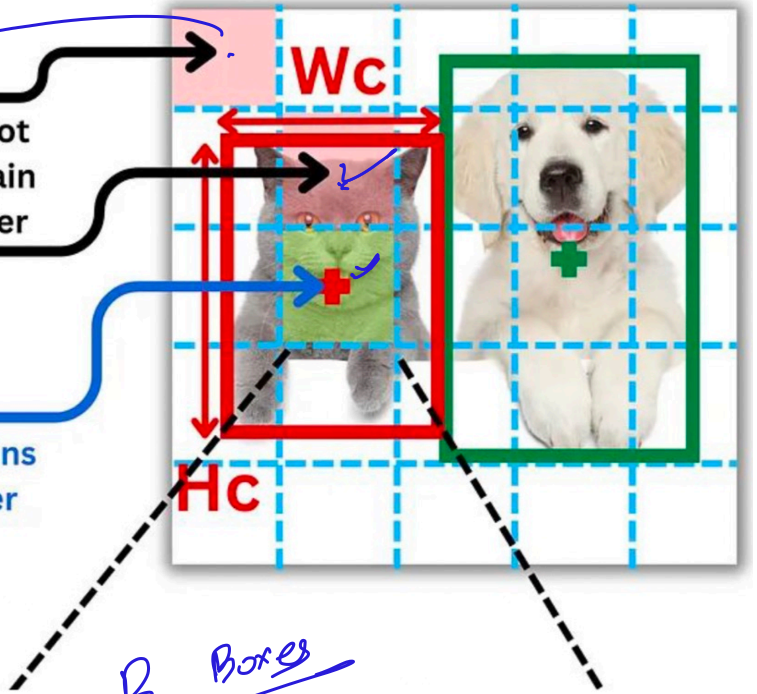
Objects can be larger than the bounding boxes

5 (B=1). $\boxed{C=2}$ Don't care

Center X	Center Y	Width	Height	Contains object?	Cat	Dog	
-	-	-	-	0	0	0	Do not contain center
-	-	-	-	0	0	0	Do not contain center
Xc	Yc	Wc	Hc	1	1	0	Contains center

$\underbrace{\text{Center X, Center Y, Width, Height}}_{\text{Bounding box position}}$ $\underbrace{\text{Contains object?}}_{\text{Objectness}}$ $\underbrace{\text{Cat, Dog}}_{\text{Class labels}}$



B Boxes

	Type	Filters	Size/Stride	Output
	Conv	64	$7 \times 7 / 2$	224×224
	Max Pool		$2 \times 2 / 2$	112×112
	Conv	192	$3 \times 3 / 1$	112×112
	Max Pool		$2 \times 2 / 2$	56×56
$1\times$	Conv	128	$1 \times 1 / 1$	56×56
	Conv	256	$3 \times 3 / 1$	56×56
	Conv	256	$1 \times 1 / 1$	56×56
	Conv	512	$3 \times 3 / 1$	56×56
	Max Pool		$2 \times 2 / 2$	28×28
$4\times$	Conv	256	$1 \times 1 / 1$	28×28
	Conv	512	$3 \times 3 / 1$	28×28
	Conv	512	$1 \times 1 / 1$	28×28
	Conv	1024	$3 \times 3 / 1$	28×28
	Max Pool		$2 \times 2 / 2$	14×14
$2\times$	Conv	512	$1 \times 1 / 1$	14×14
	Conv	1024	$3 \times 3 / 1$	14×14
	Conv	1024	$3 \times 3 / 1$	14×14
	Conv	1024	$3 \times 3 / 2$	7×7
	Conv	1024	$3 \times 3 / 1$	7×7
	Conv	1024	$3 \times 3 / 1$	7×7
	FC		4096	4096
	Dropout 0.5			4096
	FC		$\boxed{7 \times 7 \times 30}$	$7 \times 7 \times 30$

Training: YOLOv1

- First 20 layers pretrained using ImageNet
- Last 4 layers initialized randomly and trained with the task-specific dataset
- Various data augmentation methods
- Loss function for localization as well as confidence and classification

$7 \times 2 \times [2 \times 5 + 20]$

How does the training work?

- **Minimizing the error for the box center positions:**

$$L_{\text{position}} = \sum_{i \in \text{grid}} \sum_{j \in \text{boxes}} \mathbb{I}_{\{\text{if object in } i\}} \left[\left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2 \right]$$

We simply minimize the MSE for ***x*** and ***y*** for each box from the ground truth values when an object exists in the cell. Here $\boldsymbol{I}_{\{condition\}}$ is the indicator function ($\boldsymbol{I}_{\{condition\}}$ = *1* if condition and *0* otherwise).

How does the training work?

- **Minimizing the error for the box shape:**

$$L_{\text{shape}} = \sum_{i \in \text{grid}} \sum_{j \in \text{boxes}} \mathbb{I}_{\{\text{if object in } i\}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

We minimize the square-root values of ***w*** and ***h*** for each box from the ground truth when an object exists in the cell. This is done to give more balance to smaller boxes during training.

How does the training work?

large ≈ 1 ≈ 0 small

- **Minimizing the error for the objectness:**

$$L_{\text{objectness}} = \sum_{i \in \text{grid}} \sum_{j \in \text{boxes}} \mathbb{I}_{\{\text{if object in } i\}} a \left(1 - \hat{C}_i\right)^2 + \mathbb{I}_{\{\text{if object not in } i\}} b \hat{C}_i^2$$

The weights ***a*** and ***b*** are different (***a = 5*** and ***b = 0.5***) depending if there is an object or not in the cell because there are way more cells without objects than there are. This avoids biasing the model into overly predicting small values.

How does the training work?

- **Minimizing the error for the predicted classes:**

$$L_{\text{classes}} = \sum_{i \in \text{grid}} \mathbb{I}_{\{\text{if object in } i\}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

We minimize the MSE for ***p(c)*** for each class from the ground truth values when an object exists in the cell.

The overall loss function is simply the weighted sum of all the above losses:

$$\boxed{L = a\left(L_{\text{position}} + L_{\text{shape}}\right) + L_{\text{objectness}} + L_{\text{classes}}}$$

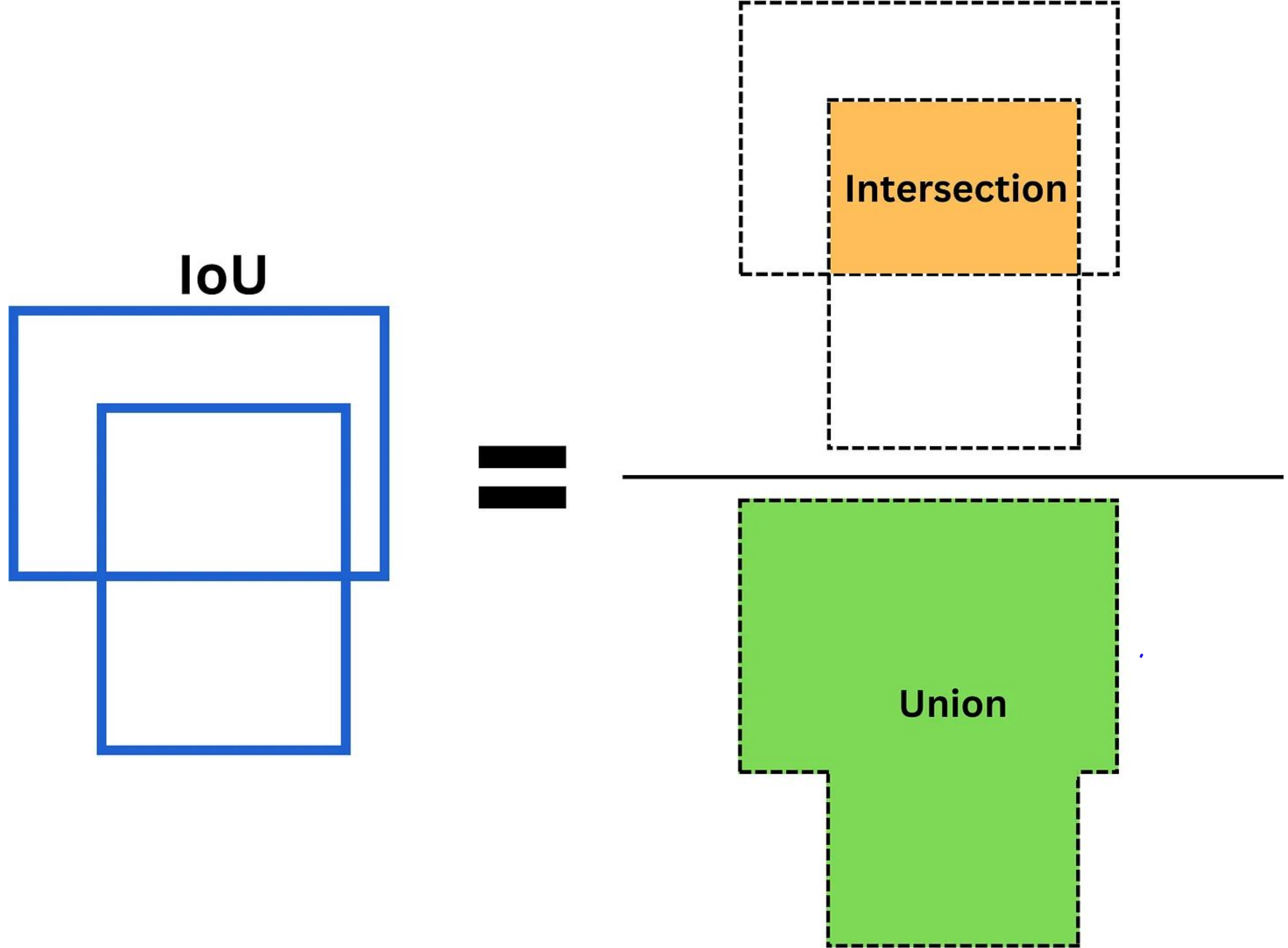
Post-processing during prediction

: Non-Maximum Suppression (NMS). a) Shows the typical output of an object detection model containing overlapping boxes. b) Shows the output after NMS.

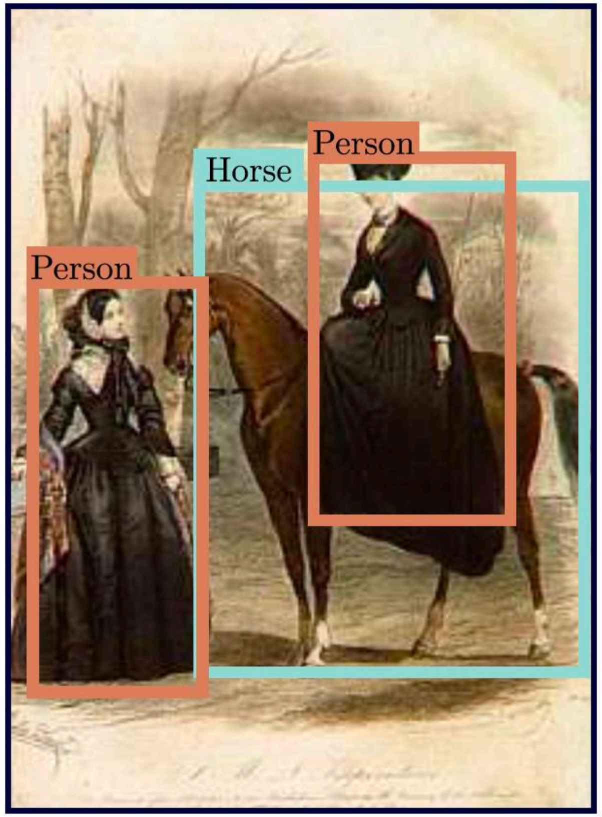
Algorithm 1 Non-Maximum Suppression Algorithm

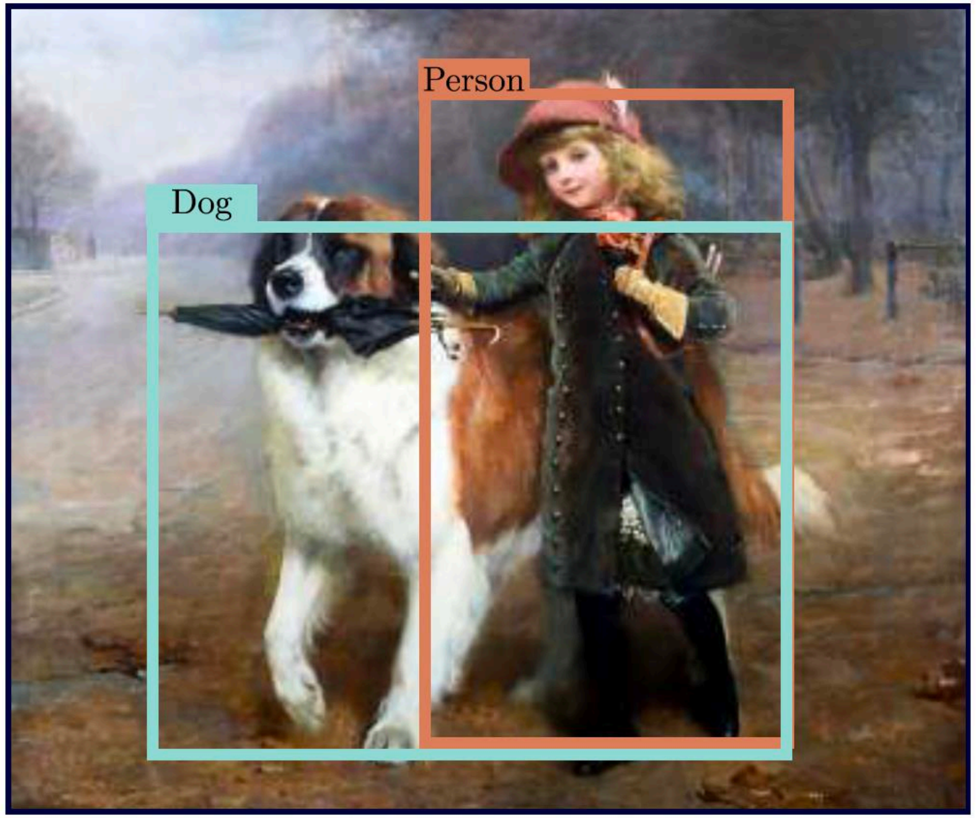
Require: Set of predicted bounding boxes B, confidence scores S, IoU threshold τ, confidence threshold T
Ensure: Set of filtered bounding boxes F

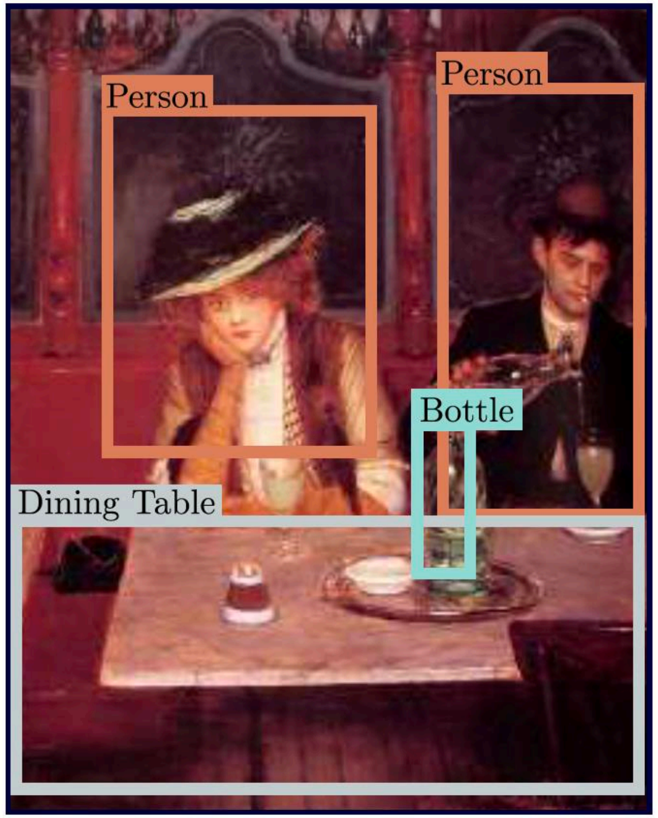
$F \leftarrow \emptyset$
Filter the boxes: $B \leftarrow \{b \in B \mid S(b) \geq T\}$
Sort the boxes B by their confidence scores in descending order
while $B \neq \emptyset$ **do**
    Select the box b with the highest confidence score
    Add b to the set of final boxes F: $F \leftarrow F \cup \{b\}$
    Remove b from the set of boxes B: $B \leftarrow B - \{b\}$
    for all remaining boxes r in B **do**
        Calculate the IoU between b and r: $iou \leftarrow IoU(b, r)$
        if $iou \geq \tau$ **then**
            Remove r from the set of boxes B: $B \leftarrow B - \{r\}$
        end if
    end for
end while


IoU
=
Intersection
Union

Results


Person
Horse
Person

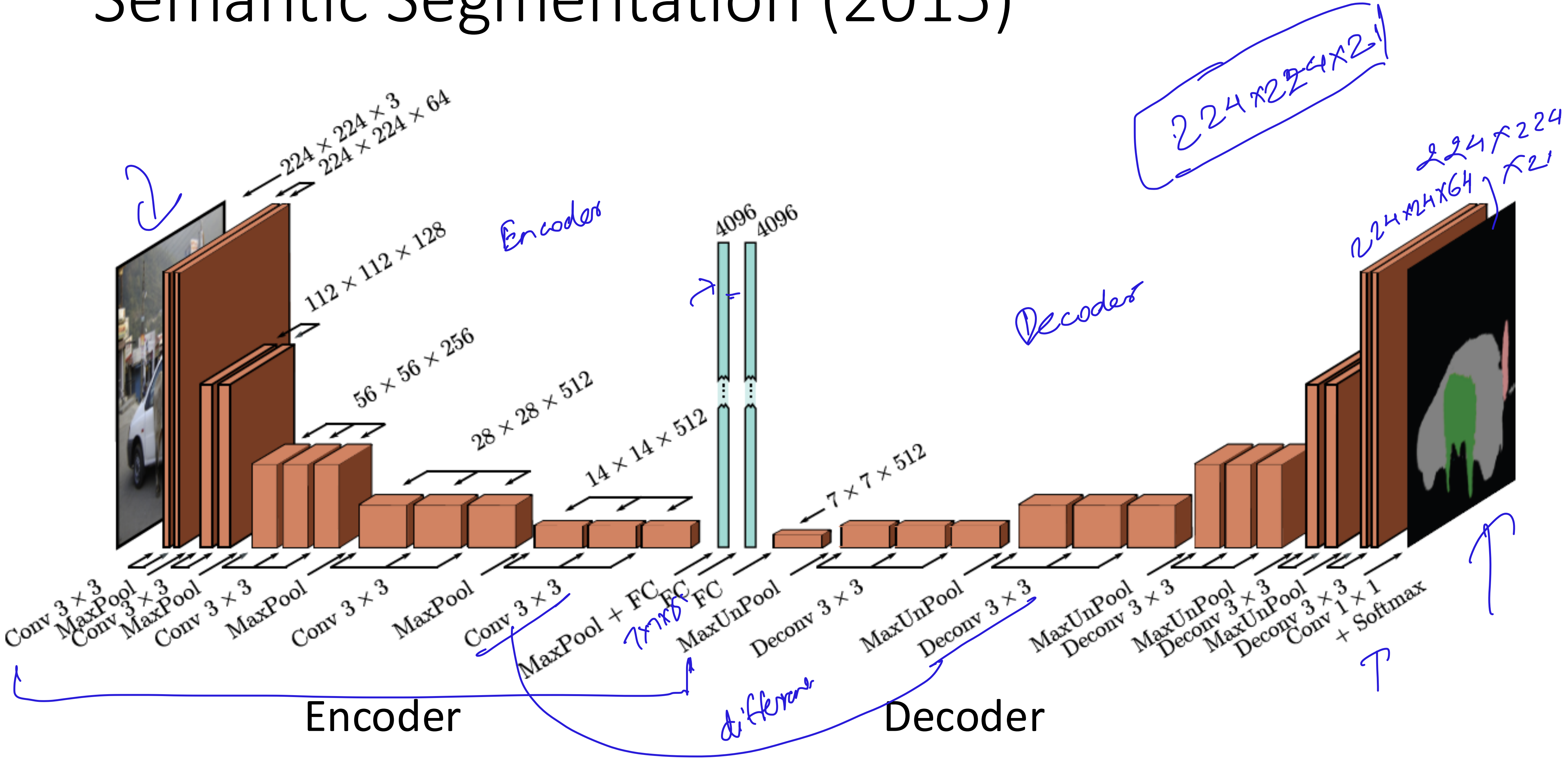

Person
Dog


Person
Person
Bottle
Dining Table

Convolution #2

- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

Semantic Segmentation (2015)



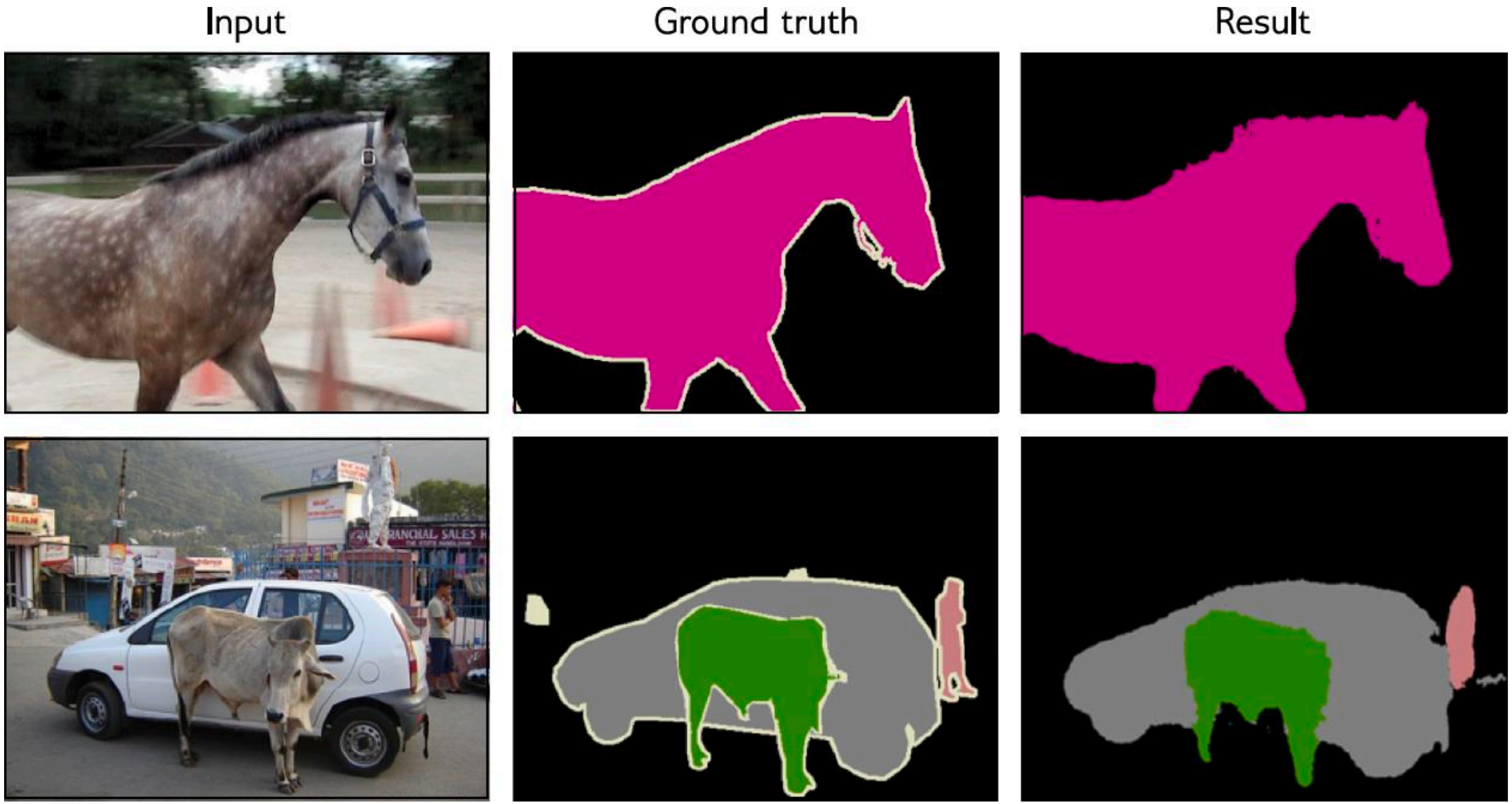
name	kernel size	stride	pad	output size
input	-	-	-	$224 \times 224 \times 3$
conv1-1	3×3	1	1	$224 \times 224 \times 64$
conv1-2	3×3	1	1	$224 \times 224 \times 64$
pool1	2×2	2	0	$112 \times 112 \times 64$
conv2-1	3×3	1	1	$112 \times 112 \times 128$
conv2-2	3×3	1	1	$112 \times 112 \times 128$
pool2	2×2	2	0	$56 \times 56 \times 128$
conv3-1	3×3	1	1	$56 \times 56 \times 256$
conv3-2	3×3	1	1	$56 \times 56 \times 256$
conv3-3	3×3	1	1	$56 \times 56 \times 256$
pool3	2×2	2	0	$28 \times 28 \times 256$
conv4-1	3×3	1	1	$28 \times 28 \times 512$
conv4-2	3×3	1	1	$28 \times 28 \times 512$
conv4-3	3×3	1	1	$28 \times 28 \times 512$
pool4	2×2	2	0	$14 \times 14 \times 512$
conv5-1	3×3	1	1	$14 \times 14 \times 512$
conv5-2	3×3	1	1	$14 \times 14 \times 512$
conv5-3	3×3	1	1	$14 \times 14 \times 512$
pool5	2×2	2	0	$7 \times 7 \times 512$
fc6	7×7	1	0	$1 \times 1 \times 4096$
fc7	1×1	1	0	$1 \times 1 \times 4096$

Encoder

decoders

deconv-fc6	7×7	1	0	$7 \times 7 \times 512$
unpool5	2×2	2	0	$14 \times 14 \times 512$
deconv5-1	3×3	1	1	$14 \times 14 \times 512$
deconv5-2	3×3	1	1	$14 \times 14 \times 512$
deconv5-3	3×3	1	1	$14 \times 14 \times 512$
unpool4	2×2	2	0	$28 \times 28 \times 512$
deconv4-1	3×3	1	1	$28 \times 28 \times 512$
deconv4-2	3×3	1	1	$28 \times 28 \times 512$
deconv4-3	3×3	1	1	$28 \times 28 \times 256$
unpool3	2×2	2	0	$56 \times 56 \times 256$
deconv3-1	3×3	1	1	$56 \times 56 \times 256$
deconv3-2	3×3	1	1	$56 \times 56 \times 256$
deconv3-3	3×3	1	1	$56 \times 56 \times 128$
unpool2	2×2	2	0	$112 \times 112 \times 128$
deconv2-1	3×3	1	1	$112 \times 112 \times 128$
deconv2-2	3×3	1	1	$112 \times 112 \times 64$
unpool1	2×2	2	0	$224 \times 224 \times 64$
deconv1-1	3×3	1	1	$224 \times 224 \times 64$
deconv1-2	3×3	1	1	$224 \times 224 \times 64$
output	1×1	1	1	$\boxed{224 \times 224 \times 21}$

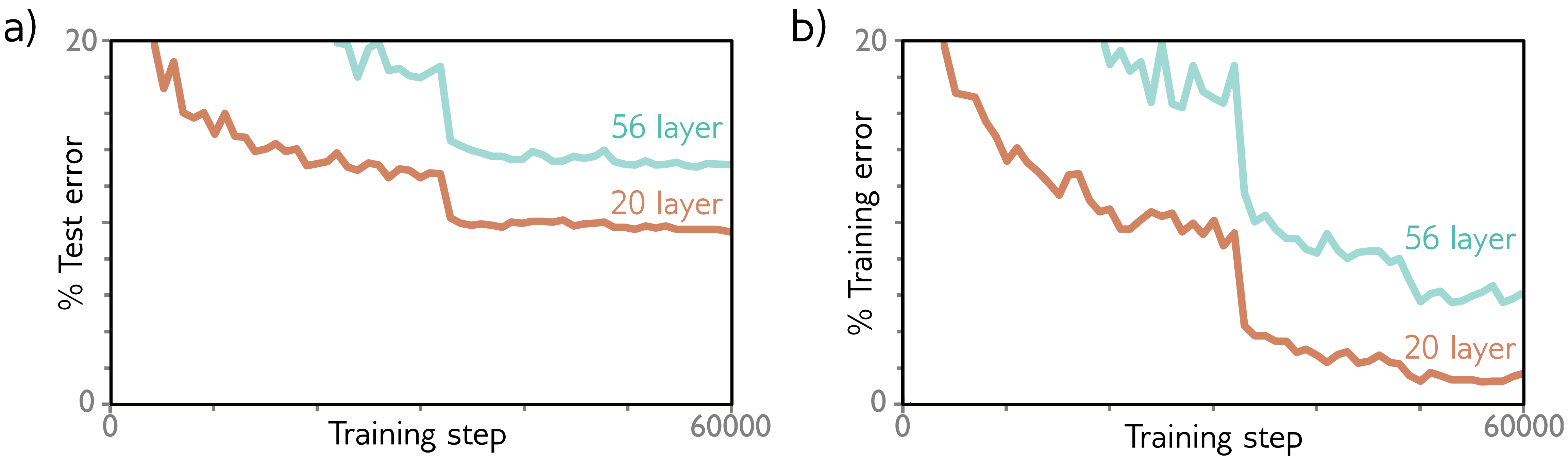
Semantic segmentation results



Convolution #2

- 2D Convolution
- Downsampling and upsampling, 1x1 convolution
- Image classification
- Object detection
- Semantic segmentation
- Residual networks
- U-Nets and hourglass networks

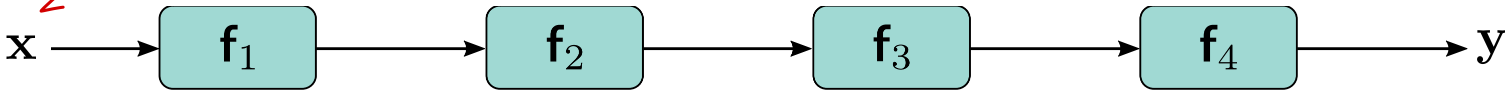
CIFAR Image classification for deeper networks



Regular network:

$$\mathbf{h}_1 = \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1]$$
$$\mathbf{h}_2 = \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2]$$
$$\mathbf{h}_3 = \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3]$$
$$\mathbf{y} = \mathbf{f}_4[\mathbf{h}_3, \boldsymbol{\phi}_4]$$

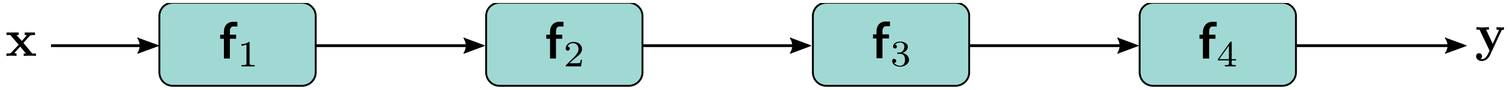
Shattered Gradients



$$\frac{\partial y}{\partial x} = \frac{\partial f_1}{\partial x}\,\frac{\partial f_2}{\partial f_1}\,\frac{\partial f_3}{\partial f_2}\,\frac{\partial f_4}{\partial f_3}$$

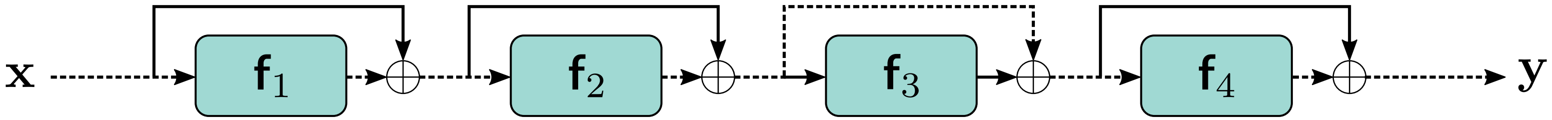
Regular network:

$$
\begin{aligned}
\mathbf{h}_1 &= \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1] \\
\mathbf{h}_2 &= \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2] \\
\mathbf{h}_3 &= \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3] \\
\mathbf{y} &= \mathbf{f}_4[\mathbf{h}_3, \boldsymbol{\phi}_4]
\end{aligned}
$$



Residual network (2016):

$$
\begin{aligned}
\mathbf{h}_1 &= \mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1] \\
\mathbf{h}_2 &= \mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2] \\
\mathbf{h}_3 &= \mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3] \\
\mathbf{y} &= \mathbf{h}_3 + \mathbf{f}_4[\mathbf{h}_3, \boldsymbol{\phi}_4]
\end{aligned}
$$



$f_2(x + f_1(x)) + f_1(x)$

Batch Normalization

Consider a single layer $y = Wx$

The following could lead to tough optimization:
- Inputs x are not *centered around zero* (need large bias)
- Inputs x have different scaling per-element (entries in W will need to vary a lot)

Idea: force inputs to be "nicely scaled" at each layer!

Batch Normalization

[Ioffe and Szegedy, 2015]

"you want zero-mean unit-variance activations? just make them so."

consider a batch of activations at some layer. To make each dimension zero-mean unit-variance, apply:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

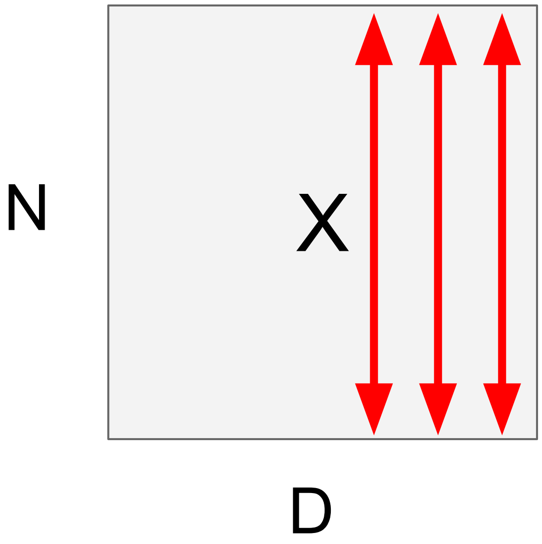
this is a vanilla differentiable function...

Batch Normalization

[Ioffe and Szegedy, 2015]

samples

Input: $x : N \times D$ → size of hidden layers



$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

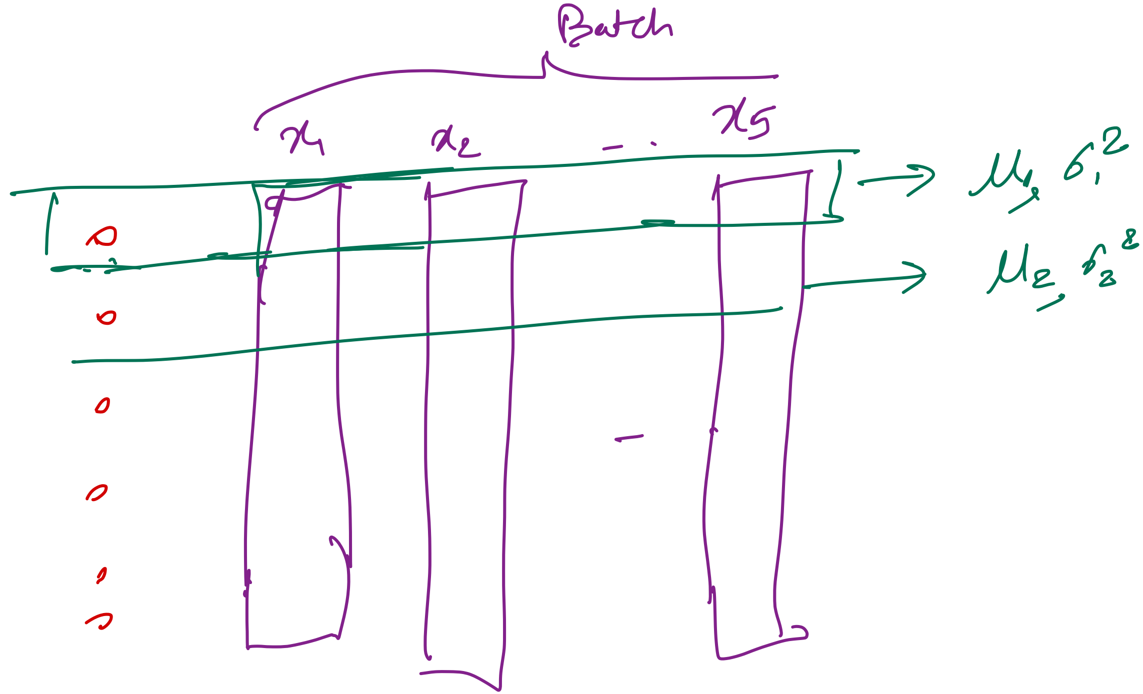
Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} (x_{i,j} - \mu_j)^2$$

Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D



6 hidden units

Batch Normalization for ConvNets

Batch Normalization for **fully-connected** networks

$\mathbf{x}$: N × D

Normalize

$\boldsymbol{\mu}, \boldsymbol{\sigma}$: 1 × D

$\boldsymbol{\gamma}, \boldsymbol{\beta}$: 1 × D

$\mathbf{y} = \boldsymbol{\gamma}(\mathbf{x}-\boldsymbol{\mu})/\boldsymbol{\sigma}+\boldsymbol{\beta}$

Batch Normalization for **convolutional** networks (Spatial Batchnorm, BatchNorm2D)

$\mathbf{x}$: N×C×H×W

Normalize

$\boxed{\boldsymbol{\mu}, \boldsymbol{\sigma}\text{: } 1\times C\times 1\times 1}$

$\boldsymbol{\gamma}, \boldsymbol{\beta}$: 1×C×1×1

$\mathbf{y} = \boldsymbol{\gamma}(\mathbf{x}-\boldsymbol{\mu})/\boldsymbol{\sigma}+\boldsymbol{\beta}$