

# CS342 - Section 7

- Homework 6
  - Due 10/11, can still turn it in until Sunday
  - How many layers does your model have?
- Homework 7
  - Posted, due next Thursday
  - On image segmentation
- Homework 5
  - Grades posted

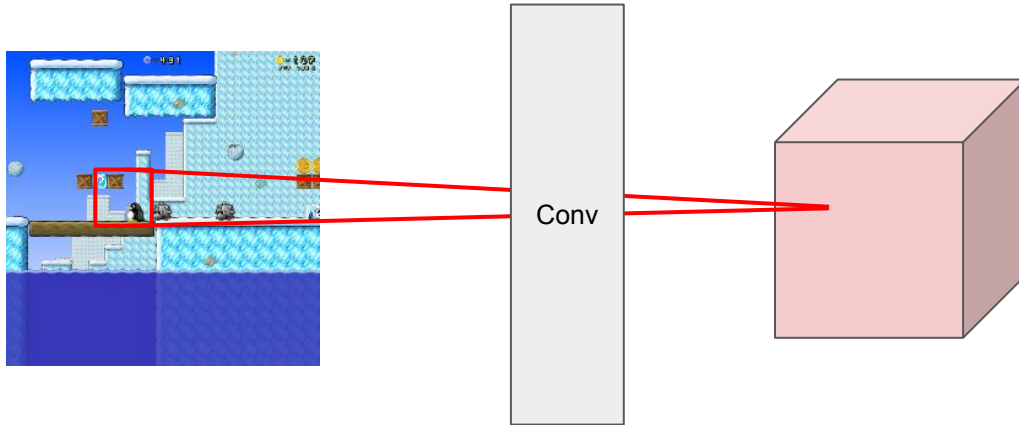
Questions?

# Agenda

- Receptive Field
- Fully connected layer  $\Leftrightarrow$  convolutional layer
- Up-convolution
- Linear-upsampling
- PyTorch code walk through on up-convolution/FCN

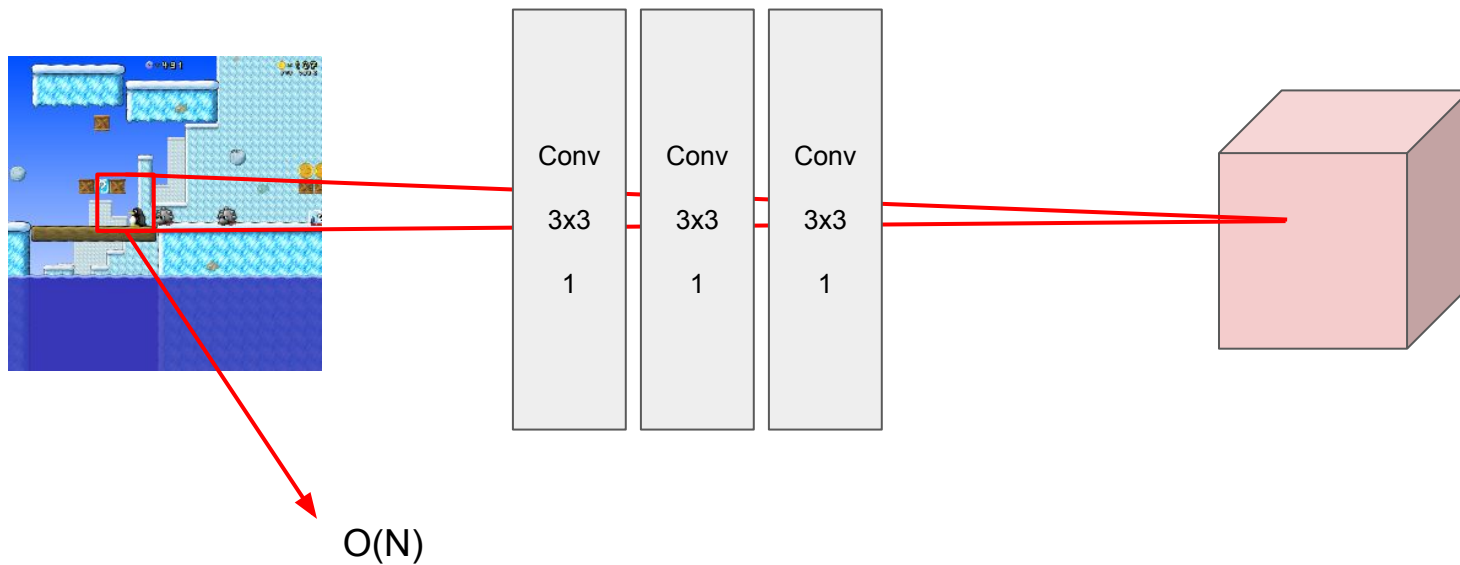
# Receptive Field

- Region of input that affect the value of certain pixel in later layers



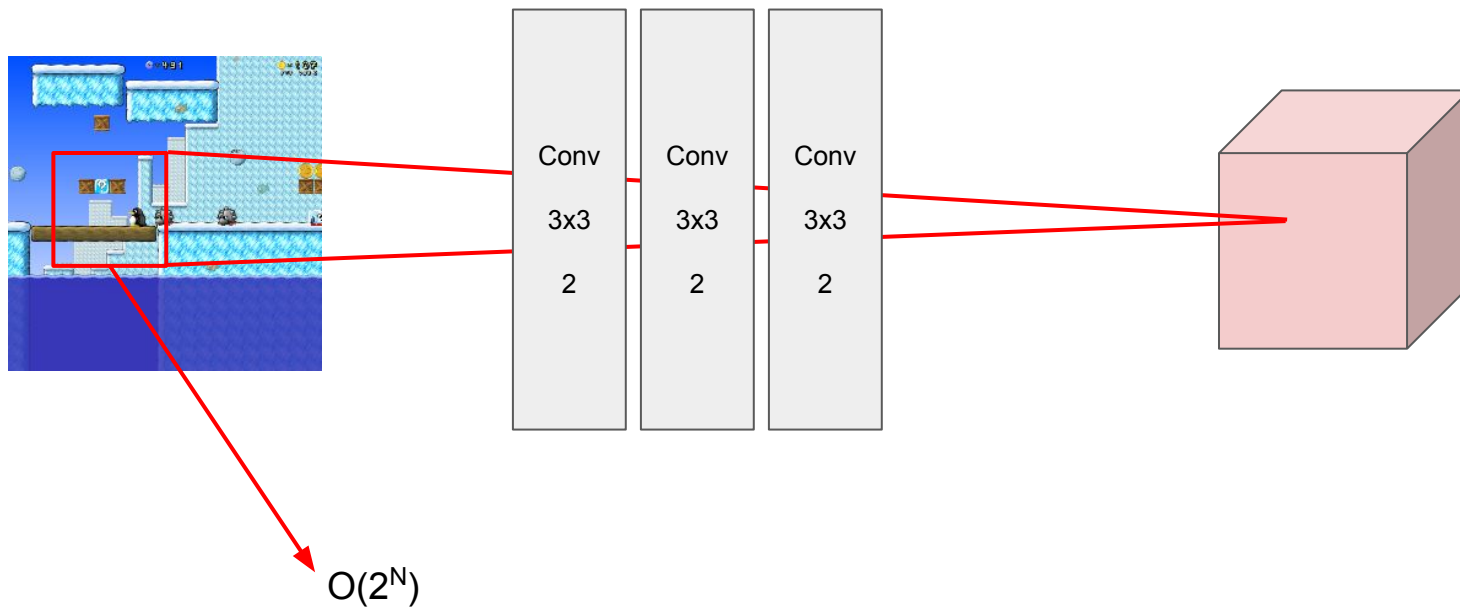
# Receptive Field

- How fast does it grow if there is no stride/pooling?



# Receptive Field

- How fast does it grow if there is stride/pooling?



# Receptive Field

- How to calculate the receptive field
  - By hand
    - Use kernel size, stride, pooling, pad in each layer
    - Corner cases can be really dirty
  - NaN trick
    - Pass in NaN as gradient, back prop to input to see how many end up NaN too
    - PyTorch Demo
    - <https://gist.github.com/dianchen96/8ac971ab667cc875619240cb2254425a>

# Receptive Field

- Online receptive field calculation
  - <http://fomoro.com/tools/receptive-fields/>

A convolutional layer operates over a local region of the input to that layer with the size of this local region usually specified directly. You can also compute the effective receptive field of a convolutional layer which is the size of the input region to the network that contributes to a layers' activations. For example, if the first convolutional layer has a receptive field of 3x3 then it's effective receptive field is also 3x3 since it operates directly on the input. However if the second layer of a convolutional network also has a 3x3 filter, then it's (local) receptive field is 3x3, but it's effective receptive field is 5x5.



## CONVOLUTIONAL FILTER INPUTS

Input Size

## NEW LAYER PROPERTIES

Kernel Size

Stride

Dilation

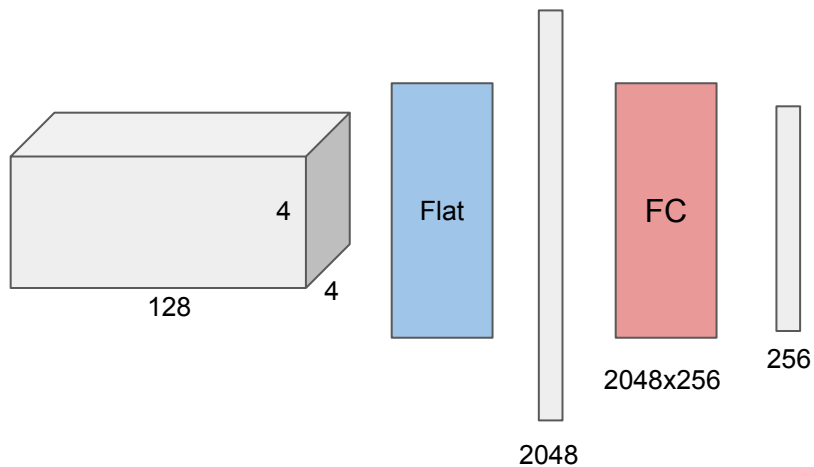
Padding

Layer #	Kernel Size	Stride	Dilation	Padding	Input Size	Output Size	Receptive Field
---------	-------------	--------	----------	---------	------------	-------------	-----------------



# Fully connected layer $\Leftrightarrow$ convolutional layer

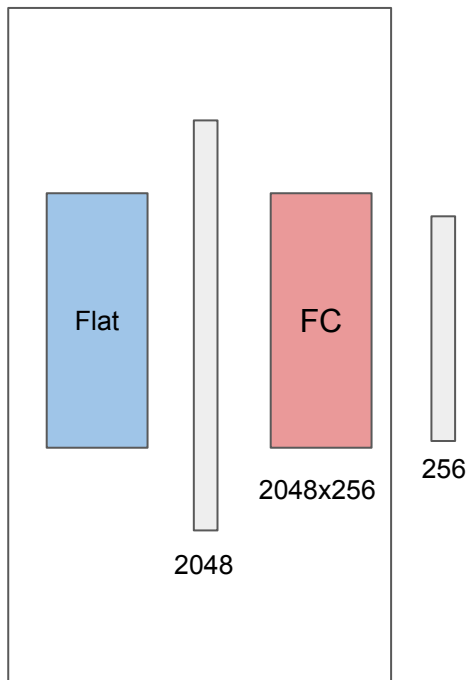
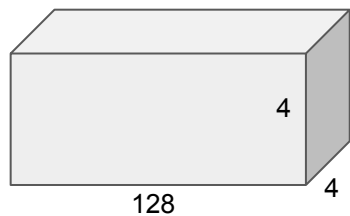
- FC  $\Rightarrow$  Conv



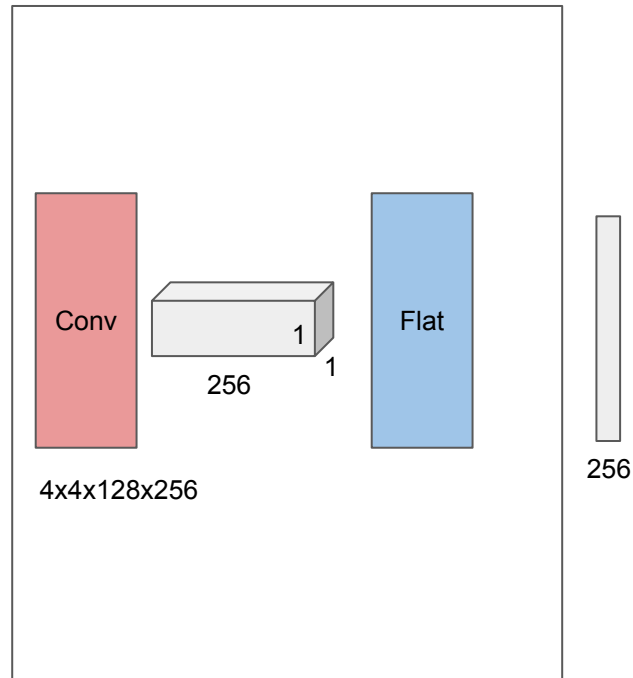
- How to convert this into convolution?

# Fully connected layer $\Leftrightarrow$ convolutional layer

- FC  $\Rightarrow$  Conv



$\Leftrightarrow$



# Linear Upsampling

- Recap
- One way to upsample
  - Pro: best results
  - Con: slow
- PyTorch
  - `torch.nn.Upsample(mode='bilinear')`

# Up-convolution

- Recap
- Another way to upsample
  - Pro: faster
  - PyTorch
- PyTorch
  - `torch.nn.ConvTranspose2d(...)`

# PyTorch Code walkthrough