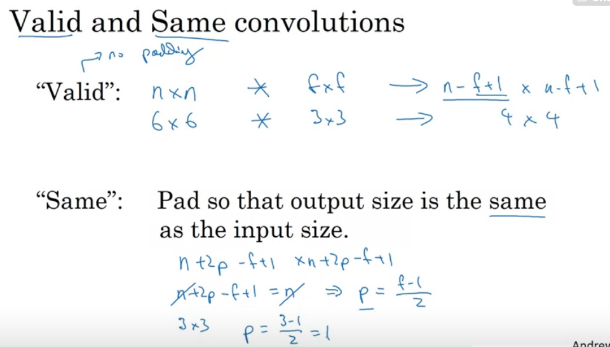
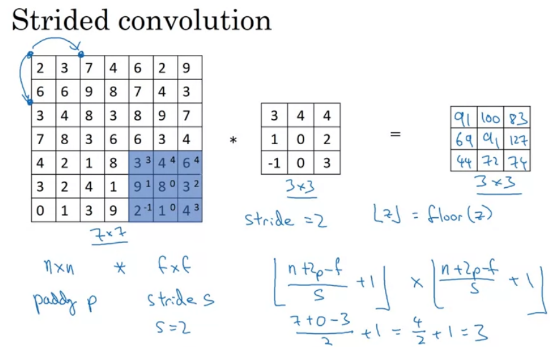
## Why convolutions:

* Parameter sharing: a feature detector that’s useful I one part of the image is probably useful in another part of the image as well.
* Sparsity of connections: in each layer, each output value depends only on a small number of inputs.

## 

## Padding and striding

## Pooling:

* Max or average; hyperparameters ‘filter’ (window) size (f) and stride (s). Typically max is used. \*\*Note that there are no parameters to learn!
* Intuition: If you think of a region in the output as some set of features, the activations in some layer of the neural network, then a large number, it means that it's maybe detected a particular feature. So, the upper left-hand quadrant has this particular feature. It maybe a vertical edge. So what the max operation does is a lots of features detected anywhere, and one of these quadrants , it then remains preserved in the output of max pooling. So, what the max operates to does is really to say, if these features detected anywhere in this filter, then keep a high number. But if this feature is not detected, so maybe this feature doesn't exist in the upper right-hand quadrant. Then the max of all those numbers is still itself quite small., marked from 0 hours 2 minutes 43 seconds until 0 hours 3 minutes 11 secondso what the max operation does is a lots of features detected anywhere, and one of these quadrants , it then remains preserved in the output of max pooling. So, what the max operates to does is really to say, if these features detected anywhere in this filter, then keep a high number. But if this feature is not detected, so maybe this feature doesn't exist in the upper right-hand quadrant. Then the max of all those numbers is still itself quite small.

## Conv layer hyperparameter tips:

* See how others have done this, and learn if they were successful. Common workflow is to loo for pre-trained architectures, e.g. on github.
* Relatively few parameters in conv layers, none in pool layer, but really a lot in the fully connected layers.
* Height and width goes down as you go deeper into the network, whereas the number of channels increases. Possibly, doubling the number of channels on each layer, like in VGG-16.
* Activation size goes down gradually as you go deeper into the network, don’t let it drop to fast.
* Common pattern: conv layer followed by pool, and repeating that cascaded, to finally get to fully connected layers.

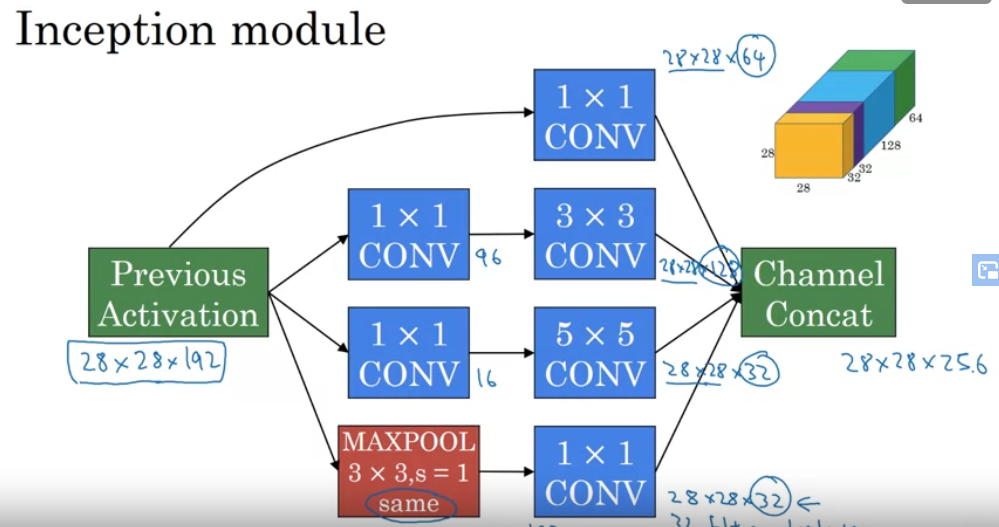
## Res nets:

* Shortcuts through network to get results much deeper into the network without going through all layers. Allow you to train much deeper networks.
* Residual block because the shortcut is added to the current activation result, before the ReLU non-linearity. Injected after the linear part.
* Why do they work? Because it is easy to learn the identity function, combat vanishing gradient I guess.

## 1x1 convolution

* Take inputs of multiple channels on same position in image to go to one neuron.

## Inception networks

* Why choose layer type and hyperparameters (convolution or pool layer), just try them all.
* Computational costs… introduce bottle neck layers based on 1x1 convolutions.
* Build network from inception blocks  
  

## Data augmentation

* Mirroring, random cropping, shearing, local warping
* Color shifting (PCA colour augmentation)

