# Lecture 9:
# Object Detection and Image Segmentation



## Recall: Image Classification: A core task in Computer Vision



(assume given a set of possible labels)
{dog, cat, truck, plane, ...}

→ cat



## Computer Vision Tasks

Classification | Semantic Segmentation | Object Detection | Instance Segmentation

CAT | GRASS, CAT, TREE, SKY | DOG, DOG, CAT | DOG, DOG, CAT

No spatial extent | No objects, just pixels | Multiple Object





## Semantic Segmentation

Classification | Semantic Segmentation | Object Detection | Instance Segmentation

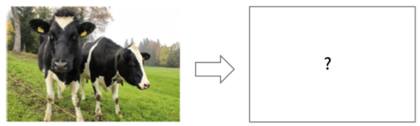
CAT | GRASS, CAT, TREE, SKY | DOG, DOG, CAT | DOG, DOG, CAT

No spatial extent | No objects, just pixels | Multiple Object

# Semantic Segmentation: The Problem

GRASS, CAT, TREE,
SKY, ...

Paired training data: for each training image, each pixel is labeled with a semantic category.




At test time, classify each pixel of a new image.



# Semantic Segmentation Idea: Sliding Window

Full image




# Semantic Segmentation Idea: Sliding Window

Full image

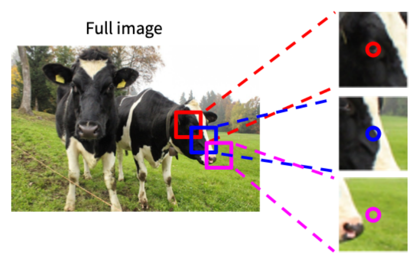

Impossible to classify without context

Q: how do we include context?



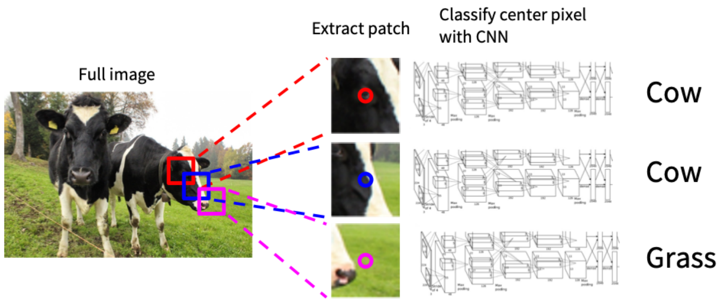
# Semantic Segmentation Idea: Sliding Window

Full image



Q: how do we model this?

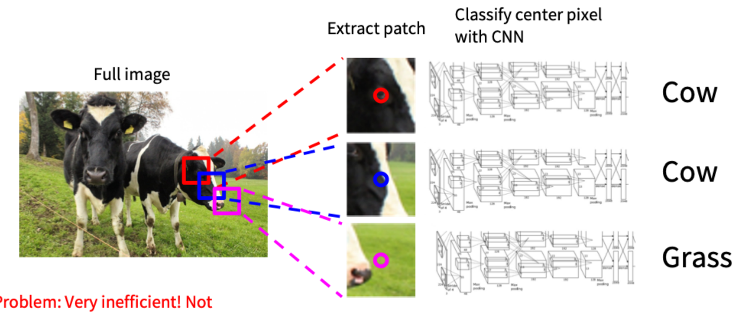
## Semantic Segmentation Idea: Sliding Window




Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014



## Semantic Segmentation Idea: Sliding Window

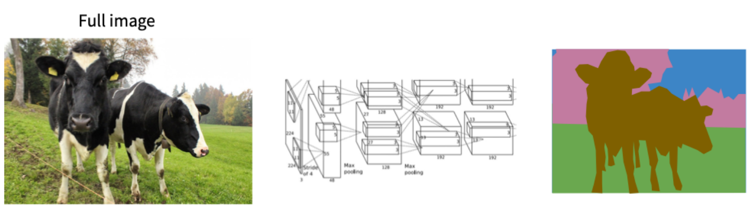



Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014



## Semantic Segmentation Idea: Convolution



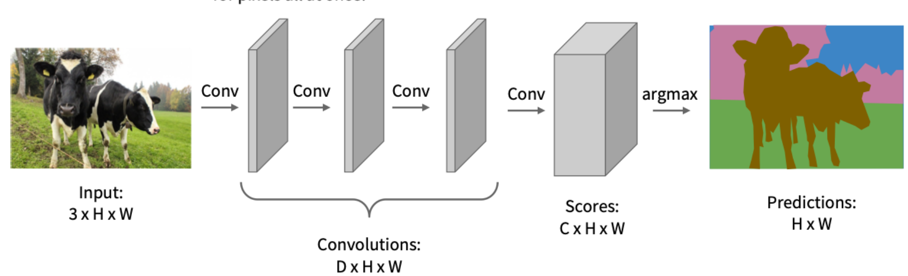

An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.



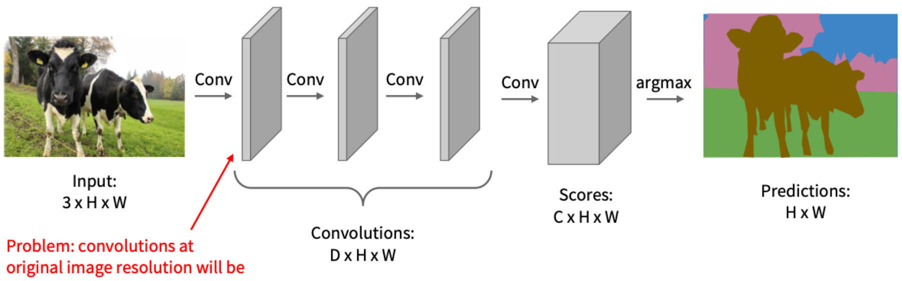
## Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



# Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!

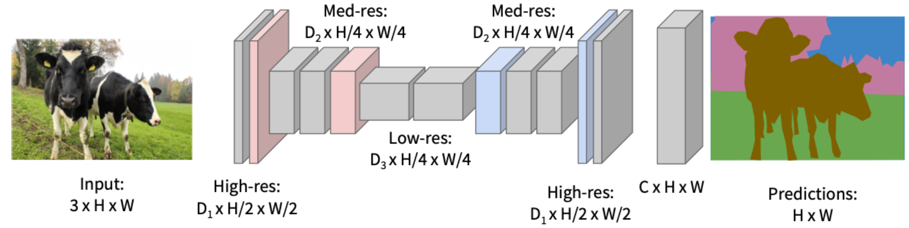



Problem: convolutions at original image resolution will be very expensive ...



# Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!




Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

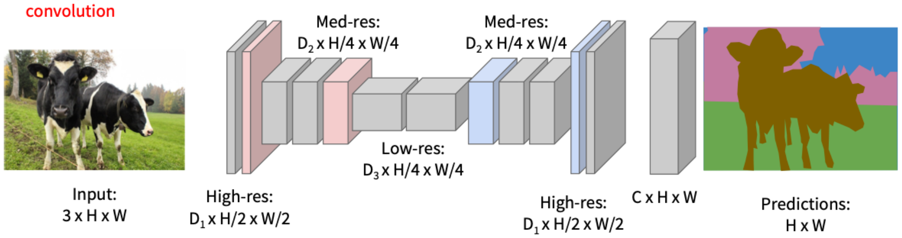


# Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!

Upsampling: ???




Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015



# In-Network upsampling: "Unpooling"

Nearest Neighbor

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

Input: 2 x 2

→

| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 2 |
| 1 | 1 | 2 | 2 |
| 3 | 3 | 4 | 4 |
| 3 | 3 | 4 | 4 |

Output: 4 x 4

"Bed of Nails"

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

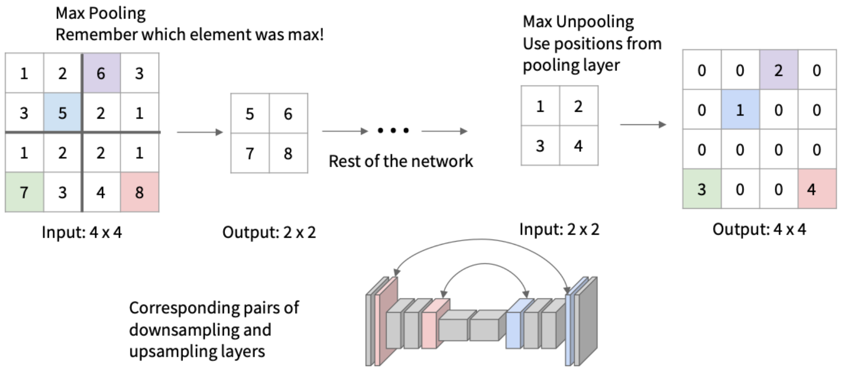
Input: 2 x 2

→

| | | | |
|---|---|---|---|
| 1 | 0 | 2 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 |

Output: 4 x 4

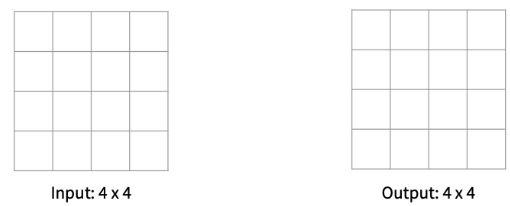
# In-Network upsampling: “Max Unpooling”






# Learnable Upsampling

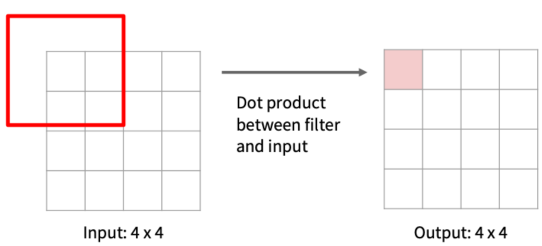
Recall: Normal 3 x 3 convolution, stride 1 pad 1






# Learnable Upsampling

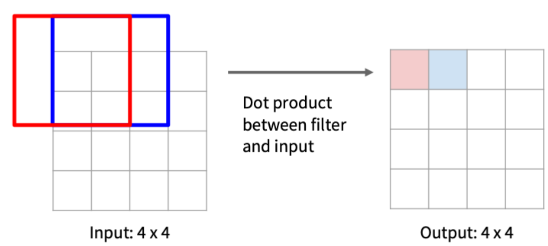
Recall: Normal 3 x 3 convolution, stride 1 pad 1





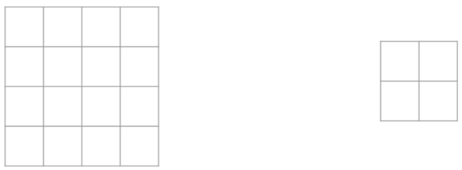

# Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 1 pad 1



# Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 2 pad 1



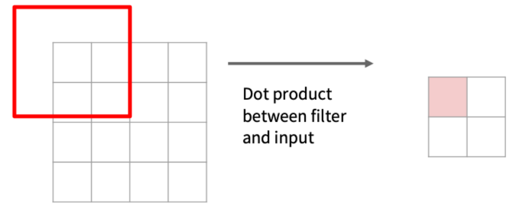
Input: 4 x 4

Output: 2 x 2



# Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 2 pad 1



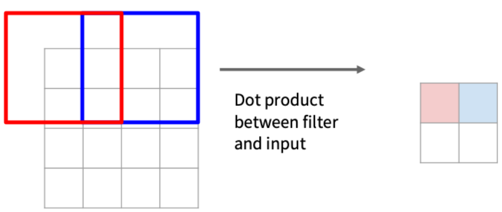

Input: 4 x 4

Output: 2 x 2



# Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 2 pad 1




Input: 4 x 4

Output: 2 x 2

Filter moves 2 pixels in the input for every one pixel in the output

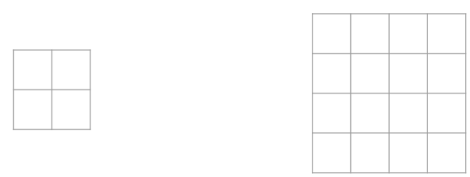
Stride gives ratio between movement in input and output

We can interpret strided convolution as "learnable downsampling".



# Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1

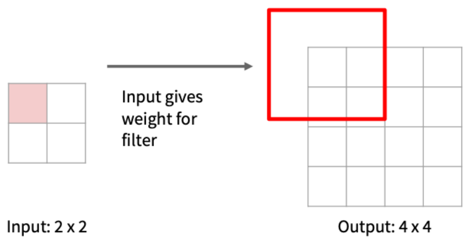


Input: 2 x 2

Output: 4 x 4

# Learnable Upsampling: Transposed Convolution

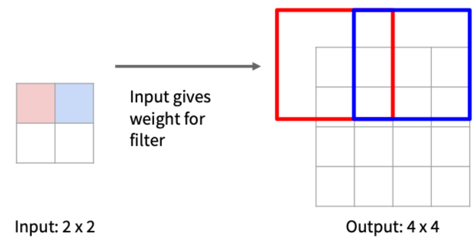
3 x 3 transposed convolution, stride 2 pad 1






# Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1



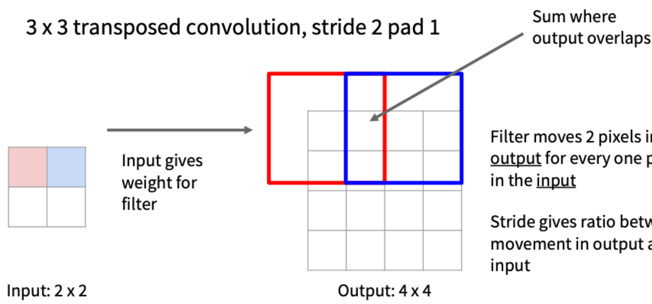

Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input



# Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1




Filter moves 2 pixels in the output for every one pixel in the input

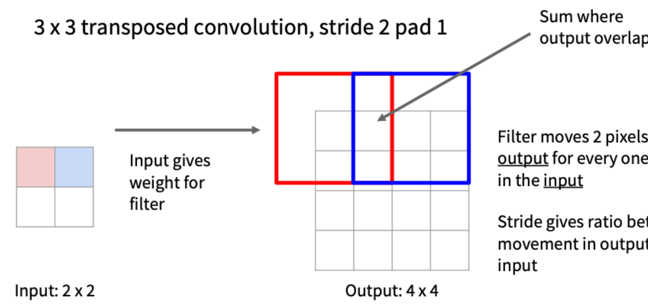
Stride gives ratio between movement in output and input



# Learnable Upsampling: Transposed Convolution

Q: Why is it called transposed convolution?

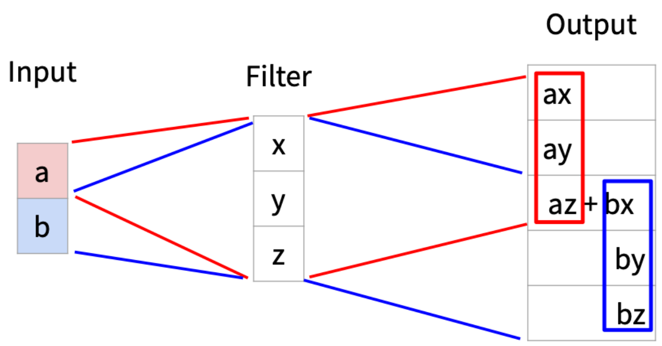
3 x 3 transposed convolution, stride 2 pad 1




Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

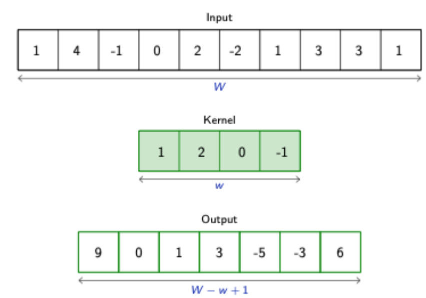
## Learnable Upsampling: 1D Example




Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

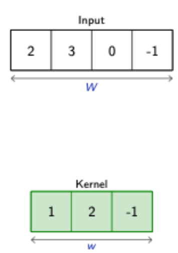


Convolution layer



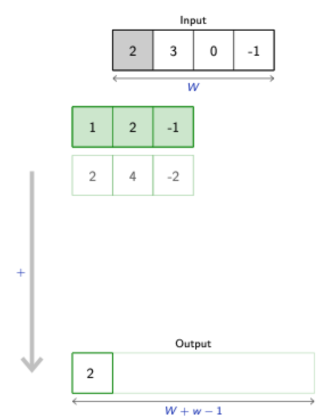



Transposed convolution layer

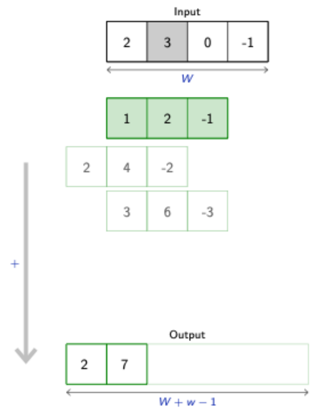





Transposed convolution layer

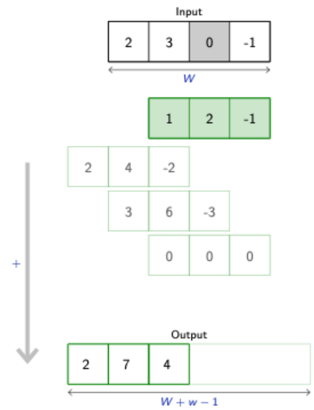


## Transposed convolution layer



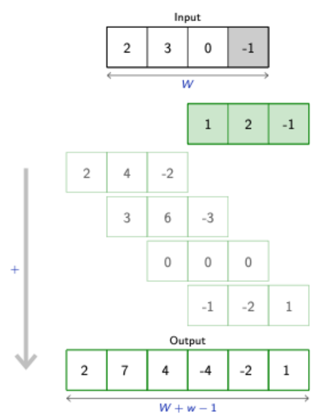



## Transposed convolution layer



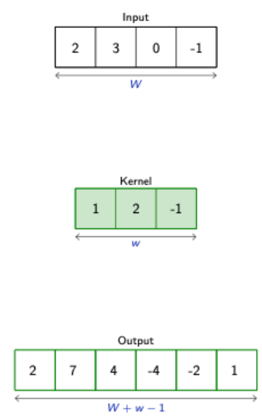



## Transposed convolution layer

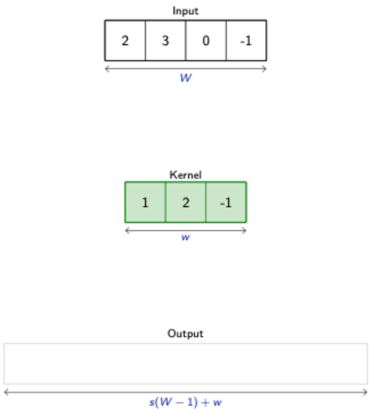





## Transposed convolution layer

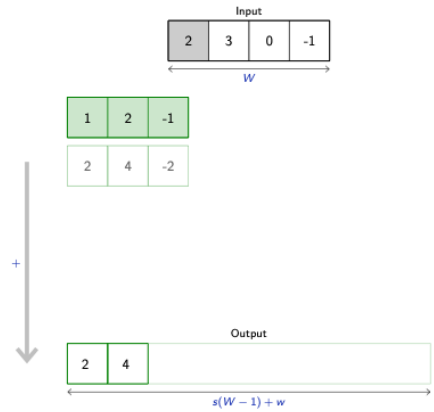


## Transposed convolution layer (stride = 2)



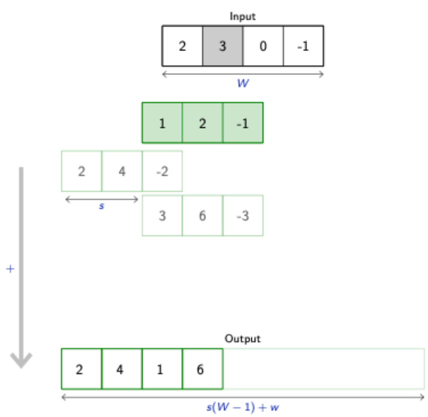



## Transposed convolution layer (stride = 2)



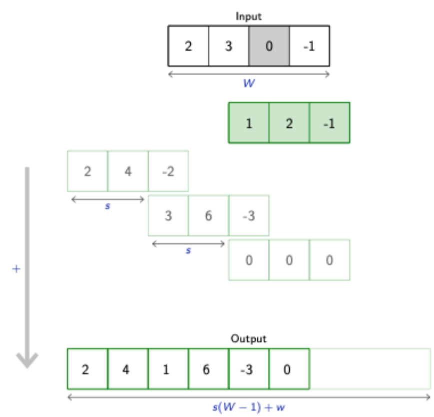



## Transposed convolution layer (stride = 2)

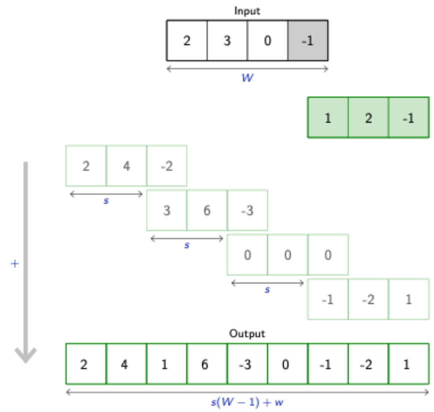





## Transposed convolution layer (stride = 2)

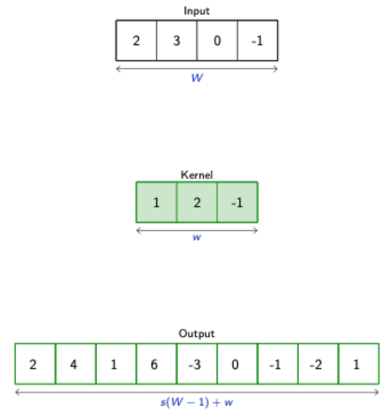


Transposed convolution layer (stride $= 2$)






Transposed convolution layer (stride $= 2$)






# Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1



# Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Transposed convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Example: 1D transposed conv, kernel size=3, stride=2, padding=0

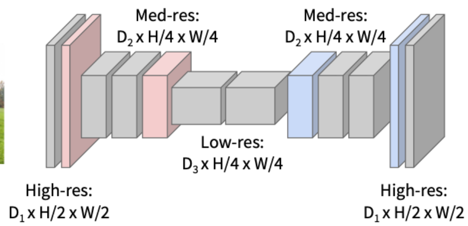
# Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!

Upsampling:
Unpooling or strided transposed convolution

Input:
3 x H x W




Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015



# Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels