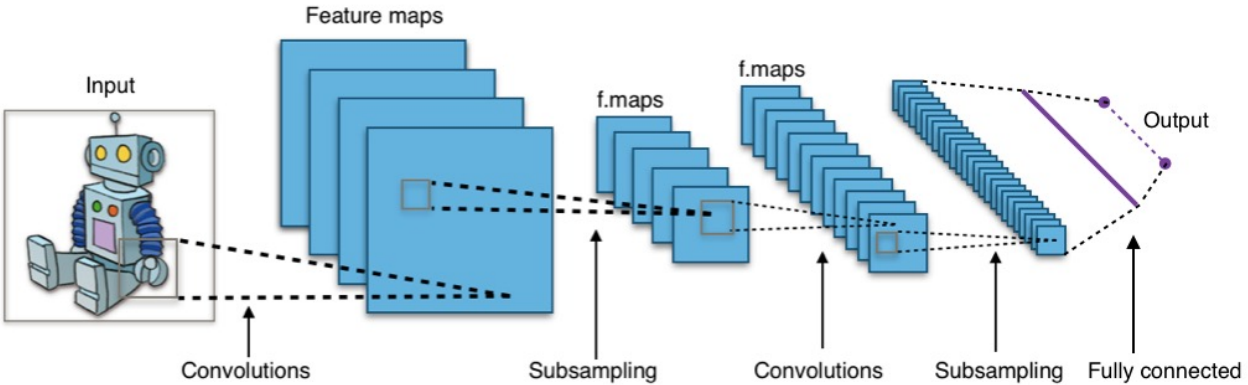


Image: CC Aphex34

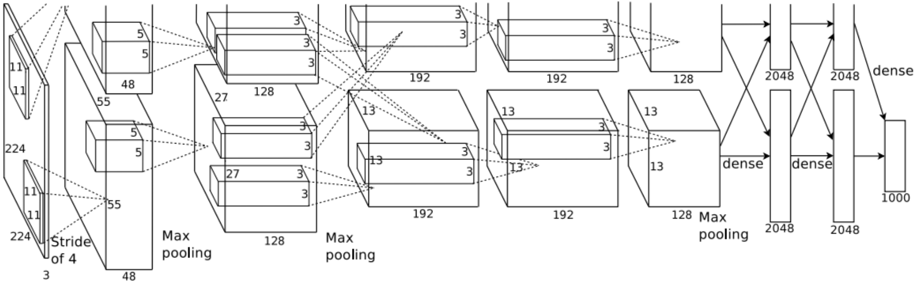
- Exploit spatial structure
- Scale to large inputs with fewer parameters
- Remarkable performance for processing visual data

AlexNet & surge in popularity

2010: ImageNet Large Scale Visual Recognition Challenge

- 10 million labeled images

First convolutional network for image classification



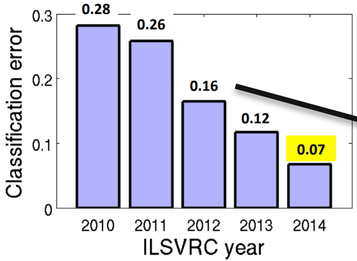
AlexNet [Krizhevsky '12]

AlexNet & surge in popularity

2010: ImageNet Large Scale Visual Recognition Challenge

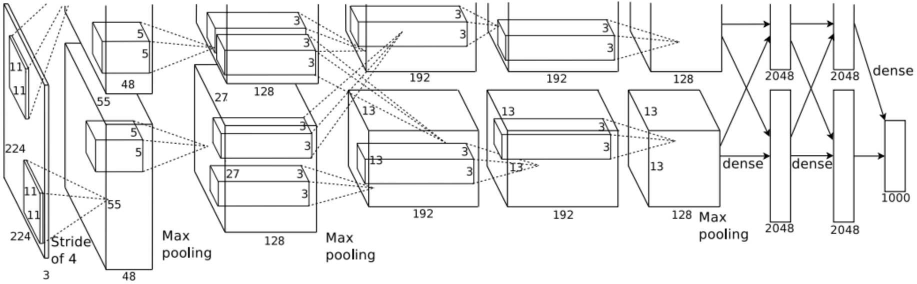
- 10 million labeled images

First convolutional network for image classification



[Russakovsky ’15]

CNNs

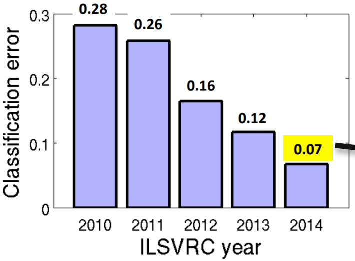


AlexNet [Krizhevsky ‘12]

AlexNet & surge in popularity

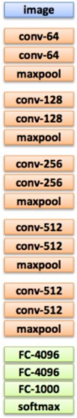
2010: ImageNet Large Scale Visual Recognition Challenge

- 10 million labeled images



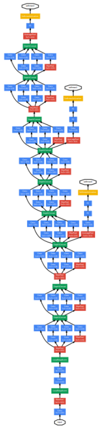
[Russakovsky ’15]

Deep networks



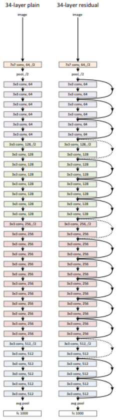
VGG

[Simonyan ‘14]



GoogLeNet

[Szegedy ‘14]

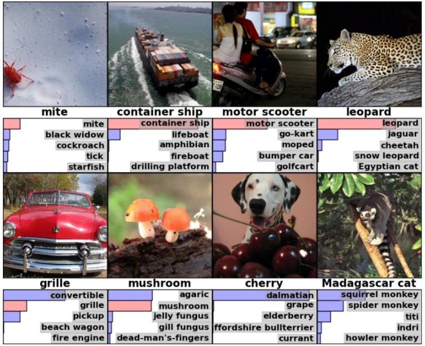


ResNet

[He ‘15]

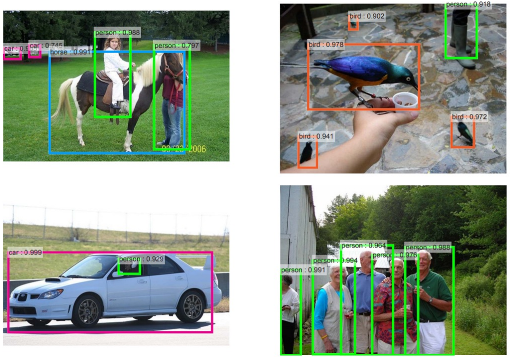
Image Classification

[Krizhevsky '12]



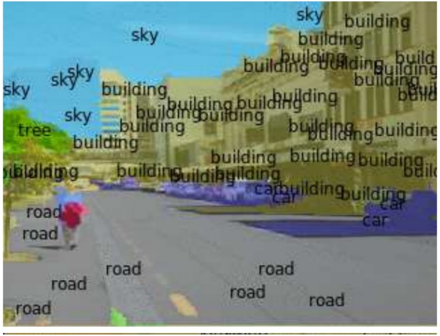
Object Detection

[Ren '16]



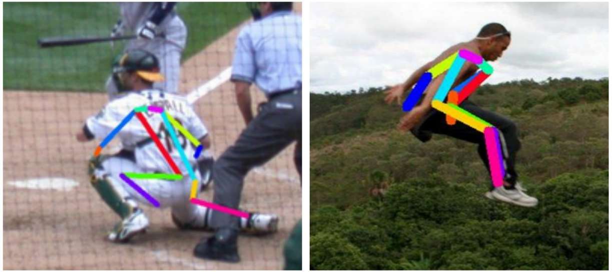
Segmentation

[Farabet '13]



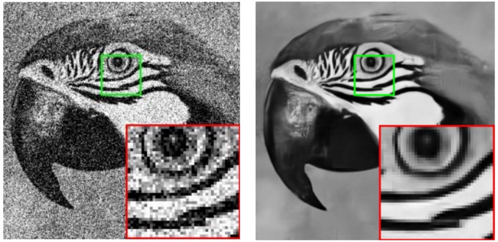
Pose estimation

[Toshev '14]



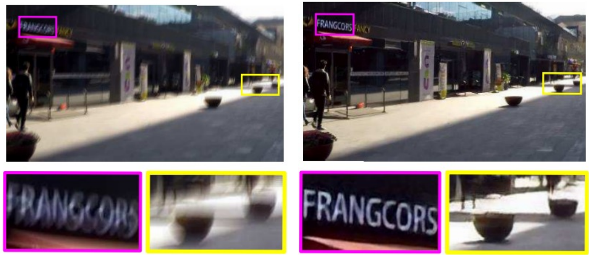
Imaging & Image processing

Image Denoising



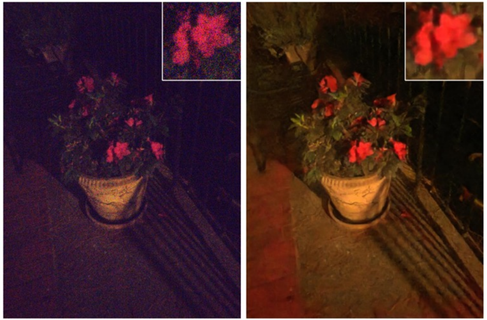
[Zhang ‘17]

Image Deblurring



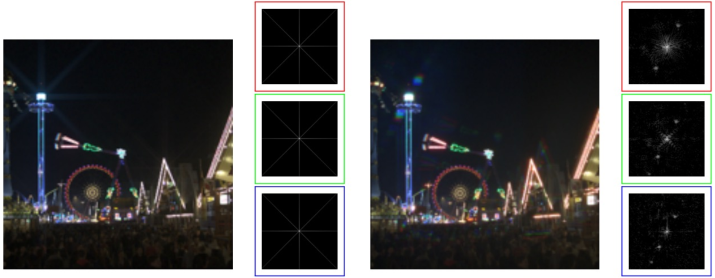
[Nah ‘17]

Learned ISPs



[Chen ‘18]

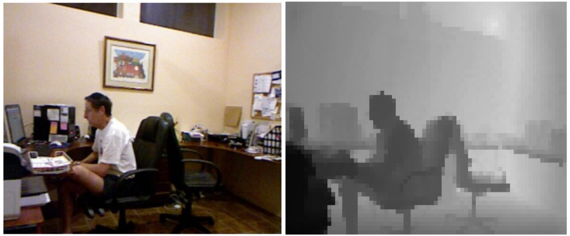
End-to-End Optimization



[Metzler ‘19]

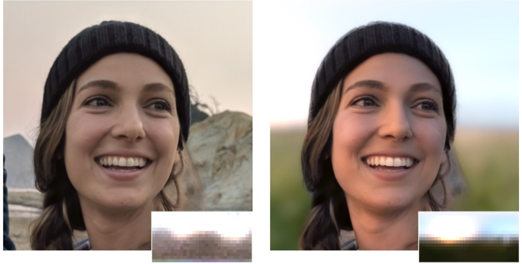
Imaging & Image processing

Monocular Depth Estimation



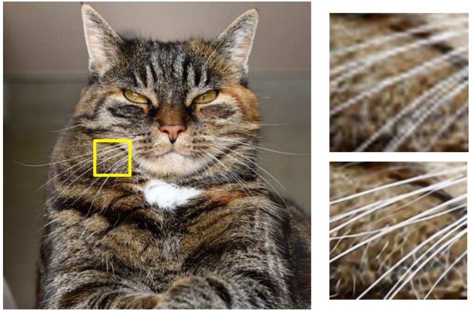
[Eigen '14]

Image Relighting



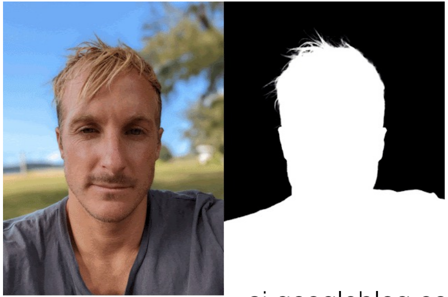
[Sun '19]

Image Super-resolution



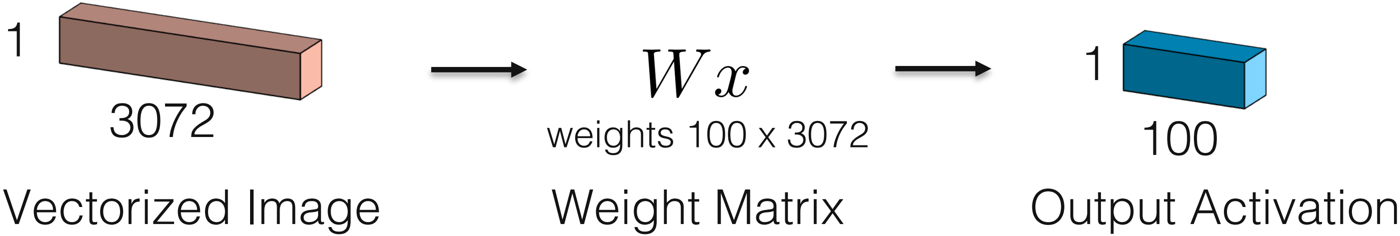
[Lim '17]

Synthetic Depth-of-Field



ai.googleblog.com

Fully-Connected Layer



1

3072

Vectorized Image

Wx

weights 100 x 3072

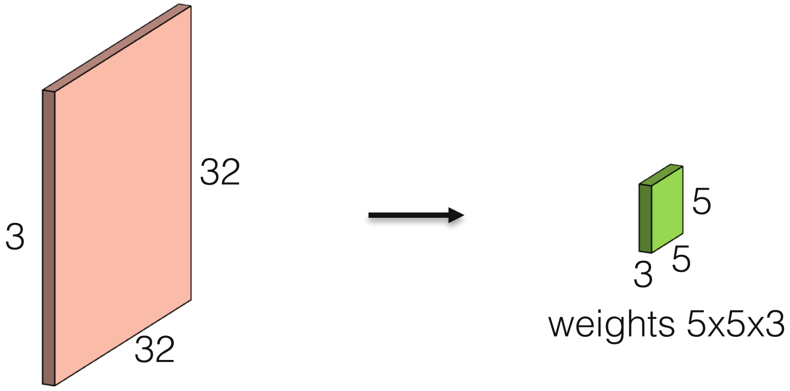
Weight Matrix

1

100

Output Activation

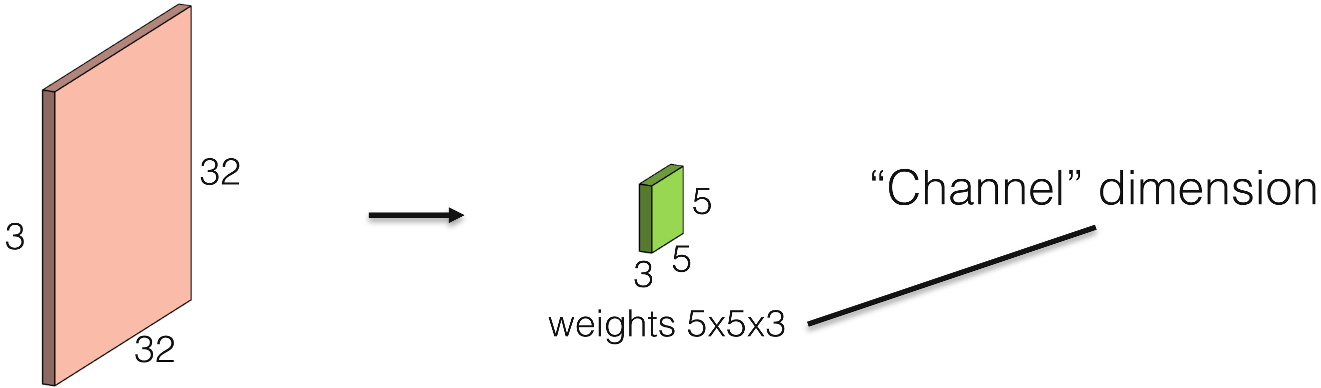
Convolutional Layer



Input Image

Filter

Convolutional Layer

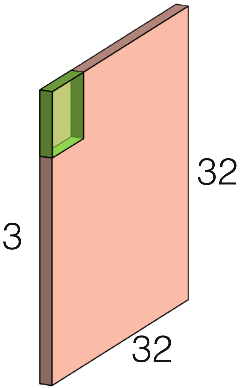


“Channel” dimension

Input Image

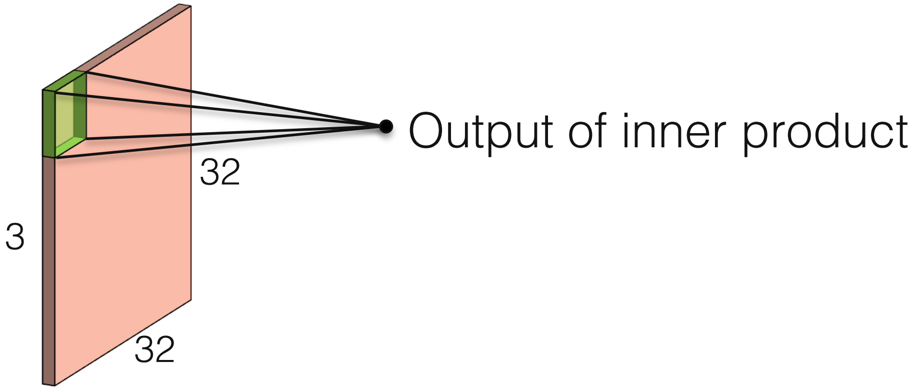
Filter

Convolutional Layer



Input Image

Convolutional Layer



Output of inner product

32

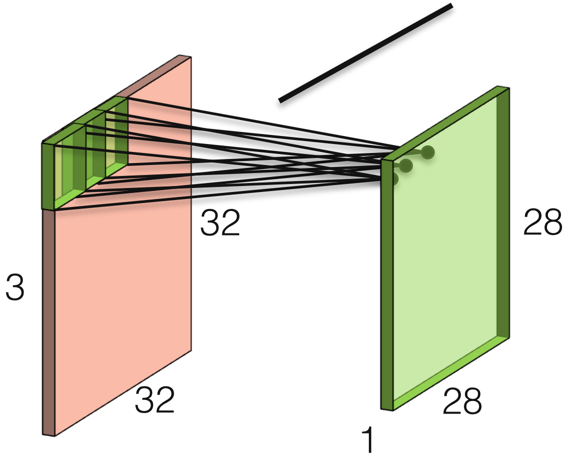
3

32

Input Image

Convolutional Layer

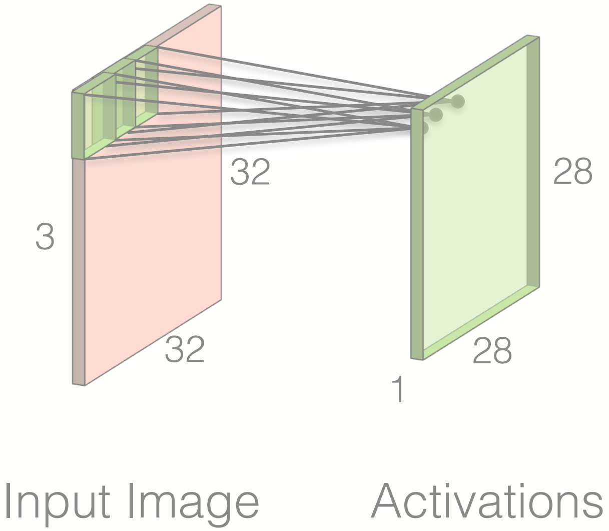
Convolution = sliding window + inner product

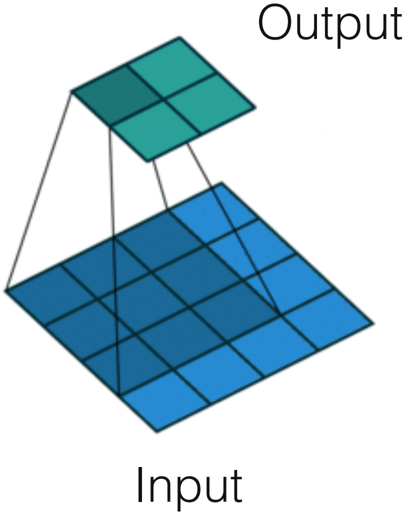


Input Image

Activations

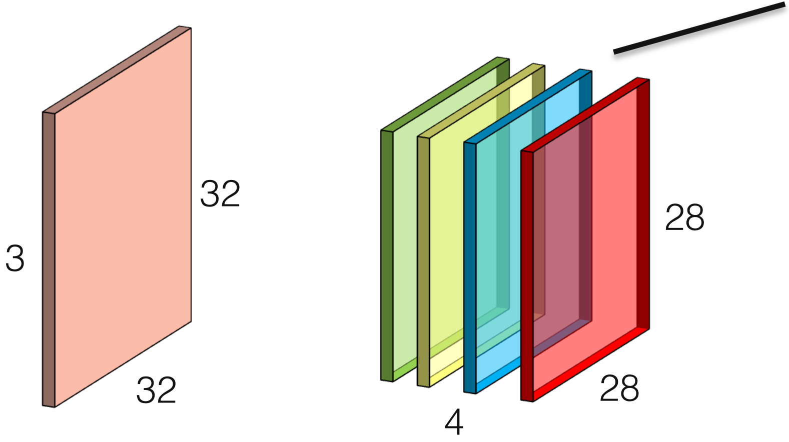
Convolutional Layer





https://github.com/vdumoulin/conv_arithmetic

Convolutional Layer

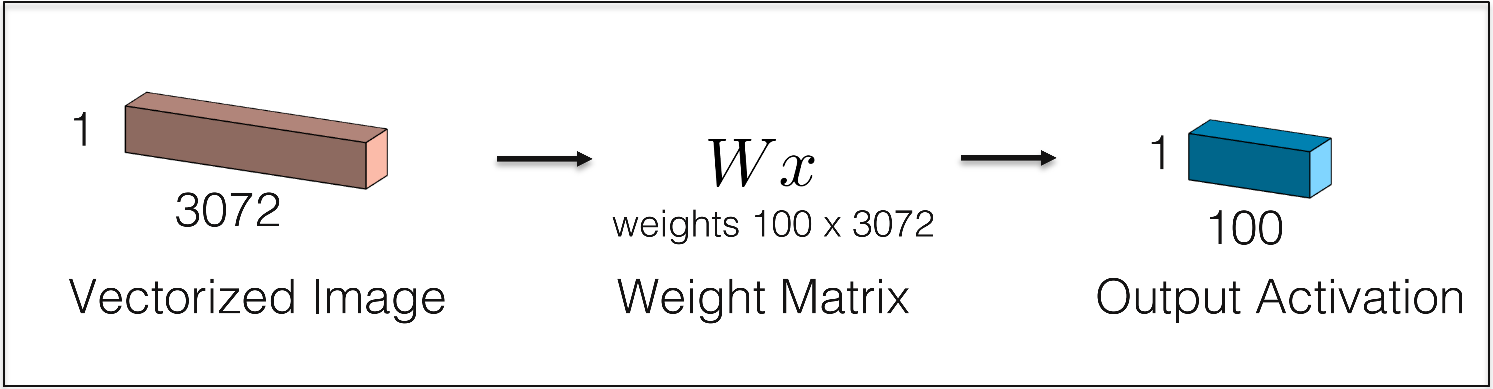


Multiple output channels using multiple filters

Input Image

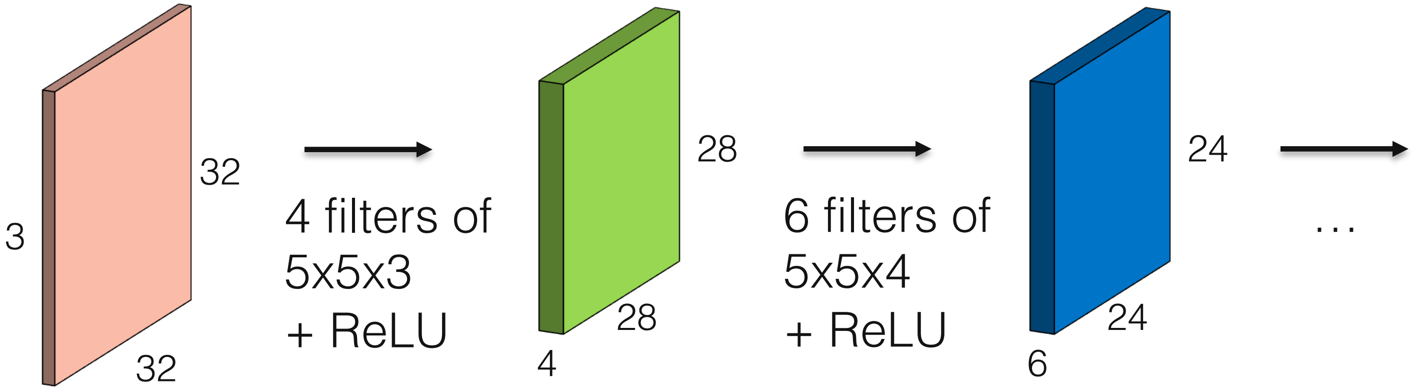
Activations

Fully-Connected Layer



Special case of convolutional layer when filter size = input size!

Convolutional Neural Network



4 filters of
5x5x3
+ ReLU

6 filters of
5x5x4
+ ReLU

Input Image

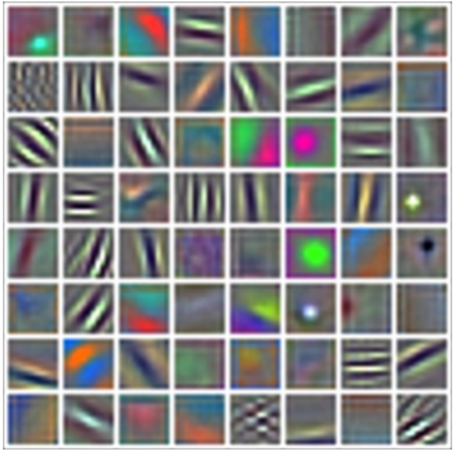
Layer 1
Activations

Layer 2
Activations

Case Study: AlexNet

Input Image

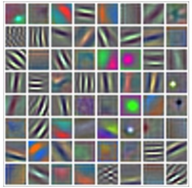
First-layer Filters

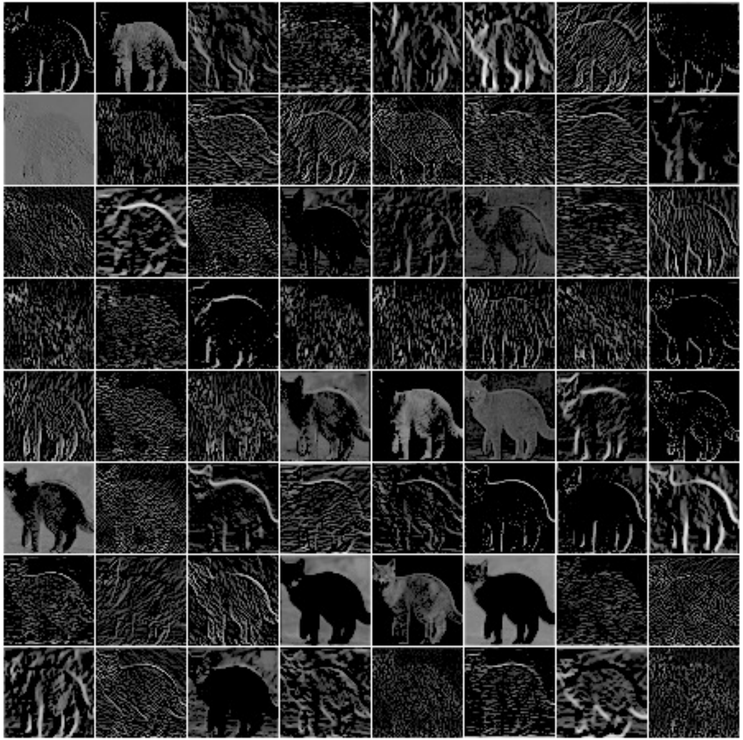


Case Study: AlexNet

Input Image

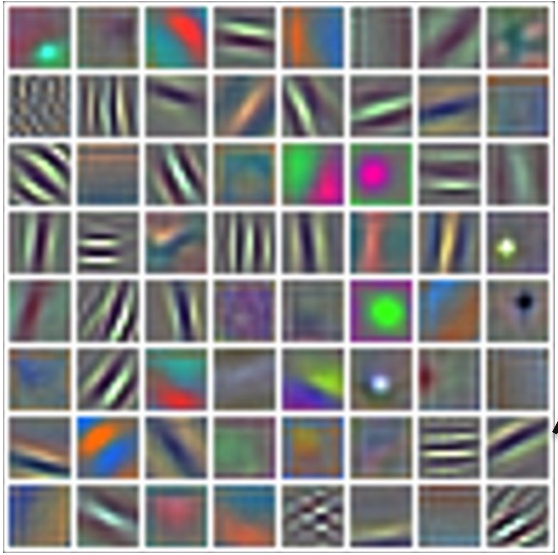
First-layer Filters





Activations

Case Study: AlexNet



First-layer Filters

Similar to simple cells in visual cortex!

- Edge detectors

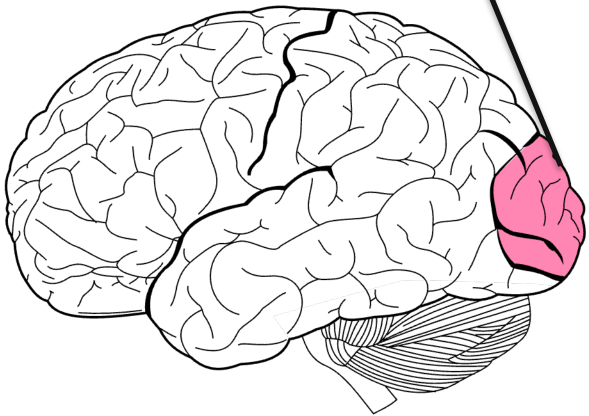
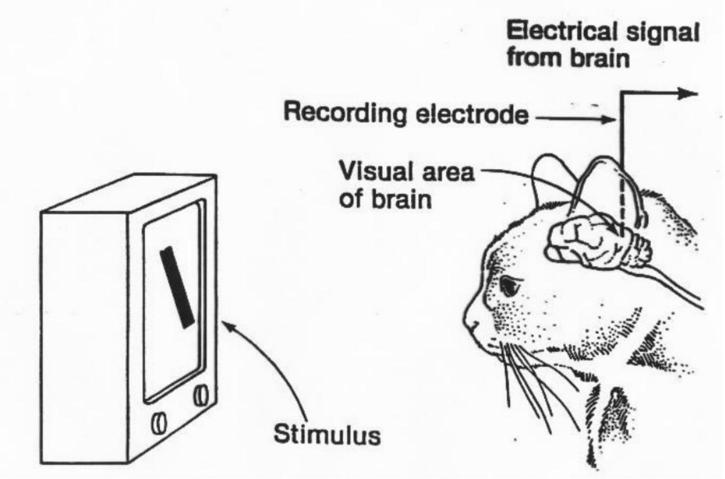


Image: CC BY-SA Selket

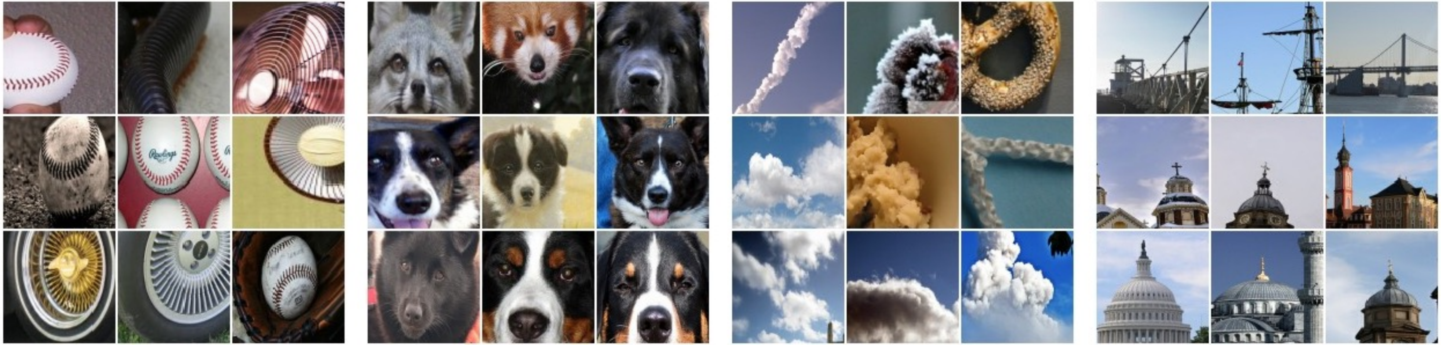
Case Study: AlexNet

[Hubel & Wiesel 1959]



Simple cells in visual cortex detect edges, complex cells compose earlier responses

CNN higher layer filters



[Olah ‘17]

Dataset examples that maximize neuron outputs

CNN Building Blocks

Design choices:

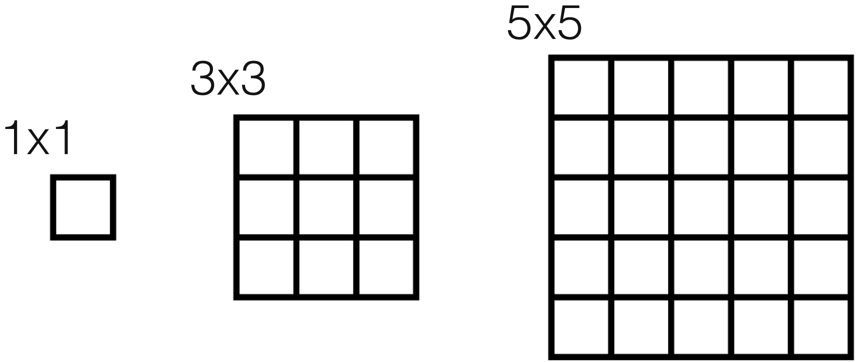
- filter size
- number of filters
- padding
- stride

Layer types:

- pooling
- transpose convolutions
- upsampling layers

CNN Building Blocks

Filter size



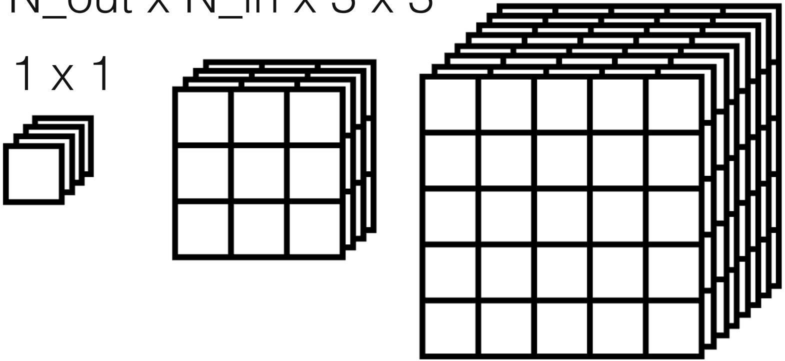
1x1

3x3

5x5

CNN Building Blocks

Number of channels



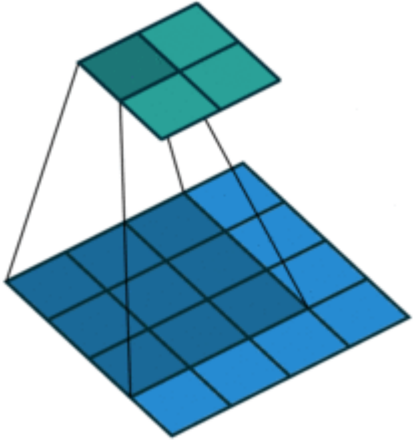
N_out x N_in x 1 x 1

N_out x N_in x 3 x 3

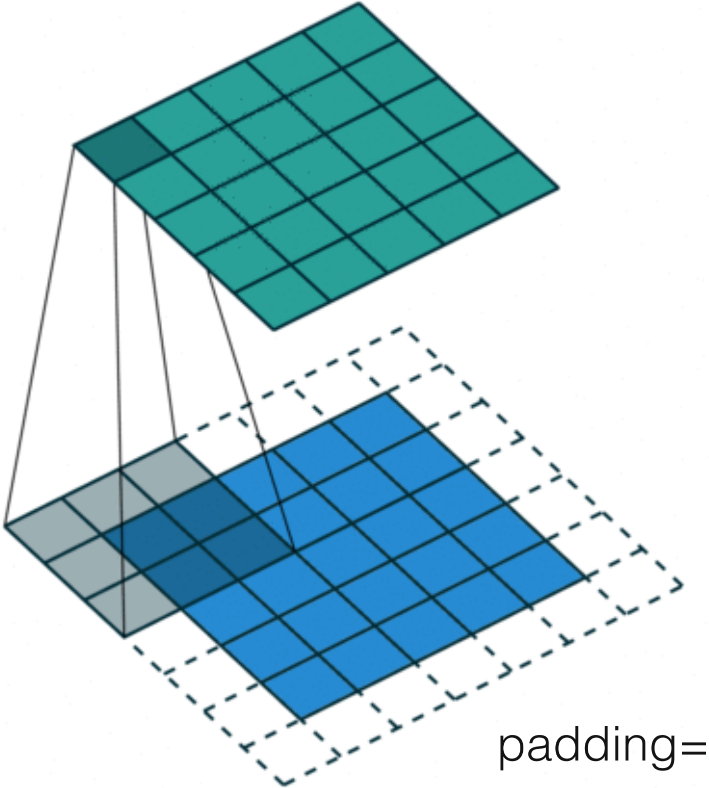
N_out x N_in x 5 x 5

CNN Building Blocks

padding



No padding

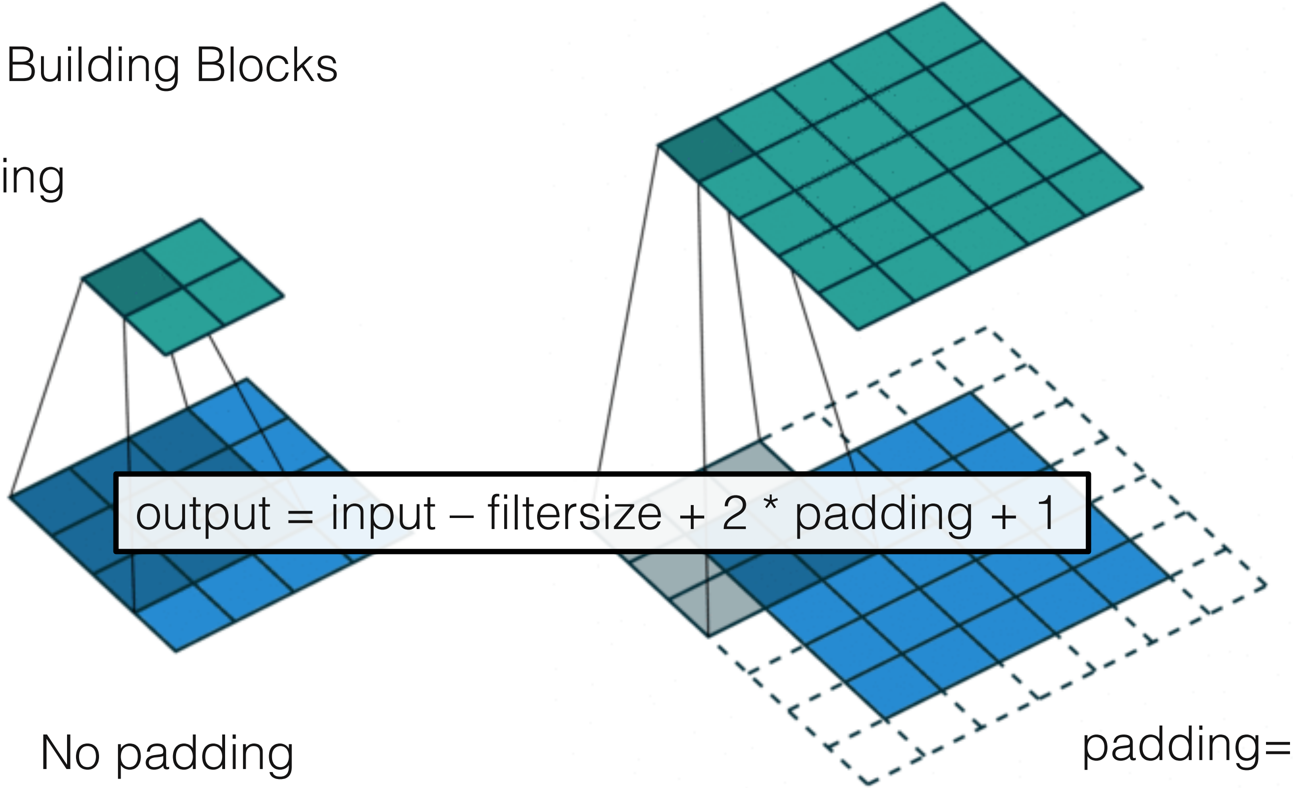


padding=1

https://github.com/vdumoulin/conv_arithmetic

CNN Building Blocks

padding



output = input – filtersize + 2 * padding + 1

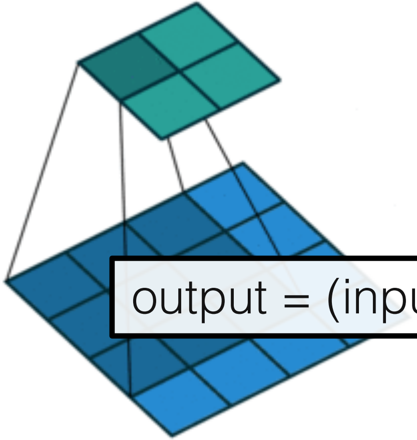
No padding

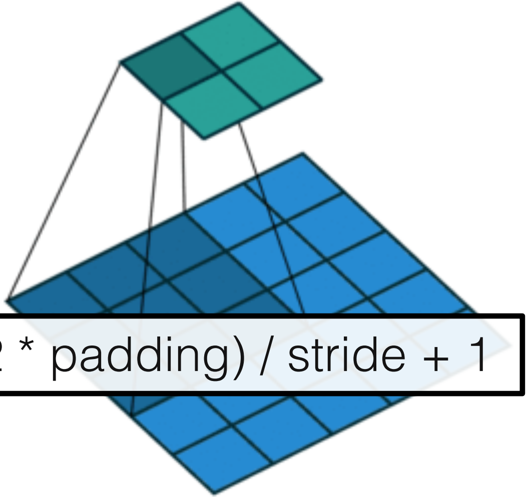
padding=1

https://github.com/vdumoulin/conv_arithmetic

CNN Building Blocks

stride





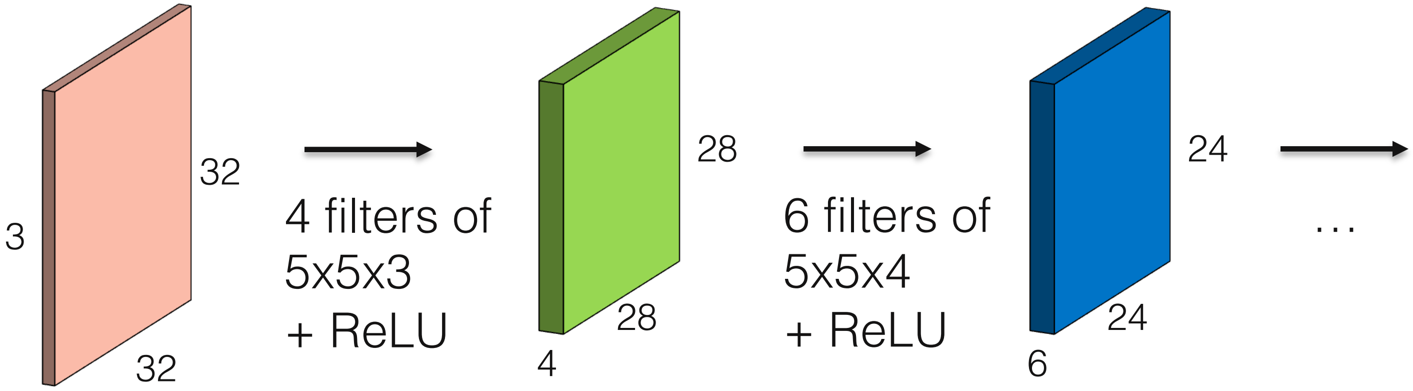
output = (input – filtersize + 2 * padding) / stride + 1

stride = 1

stride = 2

https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Network



4 filters of
5x5x3
+ ReLU

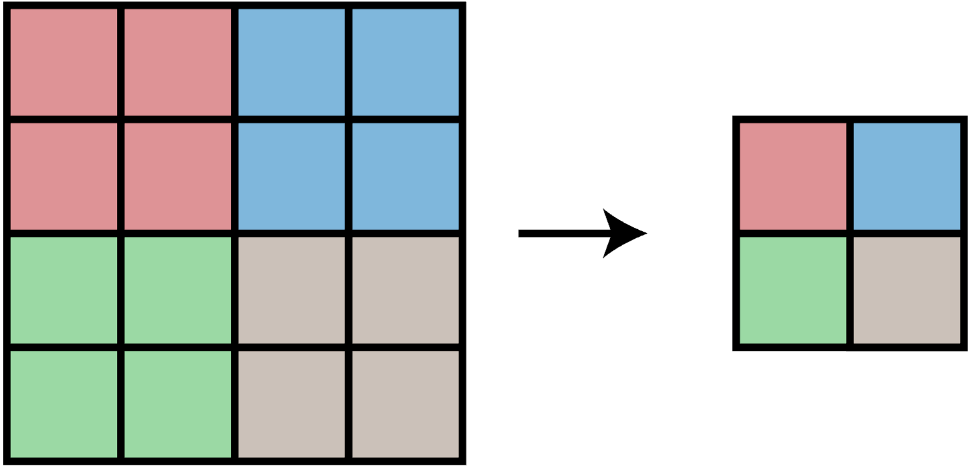
6 filters of
5x5x4
+ ReLU

Input Image

Layer 1
Activations

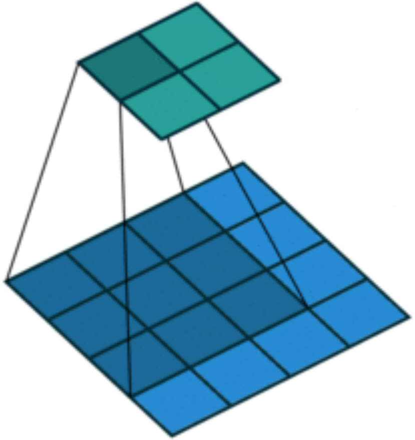
Layer 2
Activations

Layer types: Pooling

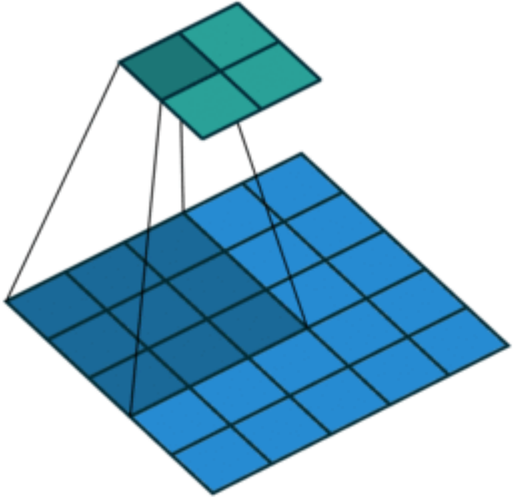


e.g., max pool size=2, stride=2

Transpose Convolution



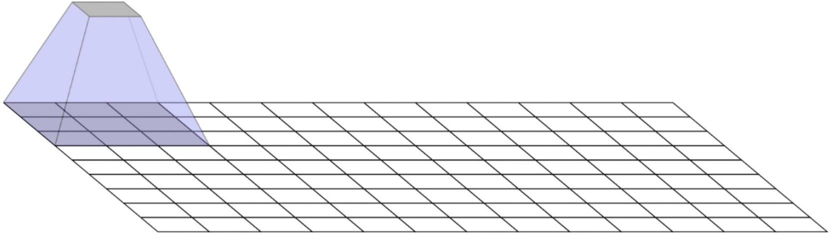
stride=1

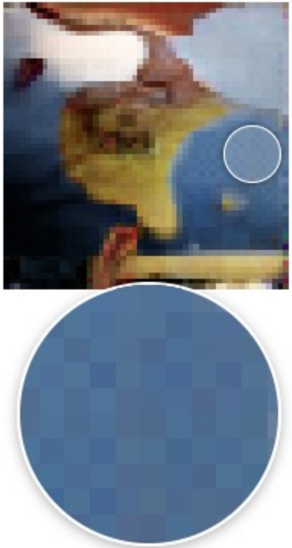


stride=2

https://github.com/vdumoulin/conv_arithmetic

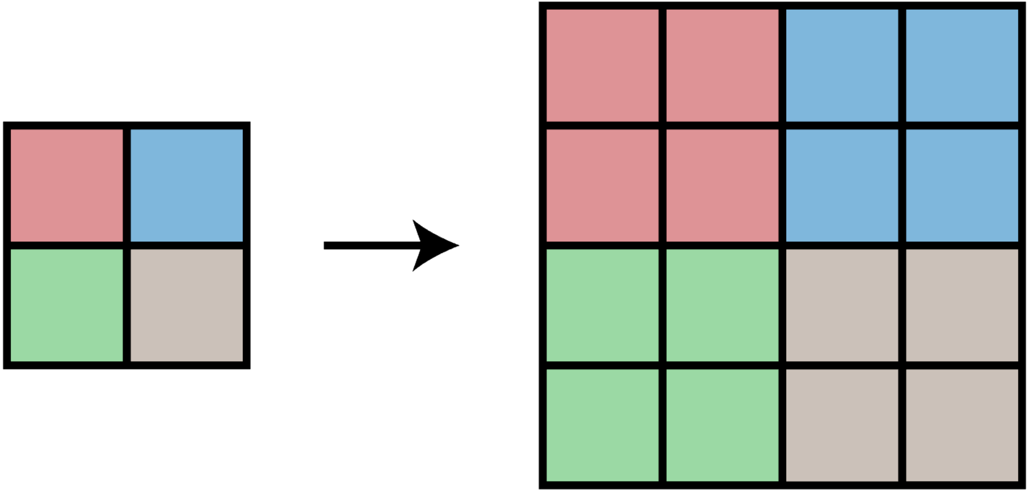
Transpose Convolution (checkerboard artifacts)





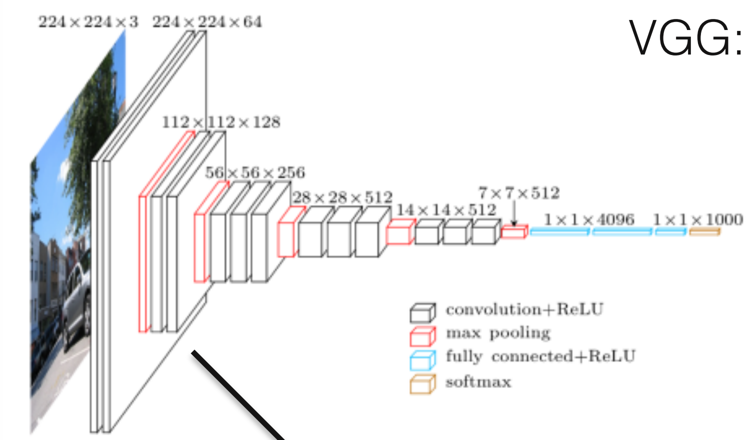
[Odena ‘16]

Upsampling layers



e.g., bilinear upsampling, nearest neighbor upsampling

Common Network Architectures



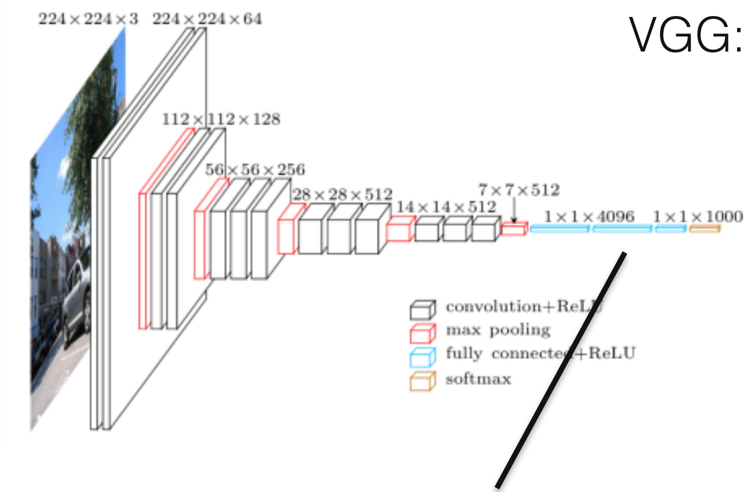
VGG: one of the first “deep” CNNs

downsampling with max pooling

Image: Davi Frossard

Common Network Architectures

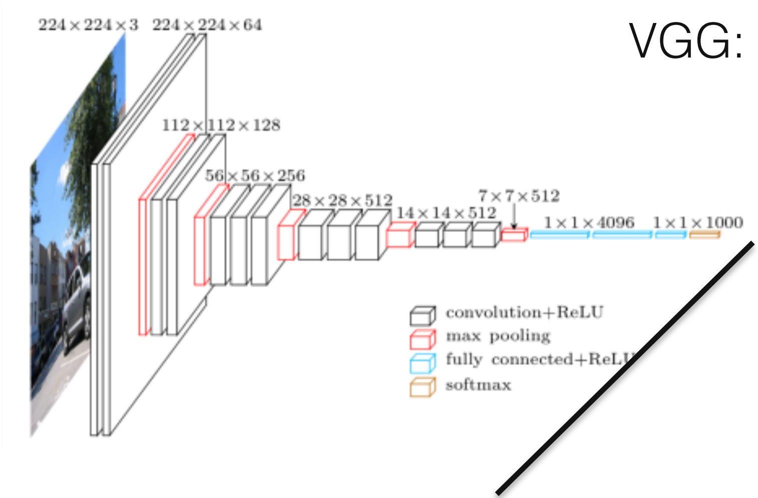
VGG: one of the first “deep” CNNs



Classification scores output with fully-connected layers

Image: Davi Frossard

Common Network Architectures



VGG: one of the first “deep” CNNs

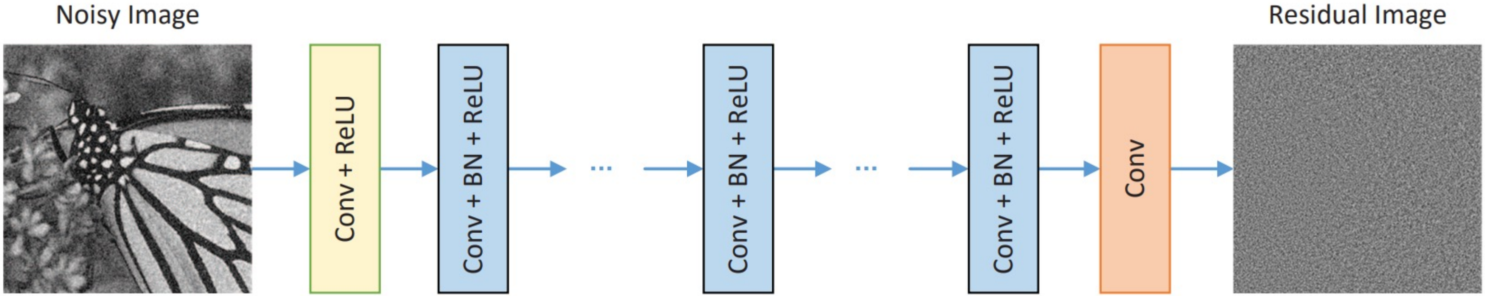
Not suitable for image processing…

Image: Davi Frossard

Today

- What is a neural network?
- Training/optimizing neural nets
- Why “neural”?
- Convolutional neural networks
- Applications & inverse problems

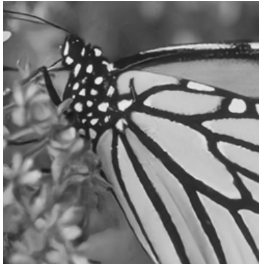
Image denoising with DnCNN

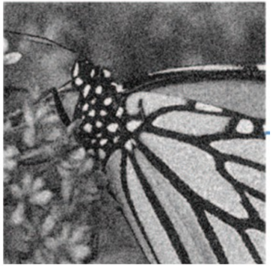


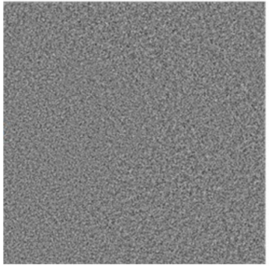
[Zhang ‘16]

Key ideas: residual learning & batch normalization

Residual Learning



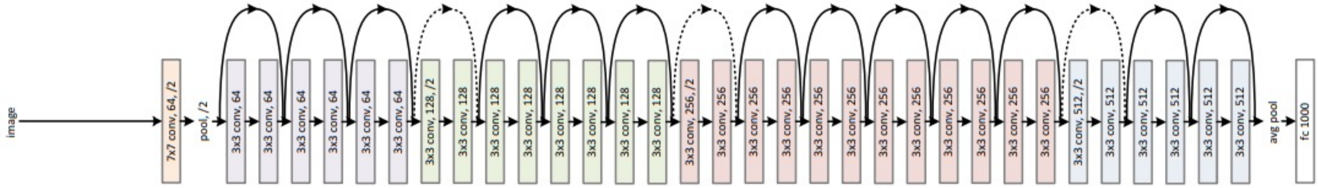




Clean image = noisy image - estimated noise

[Zhang ‘16]

Residual Learning



[He ‘15]

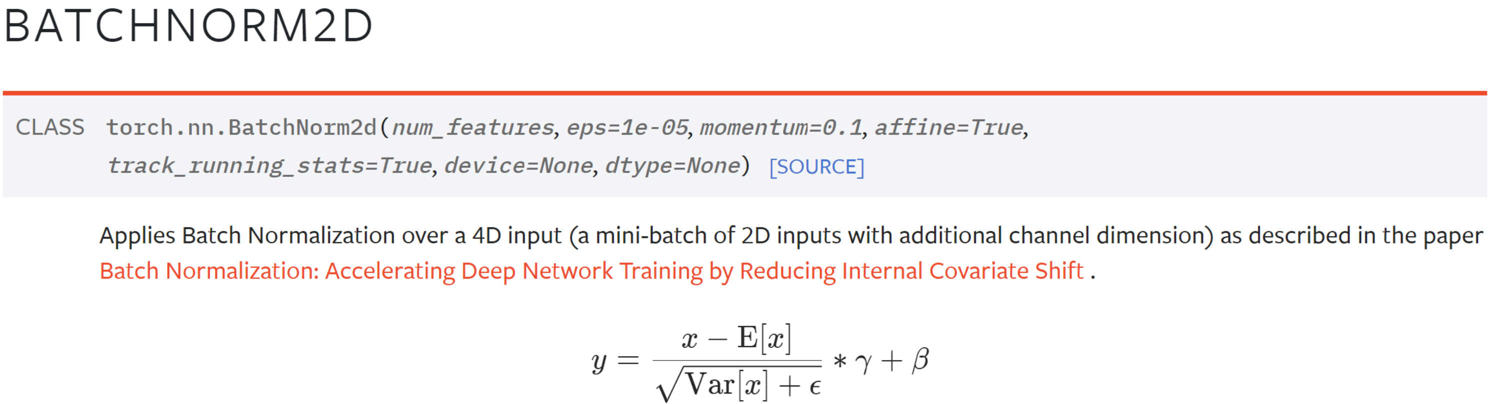
Popularized by residual nets “ResNets” for image classification

- Usually easier to optimize
- Better classification accuracy, good for many tasks!

Batch Normalization

Normalizes layer activations to zero mean, unit variance, preventing distribution shifts during training

- can speed up and stabilize training
- prevent “internal covariate shift” or smooths out loss landscape



BATCHNORM2D

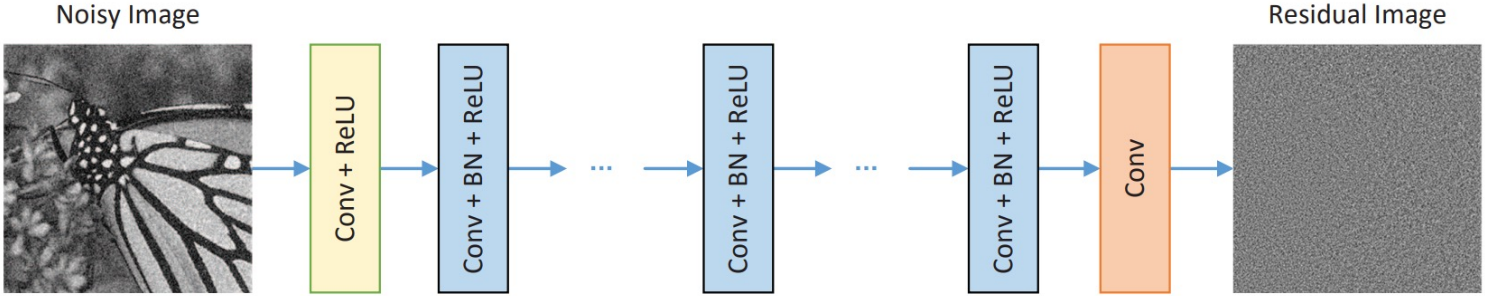
CLASS `torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True, device=None, dtype=None)` [SOURCE]

Applies Batch Normalization over a 4D input (a mini-batch of 2D inputs with additional channel dimension) as described in the paper Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift .

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html

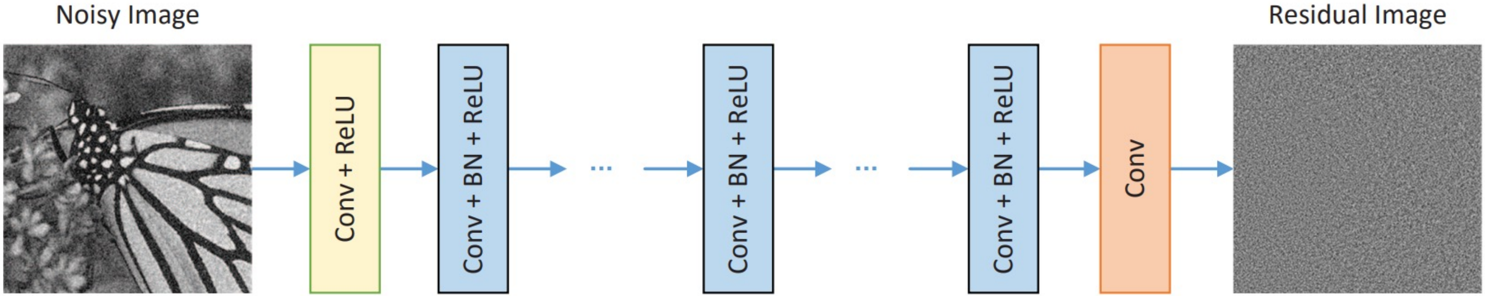
Image denoising with DnCNN



[Zhang ‘16]

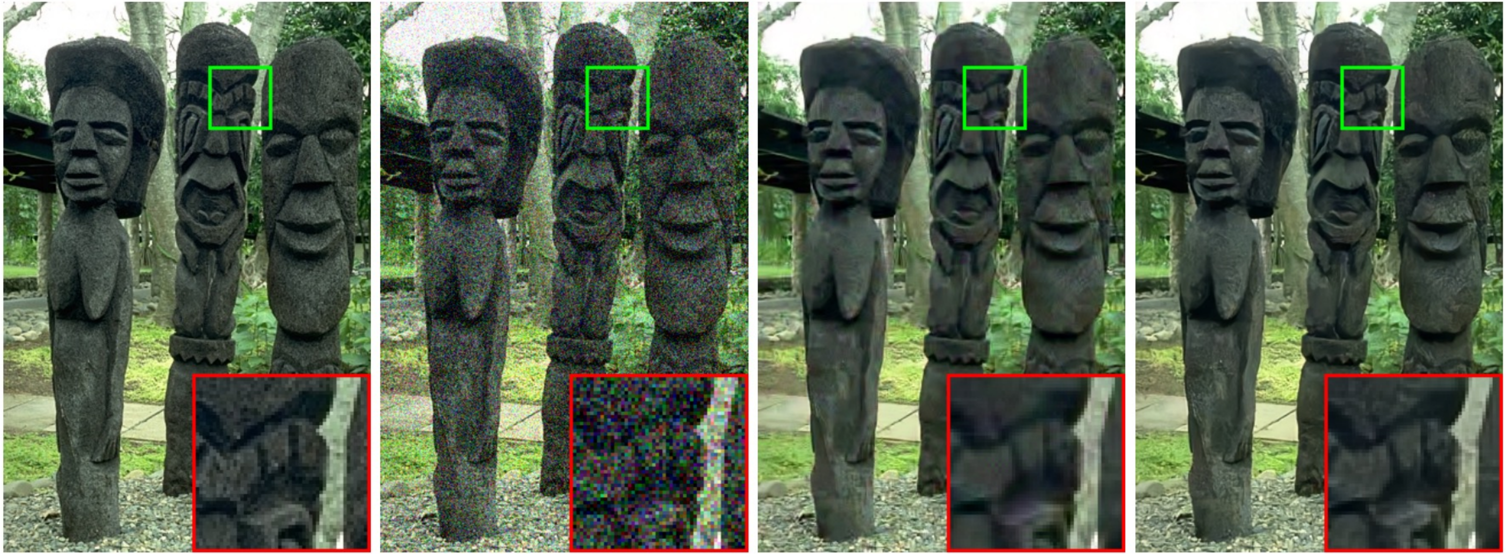
(Remember to disable the bias in your conv layer)

Image denoising with DnCNN



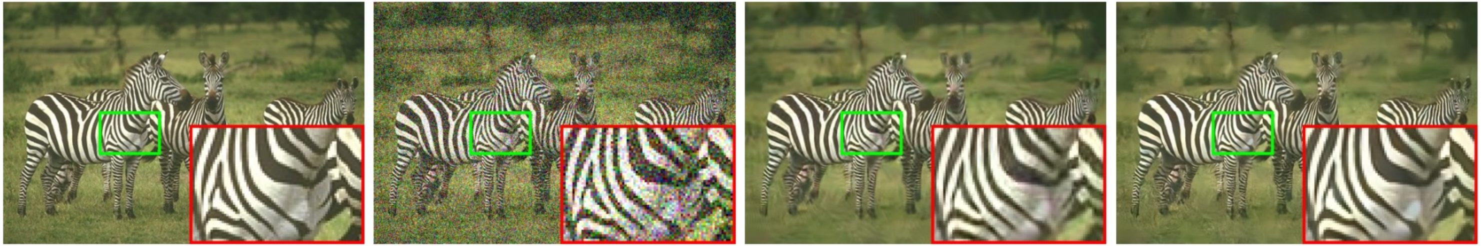
[Zhang ‘16]

No fully connected layers – can be applied to any input size



(a) Ground-truth (b) Noisy / 17.25dB (c) CBM3D / 25.93dB (d) CDnCNN-B / 26.58dB

[Zhang ‘16]



(a) Ground-truth

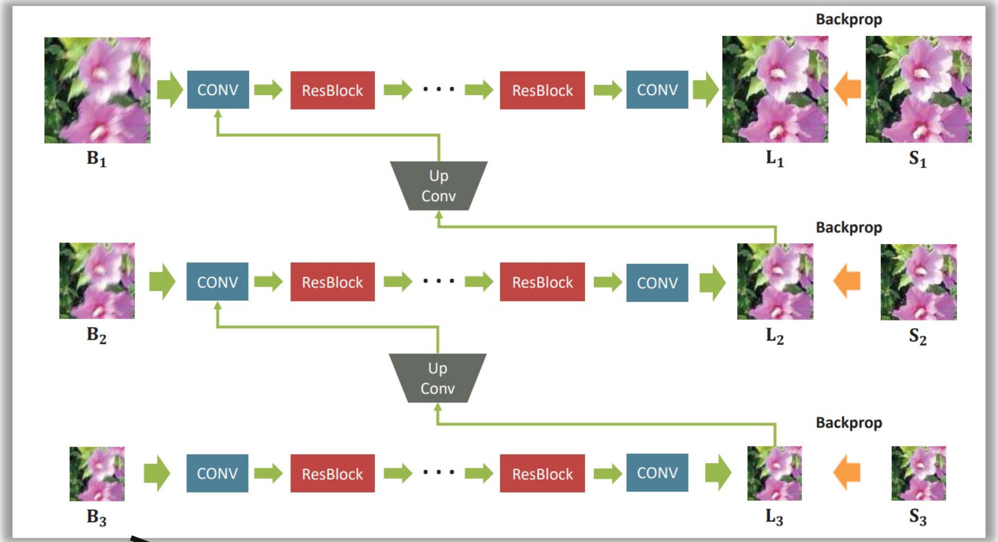
(b) Noisy / 15.07dB

(c) CBM3D / 26.97dB

(d) CDnCNN-B / 27.87dB

[Zhang ‘16]

Multi-Scale Architectures



Uses image pyramid to process & deblur

[Nah ‘18]

Multi-Scale Architectures

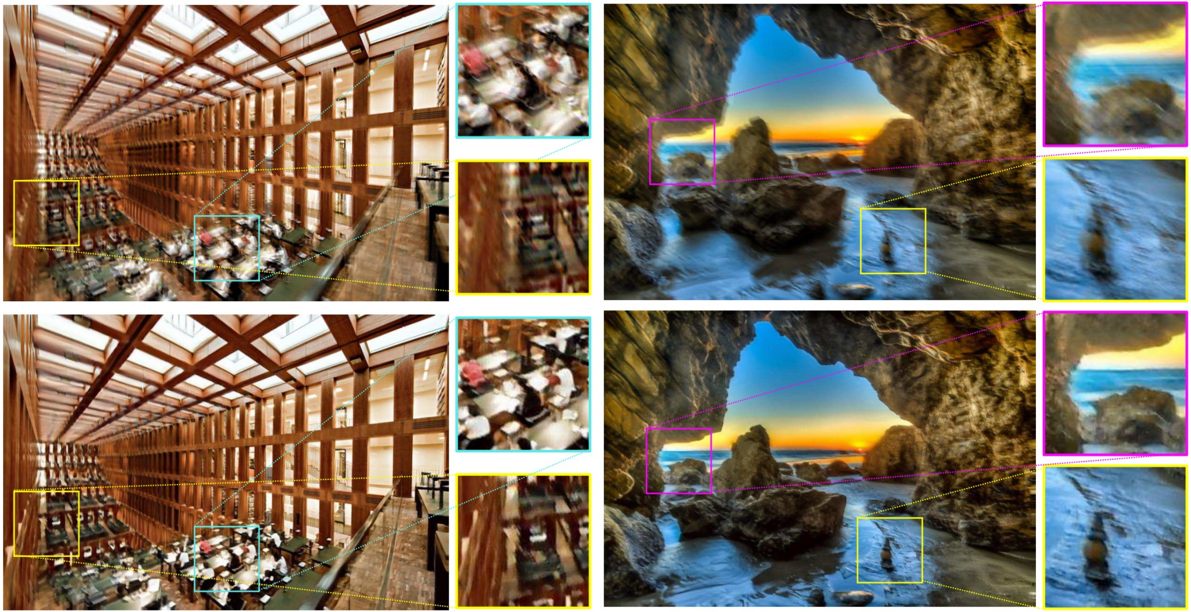
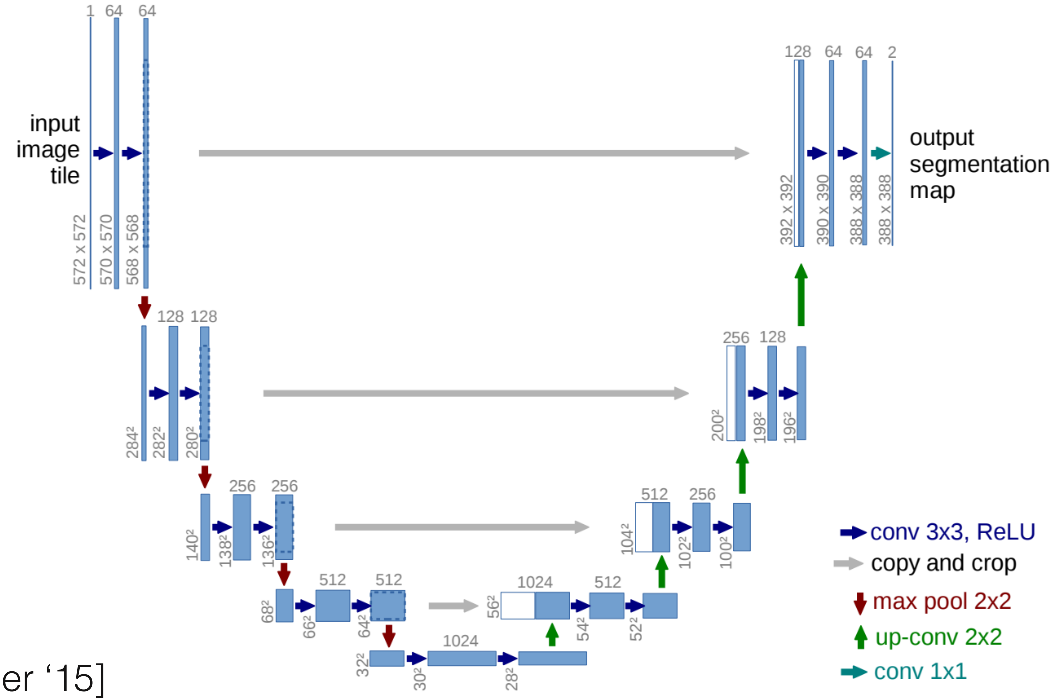


Figure 6. Deblurring results on the dataset [20]. The top row shows results of results of Sun et al. [26] and the bottom row shows our results.

[Nah '18]

U-Net: General-purpose architecture

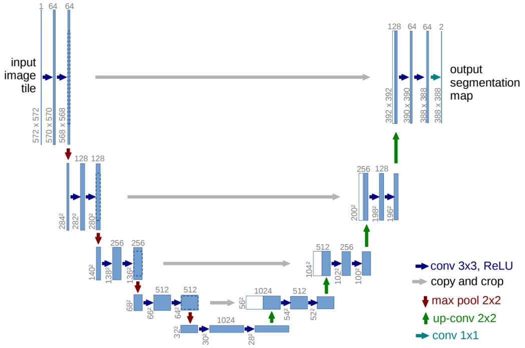


[Ronneberger '15]

U-Net: General-purpose architecture

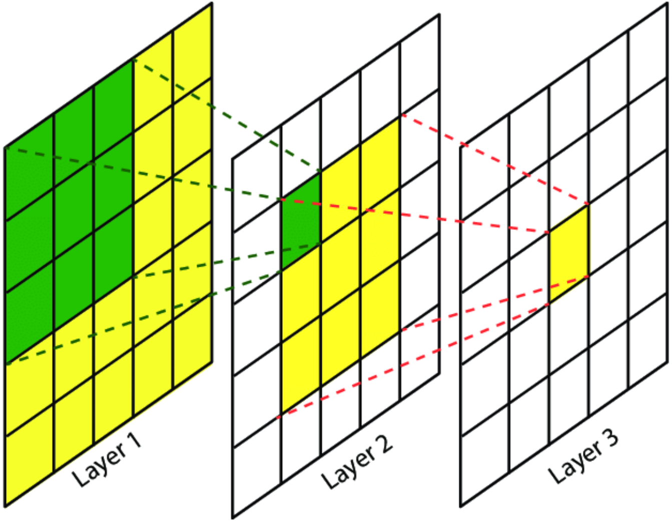
Introduced for biomedical image segmentation

- Uses residual connections
- Multi-scale processing (captures details at different scales)
- Large receptive field!



U-Net: General-purpose architecture

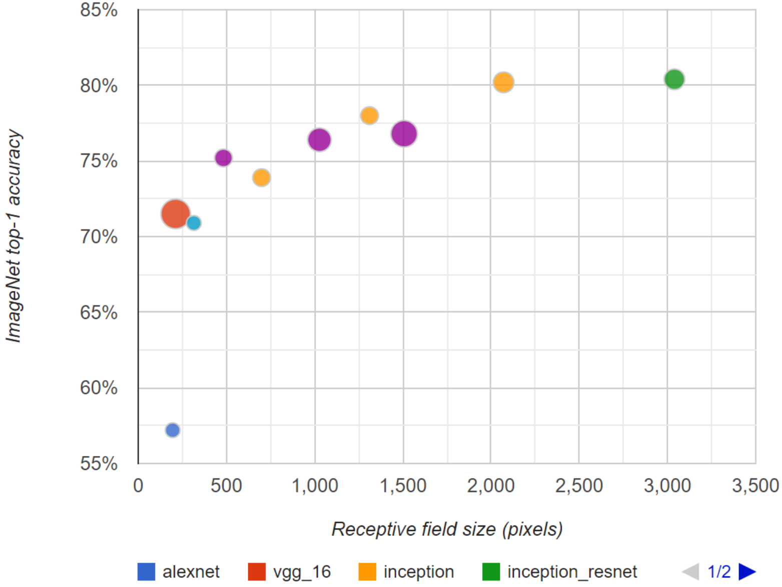
Receptive field: size of the input that contributes to the activation/output value



[Lin ‘17]

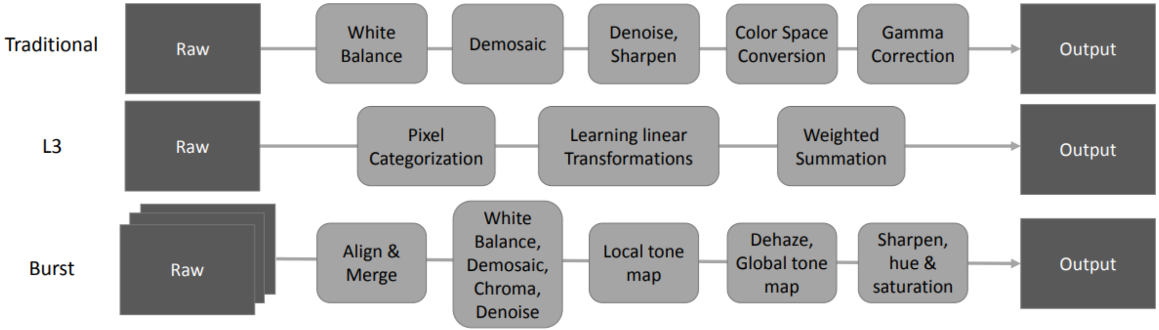
U-Net: General-purpose architecture

Large receptive field is important for high-level vision tasks and semantic understanding

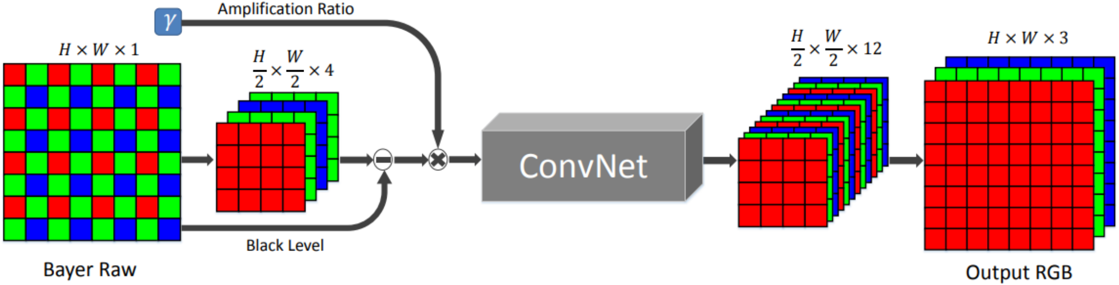


[Araujo '19]

Learned ISP

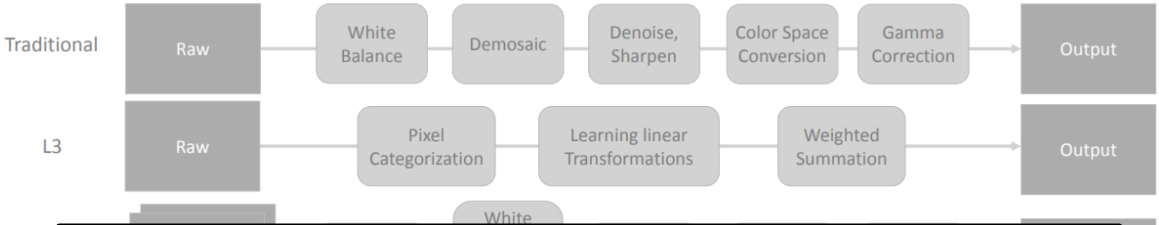


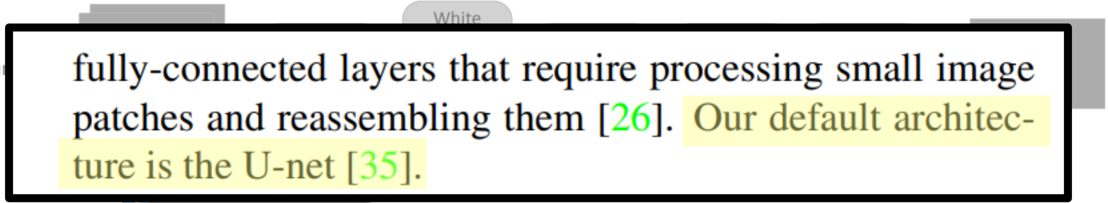
(a)



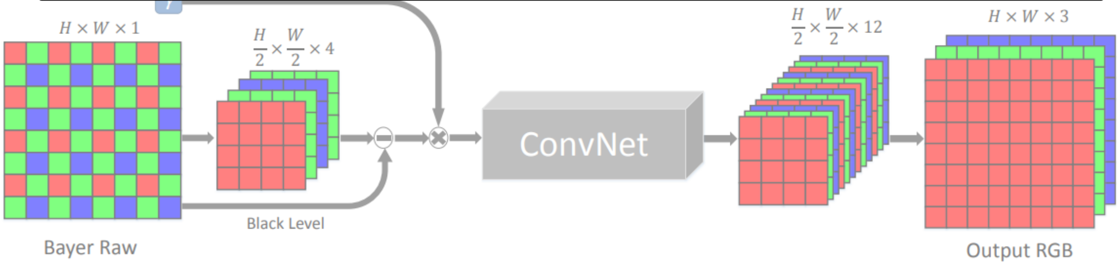
[Chen ‘18]

Learned ISP



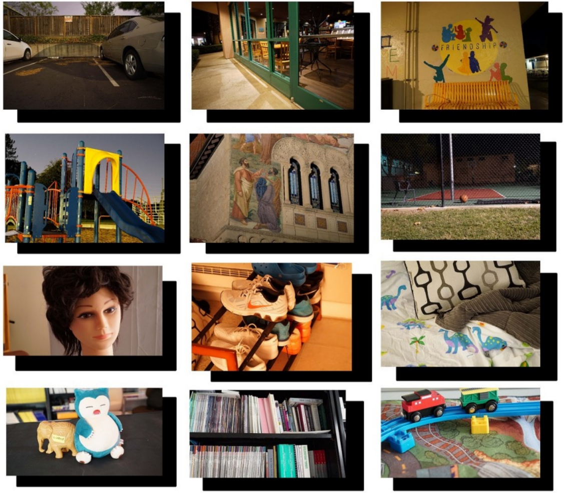


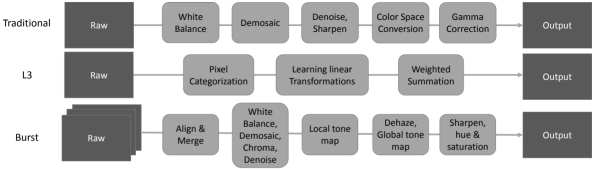
fully-connected layers that require processing small image patches and reassembling them [26]. Our default architecture is the U-net [35].


$H \times W \times 1$
$\frac{H}{2} \times \frac{W}{2} \times 4$
Black Level
ConvNet
$\frac{H}{2} \times \frac{W}{2} \times 12$
$H \times W \times 3$
Bayer Raw
Output RGB

[Chen '18]

Learned ISP



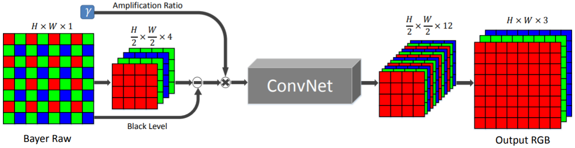


Traditional: Raw → White Balance → Demosaic → Denoise, Sharpen → Color Space Conversion → Gamma Correction → Output

L3: Raw → Pixel Categorization → Learning linear Transformations → Weighted Summation → Output

Burst: Raw → Align & Merge → White Balance, Demosaic, Chroma, Denoise → Local tone map → Dehaze, Global tone map → Sharpen, hue & saturation → Output

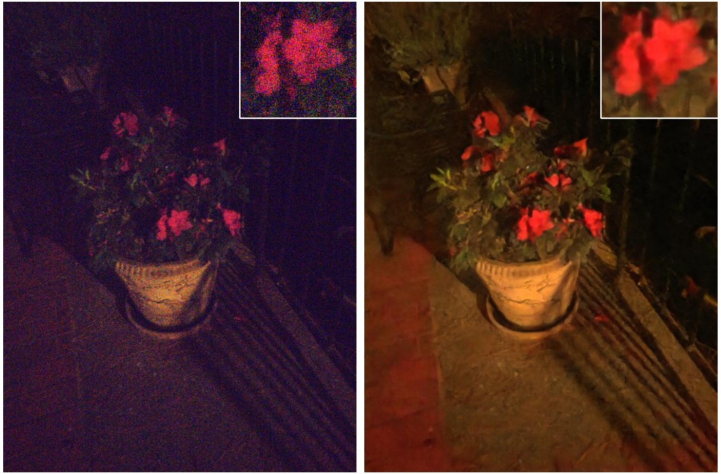
(a)



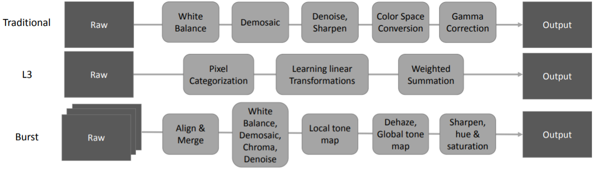
Trained on short-exposure (noisy) / long-exposure image pairs

[Chen '18]

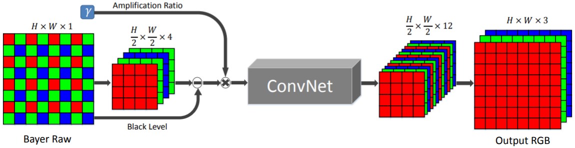
Learned ISP



(a) Traditional pipeline (b) Our result

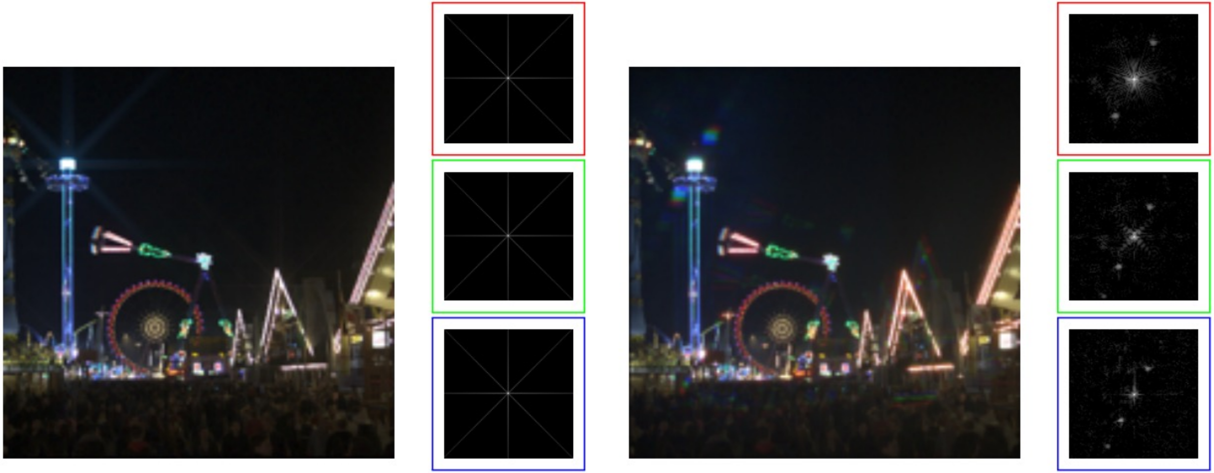


(a)



[Chen ‘18]

Deep optics for HDR imaging

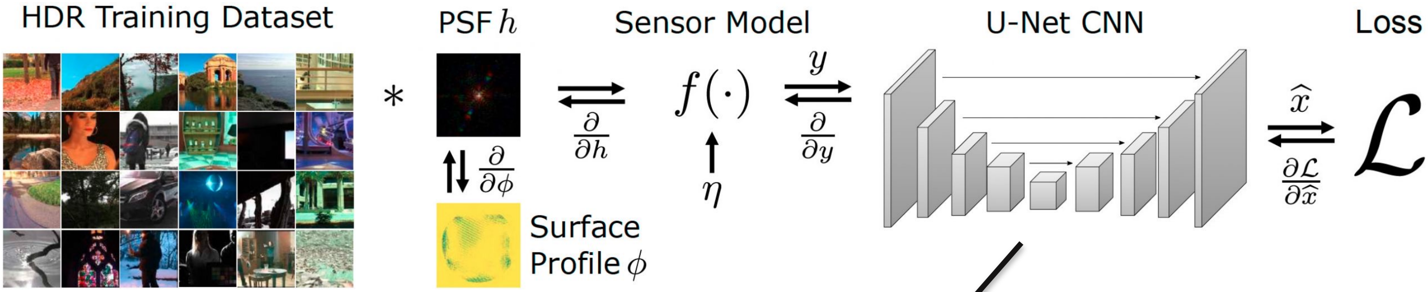


(a) Star PSF

(b) E2E PSF

[Metzler ‘20]

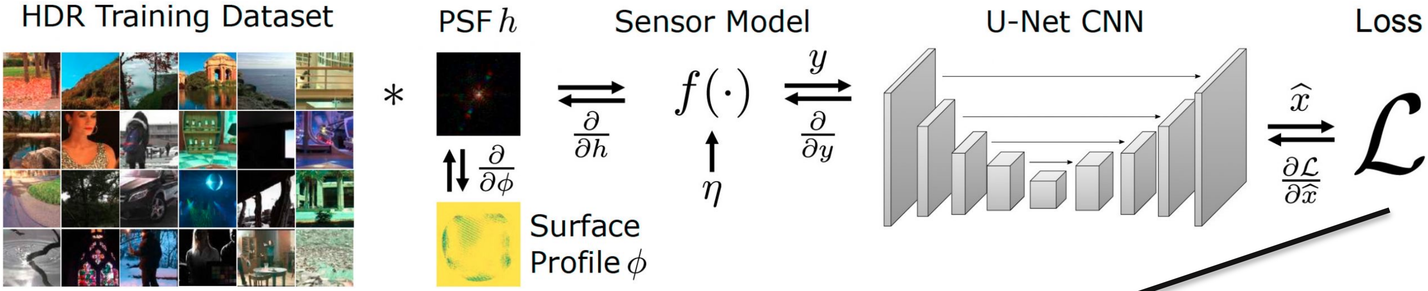
Deep optics for HDR imaging


HDR Training Dataset
*
PSF h
$\frac{\partial}{\partial \phi}$
Surface Profile ϕ
$\frac{\partial}{\partial h}$
Sensor Model
$f(\cdot)$
η
y
$\frac{\partial}{\partial y}$
U-Net CNN
$\widehat{x}$
$\frac{\partial \mathcal{L}}{\partial \widehat{x}}$
Loss
$\mathcal{L}$

U-Net again

[Metzler ‘20]

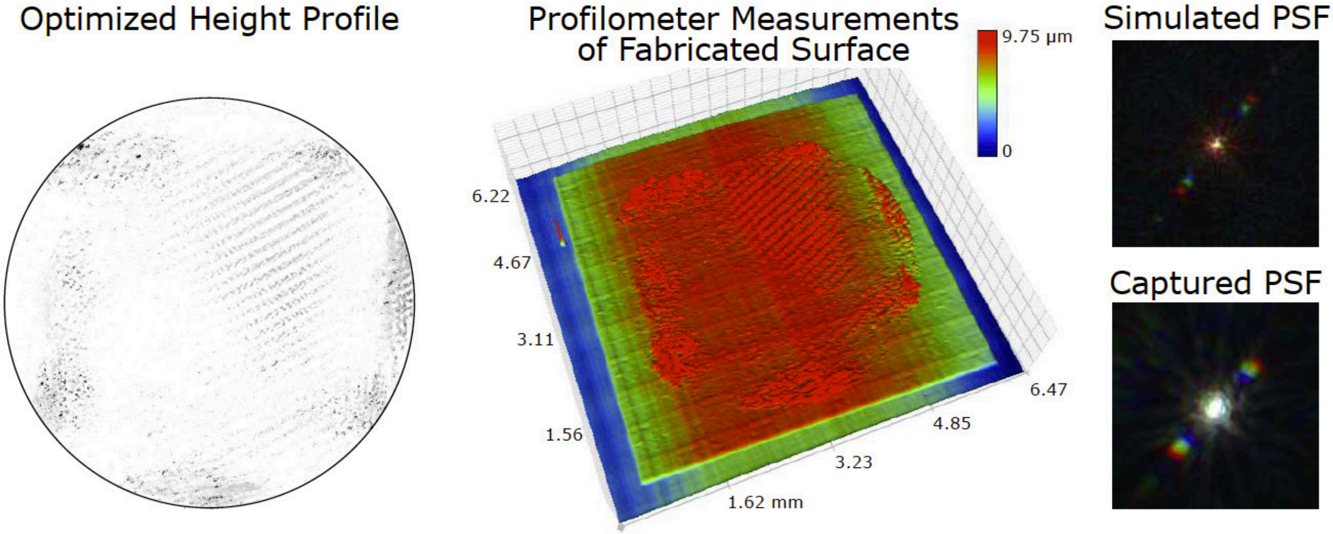
Deep optics for HDR imaging


HDR Training Dataset
$*$
PSF h
$\frac{\partial}{\partial \phi}$
Surface Profile ϕ
$\frac{\partial}{\partial h}$
Sensor Model
$f(\cdot)$
η
y
$\frac{\partial}{\partial y}$
U-Net CNN
$\widehat{x}$
$\frac{\partial \mathcal{L}}{\partial \widehat{x}}$
Loss
$\mathcal{L}$

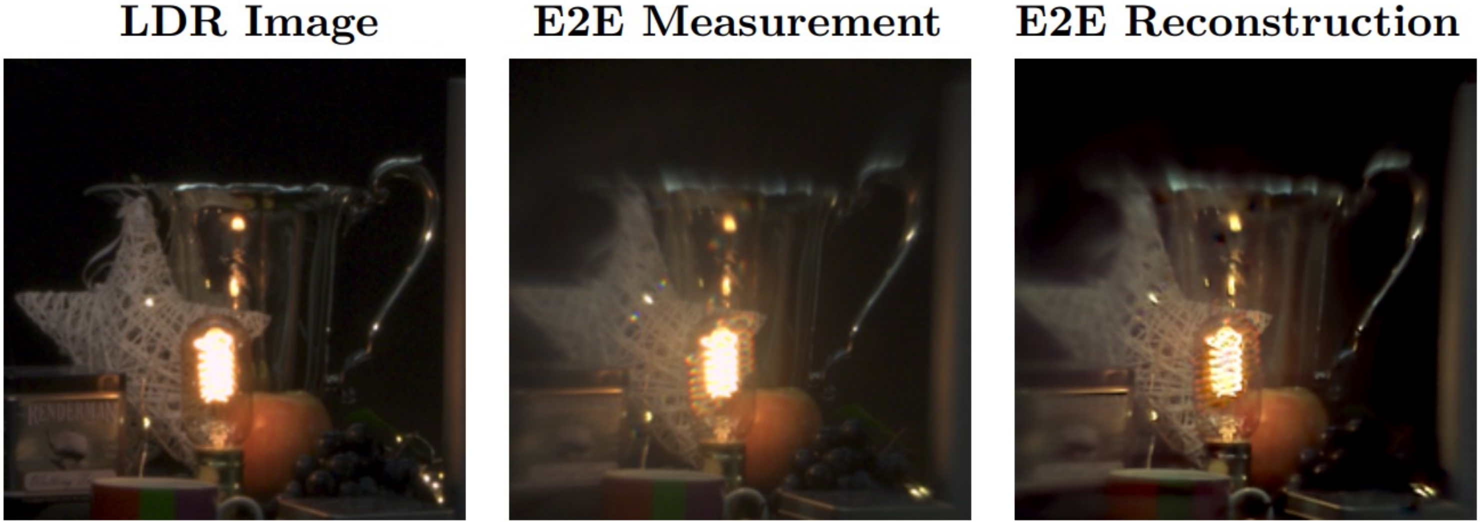
Minimize difference between reconstruction and tone-mapped GT images

[Metzler ‘20]

Deep optics for HDR imaging



[Metzler '20]



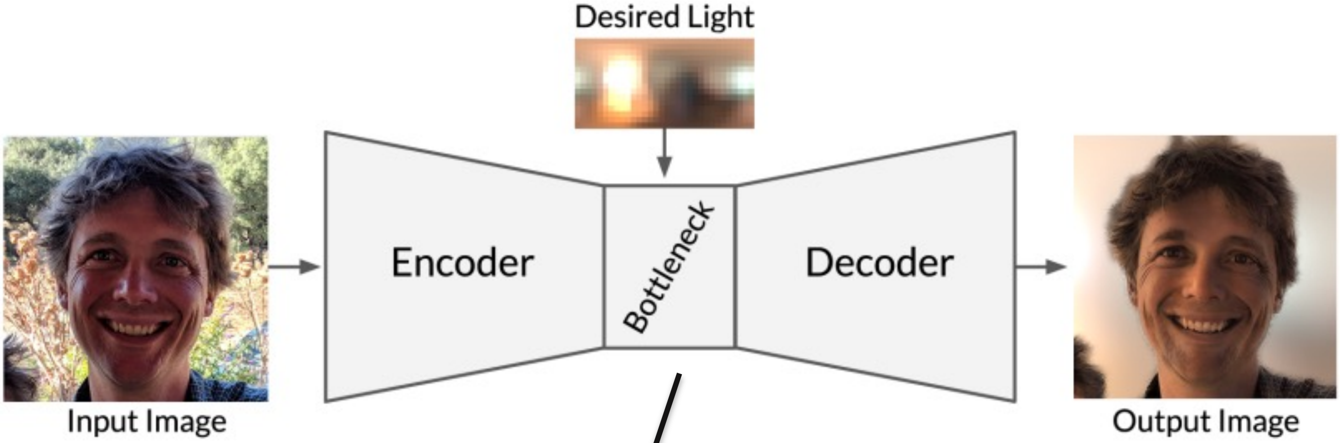
LDR Image

E2E Measurement

E2E Reconstruction

[Metzler ‘20]

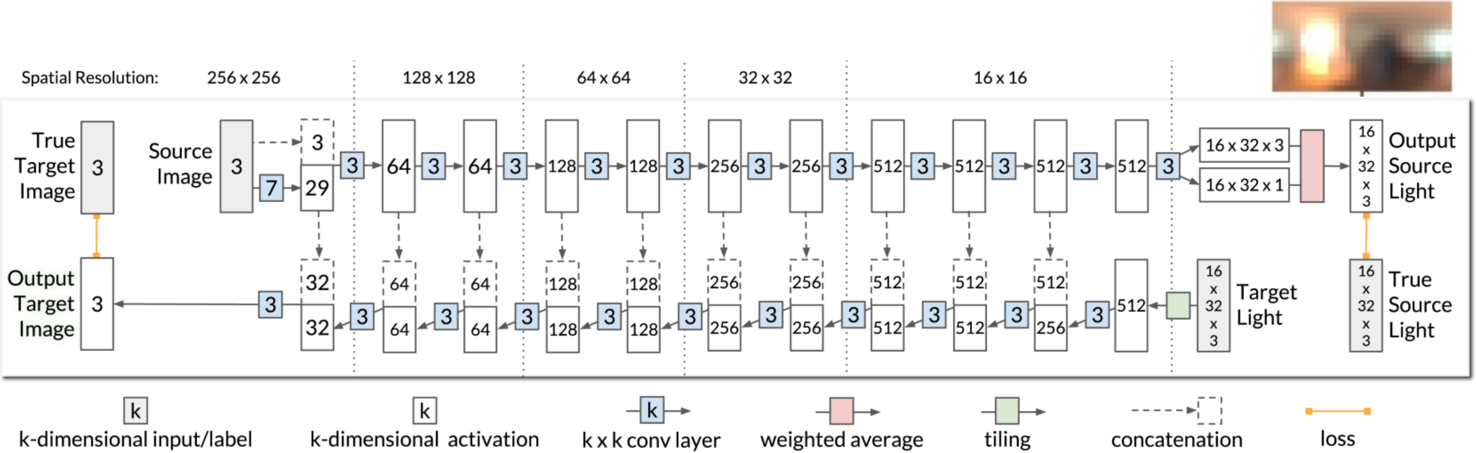
Image Relighting



U-Net

[Sun '19]

Image Relighting



[Sun '19]

Image Relighting

Light-stage dataset capture (Google)

[Sun ‘19]

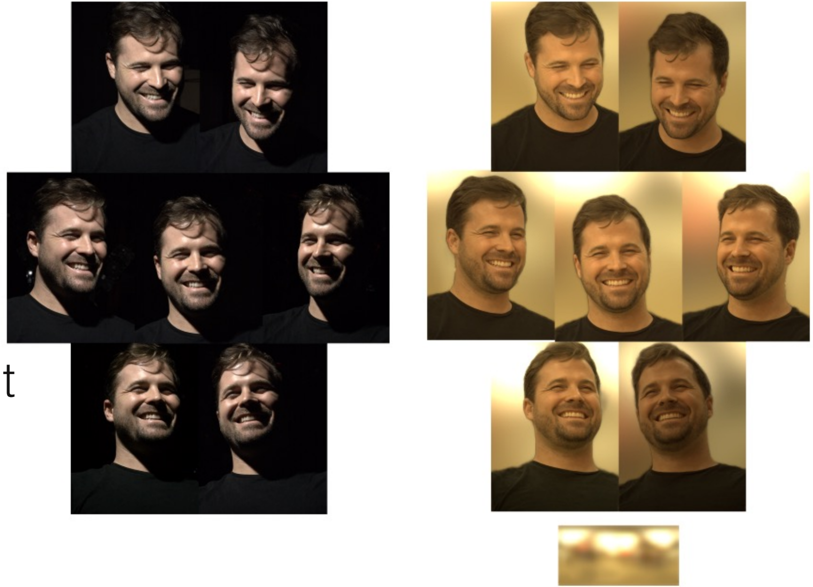
Image Relighting

OLAT photos (columns)

$$b = Ax$$

Re-rendered image

Environment map

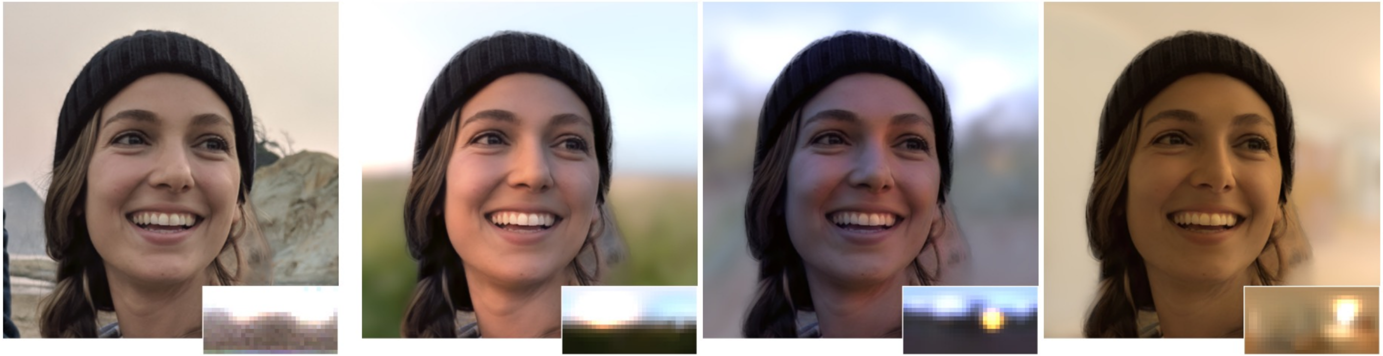


(a) OLAT images (7 cameras).

(b) Ground-truth renderings.

[Sun ‘19]

Image Relighting



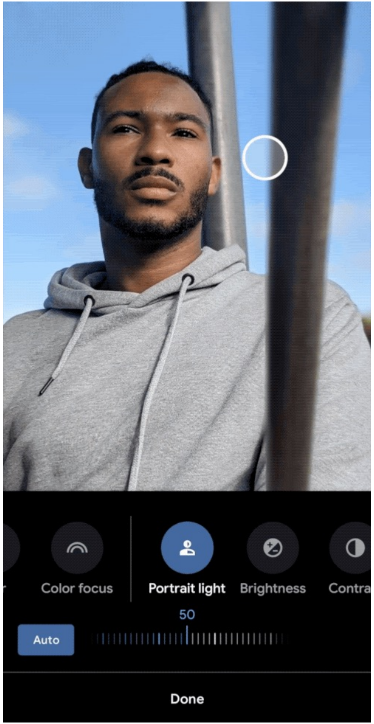
(a) Input image and estimated lighting

(b) Rendered images from our method under three novel illuminations

[Sun '19]

Image Relighting

Now a feature in pixel phones

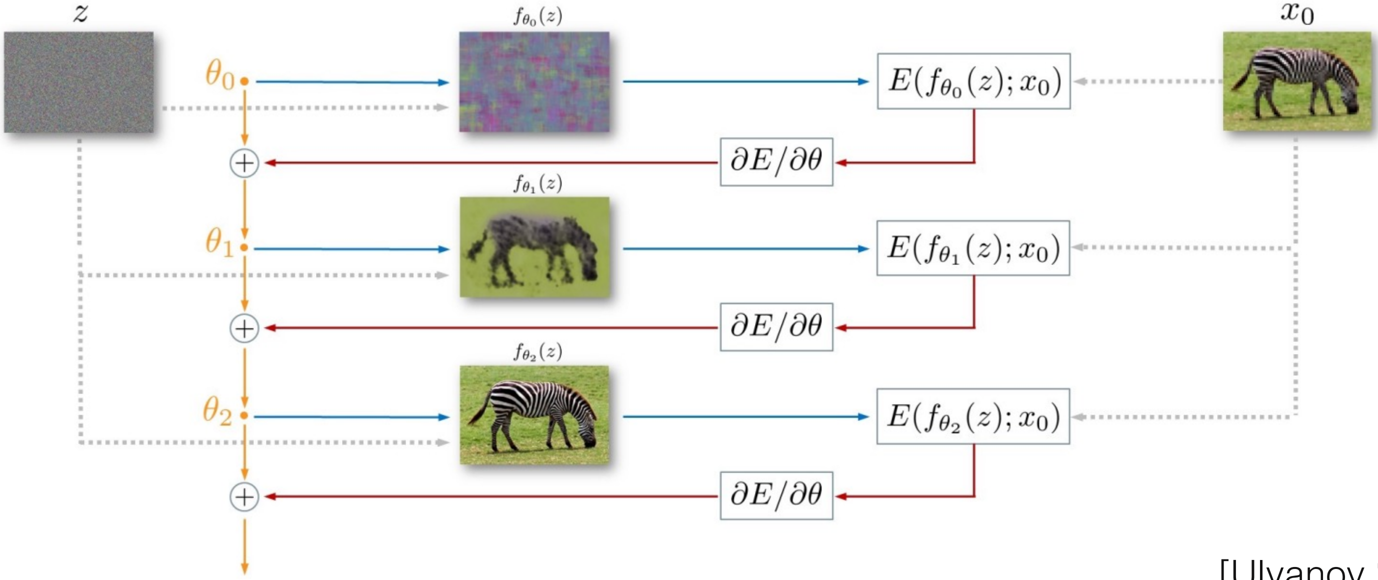


[Sun ‘19]

Do we always need training datasets?
(self-supervised learning)

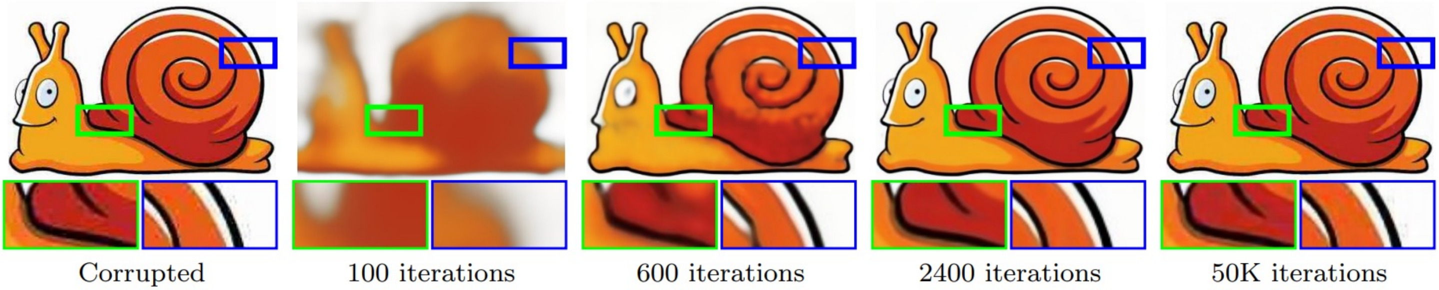
Deep image prior

Idea: Overfit a U-Net to a noisy image, but stop training early


z
$f_{\theta_0}(z)$
x_0
θ_0
$E(f_{\theta_0}(z); x_0)$
$\partial E/\partial\theta$
$f_{\theta_1}(z)$
θ_1
$E(f_{\theta_1}(z); x_0)$
$\partial E/\partial\theta$
$f_{\theta_2}(z)$
θ_2
$E(f_{\theta_2}(z); x_0)$
$\partial E/\partial\theta$

[Ulyanov ‘20]

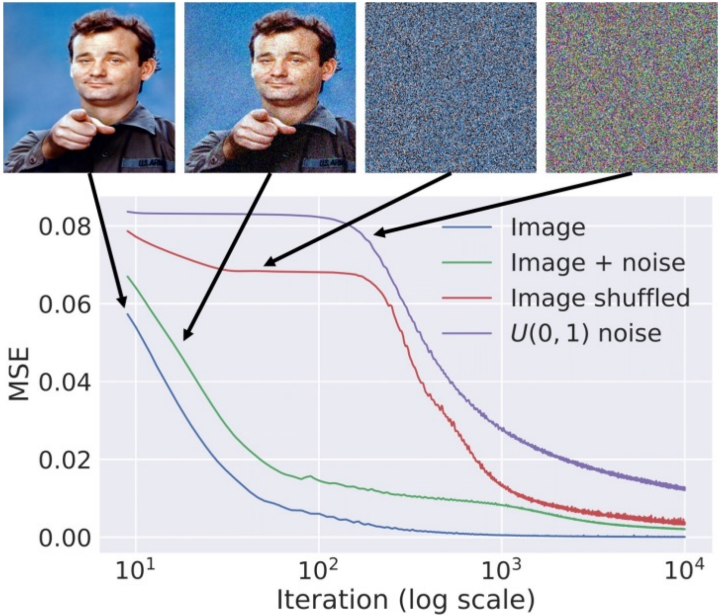
Deep image prior



The CNN itself is a good prior for natural images

[Ulyanov '20]

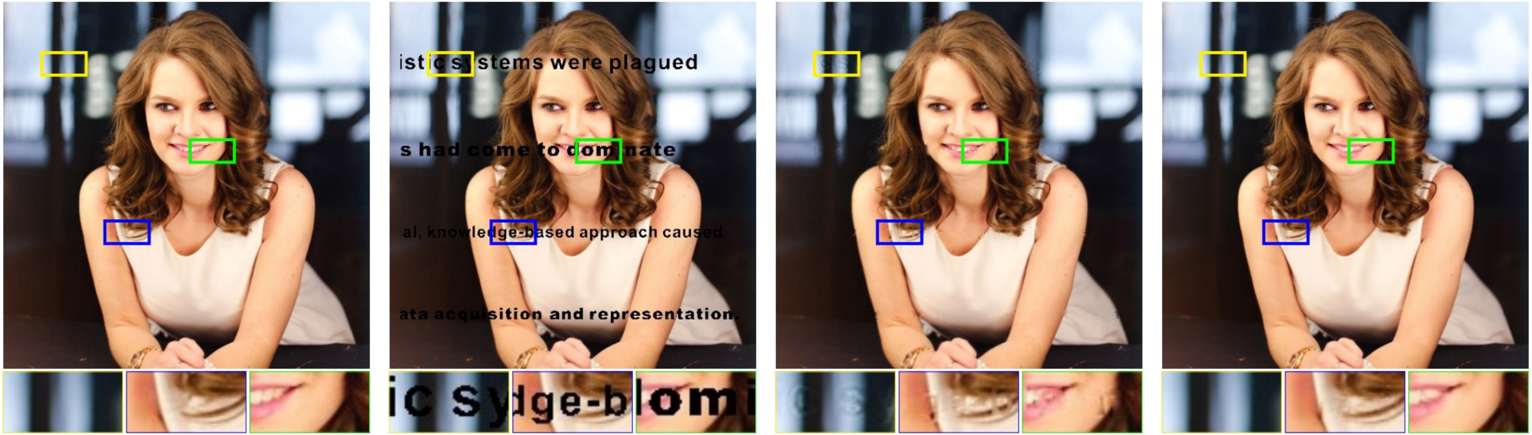
Deep image prior



During training, the network fits the image before noise

[Ulyanov '20]

Deep image prior


istic systems were plagued
s had come to dominate
al, knowledge-based approach caused
ata acquisition and representation.
ic Sydge-blomi

GT Corrupted Trained CNN DIP

[Ulyanov '20]

Summary

- “Neural” Networks & CNNs
- Building blocks of CNNs and deep networks
- Applications & inverse problems
- Just scratches the surface!
  - GANs, diffusion models, neural fields, neural rendering, text-to-image models, autoregressive models, transformers…

Next Time

- Optimization using alternating direction method of multipliers
- Hybrid techniques!

References and Further Reading

slides adapted from Stanford CS231N: http://cs231n.stanford.edu/slides/

CS229/CS231n notes on linear classifiers

https://cs231n.github.io/linear-classify/

https://cs229.stanford.edu/notes2021fall/cs229-notes1.pdf

CS231n Notes on backprop

http://cs231n.stanford.edu/handouts/linear-backprop.pdf

https://cs231n.github.io/optimization-2/

Intro to pytorch autograd

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Extending pytorch autograd functions

https://pytorch.org/docs/stable/notes/extending.html

References and Further Reading

slides adapted from Stanford CS231N: http://cs231n.stanford.edu/slides/

Araujo, André, Wade Norris, and Jack Sim. "Computing receptive fields of convolutional neural networks." *Distill* 4.11 (2019)

Chen, Chen, et al. "Learning to see in the dark." *Proc. CVPR*. 2018.

Eigen, David, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network." *Proc. NeurIPS.* (2014).

Farabet, Clement, et al. "Learning hierarchical features for scene labeling." *IEEE TPAMI.* 35.8 (2012): 1915-1929.

He, Kaiming, et al. "Deep residual learning for image recognition.“ Proc. CVPR. 2016.

Hubel, David H., and Torsten N. Wiesel. "Receptive fields of single neurones in the cat's striate cortex." *The Journal of physiology* 148.3 (1959): 574-591.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Proc. NeurIPS* 25 (2012): 1097-1105.

Lim, Bee, et al. "Enhanced deep residual networks for single image super-resolution." *Proc. CVPR Workshops*. 2017.

Lin, Haoning, Zhenwei Shi, and Zhengxia Zou. "Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network." *Remote sens.* 9.5 (2017): 480.

Metzler, Christopher A., et al. "Deep optics for single-shot high-dynamic-range imaging.“ Proc. CVPR. 2020.

Nah, Seungjun, Tae Hyun Kim, and Kyoung Mu Lee. "Deep multi-scale convolutional neural network for dynamic scene deblurring." *Proc. CVPR*. 2017.

Odena, Augustus, Vincent Dumoulin, and Chris Olah. "Deconvolution and checkerboard artifacts." *Distill* 1.10 (2016)

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert. "Feature visualization." *Distill* 2.11 (2017)

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. 2015.

Ren, Shaoqing, et al. "Faster R-CNN: towards real-time object detection with region proposal networks.“ *IEEE TPAMI.* 39.6 (2016): 1137-1149.

Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge.“ *IJCV* 115.3 (2015): 211-252.

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *Proc. ICLR* (2014).

Sun, Tiancheng, et al. "Single image portrait relighting." *ACM Trans. Graph.* 38.4 (2019): 79-1.

Toshev, Alexander, and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks." *Proc. CVPR*. 2014.

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." *Proc. CVPR*. 2018.

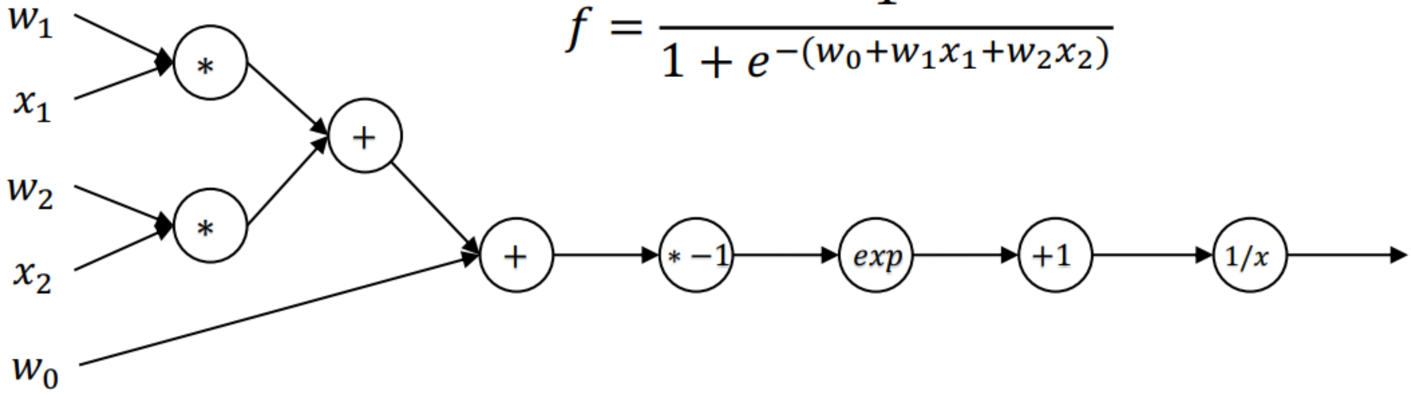
Zhang, Kai, et al. "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising." *IEEE Trans. Imag. Proc.* 26.7 (2017): 3142-3155.

Extra backpropagation example (adapted from Stanford CS231n)

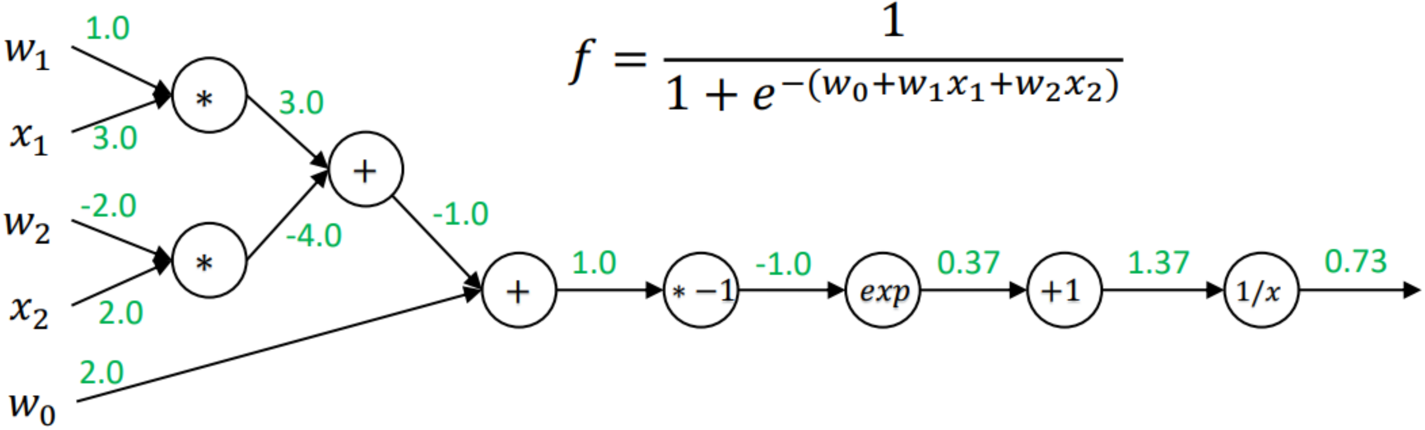
$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

Extra backpropagation example

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

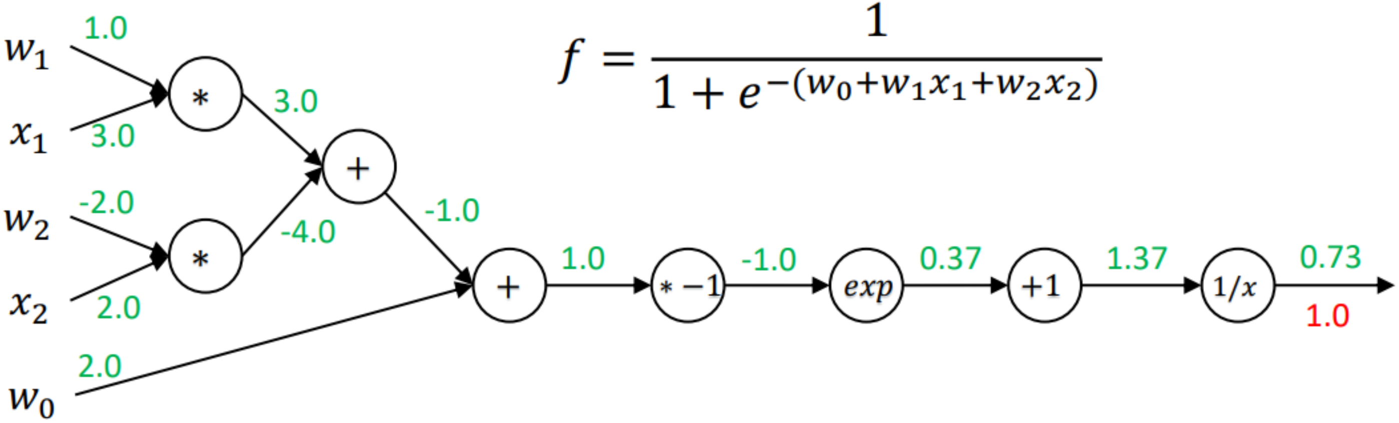


Extra backpropagation example



$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

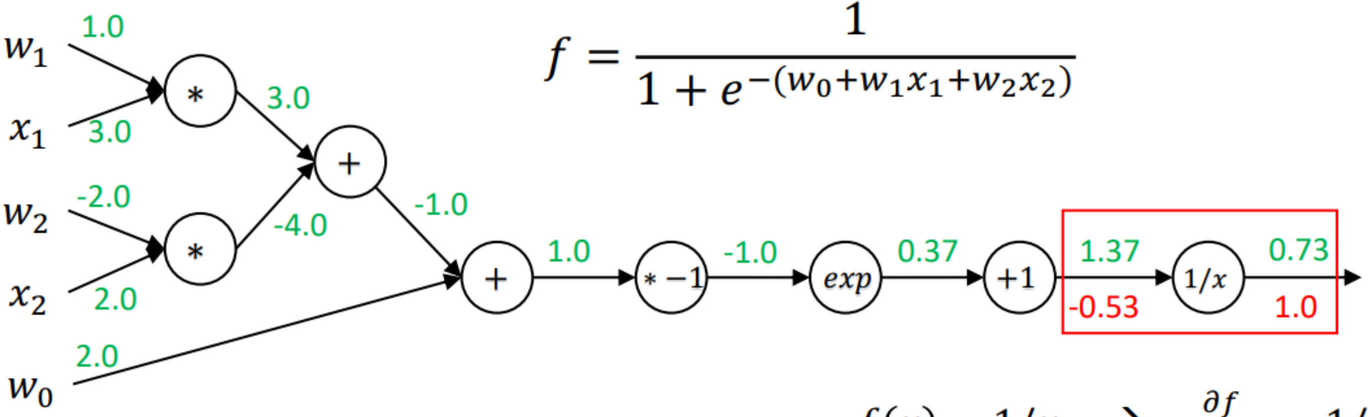
Extra backpropagation example



$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

Extra backpropagation example

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

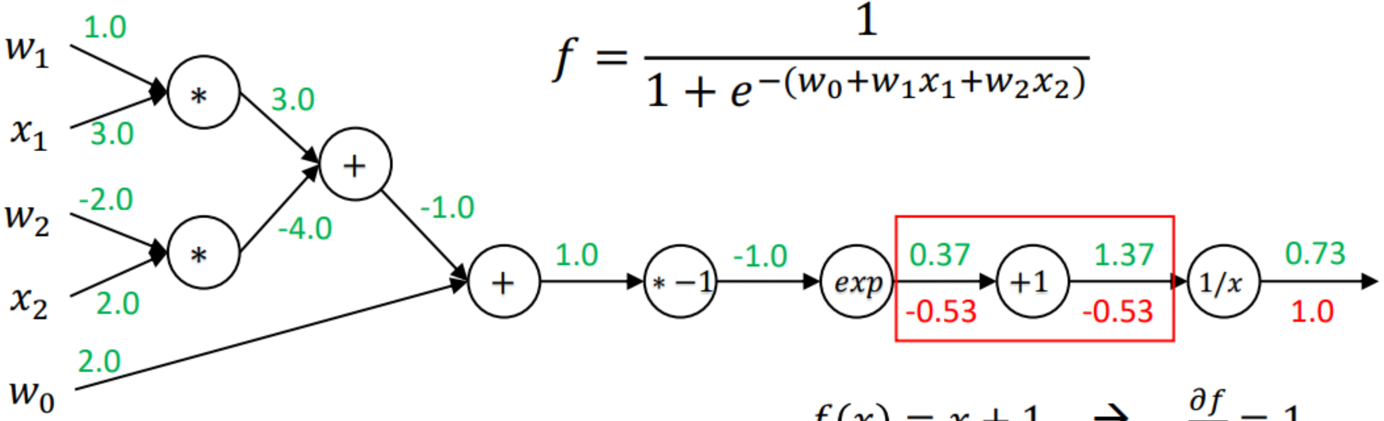


$$f(x) = 1/x \quad \rightarrow \quad \frac{\partial f}{\partial x} = -1/x^2$$

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial f}\frac{\partial f}{\partial x} = -1/x^2$$

Extra backpropagation example

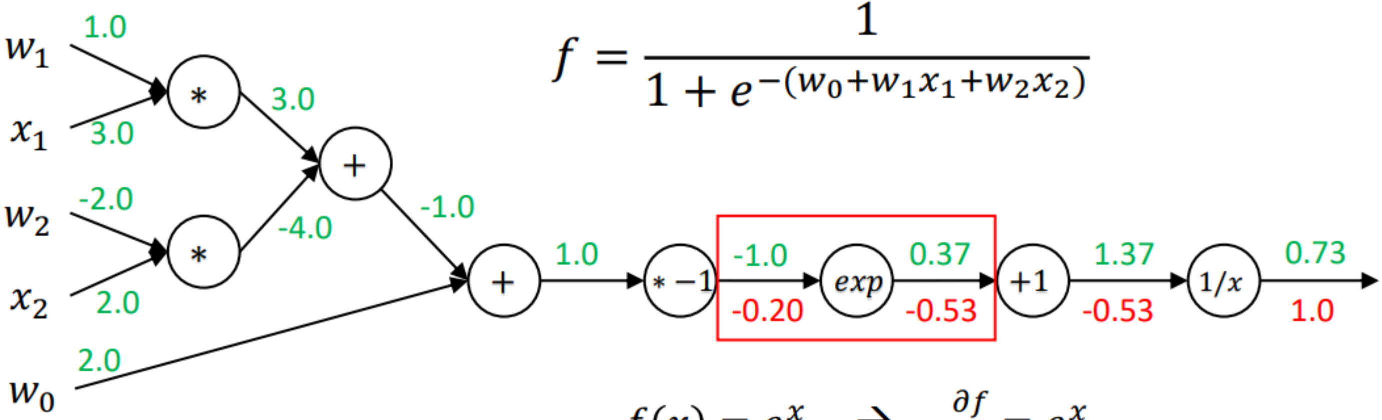
$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



$$f(x) = x + 1 \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1$$

Extra backpropagation example

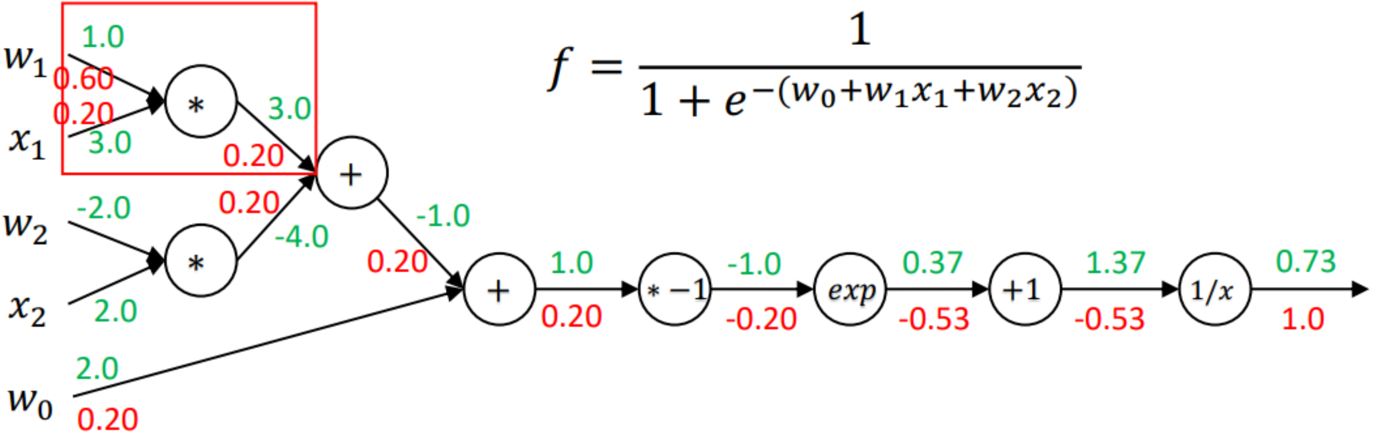
$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



$$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$$

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial f}\frac{\partial f}{\partial x} = \frac{\partial J}{\partial f} \cdot e^x$$

Extra backpropagation example

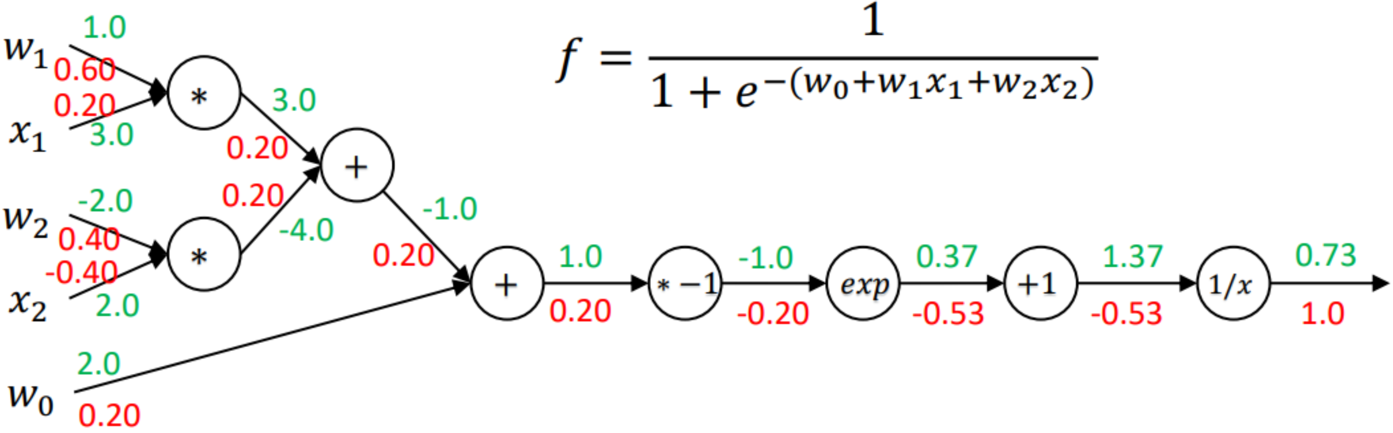


$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

$$f(x, w) = xw \quad \rightarrow \quad \frac{\partial f}{\partial x} = w, \ \frac{\partial f}{\partial w} = x$$

Extra backpropagation example

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



w_1
1.0
0.60
x_1
3.0
0.20
*
3.0
0.20
w_2
-2.0
0.40
x_2
2.0
-0.40
*
-4.0
0.20
+
-1.0
0.20
w_0
2.0
0.20
+
1.0
0.20
$* -1$
-1.0
-0.20
exp
0.37
-0.53
$+1$
1.37
-0.53
$1/x$
0.73
1.0