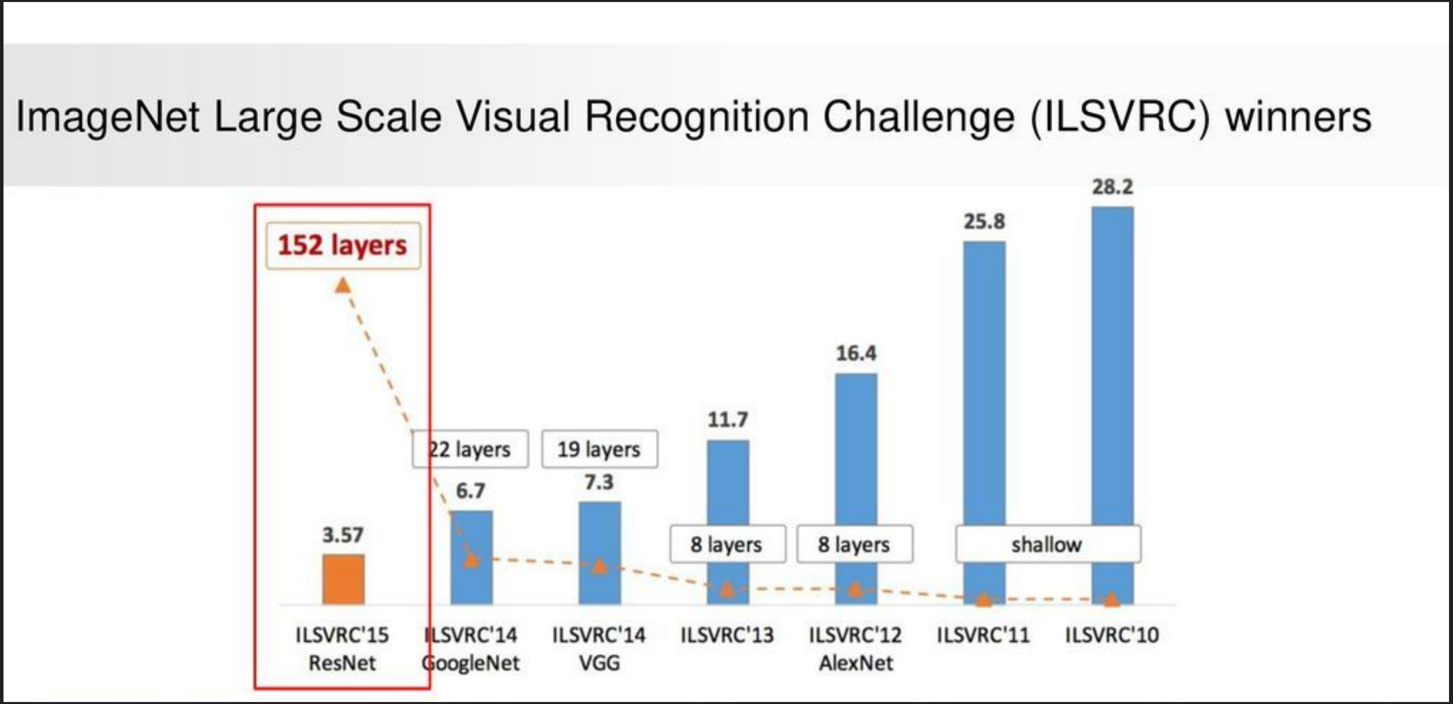
CNN Extensions and Other Comp Vision Architectures

Sources: Andrew Ng; F. Li, J. Johnson, and S. Yeung; and S. Tsang

Successful Computer Vision Architectures:

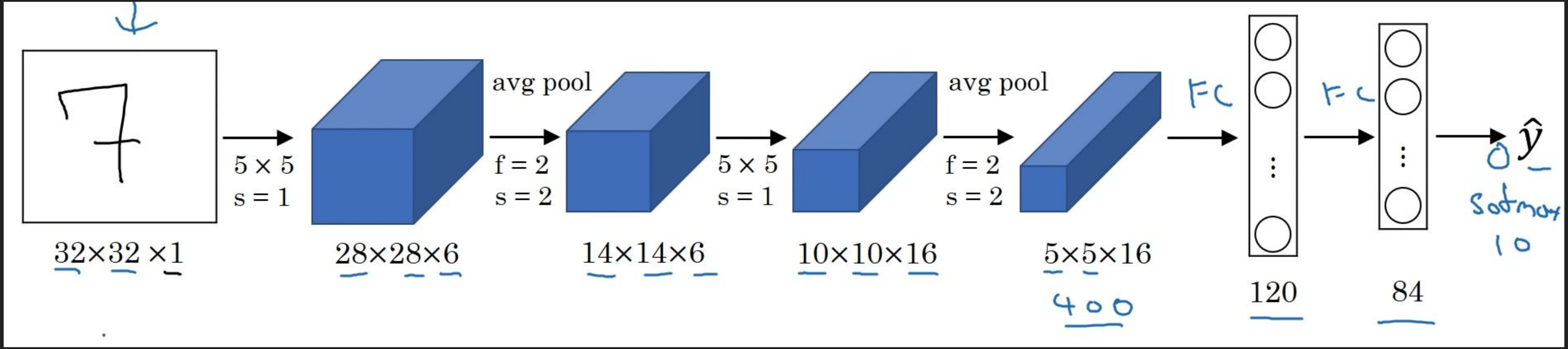


Today

- Overview of following architectures:
  - Lenet
  - Alexnet
  - Vgg
  - Resnet
  - GoogLenet
- Explanation of key innovations needed to understand new models along the way:
  - 1D Convolutions
  - Skip traces
  - Inceptions Modules (i.e. parallel operations with stacking)

Classic architectures: LeNet 5

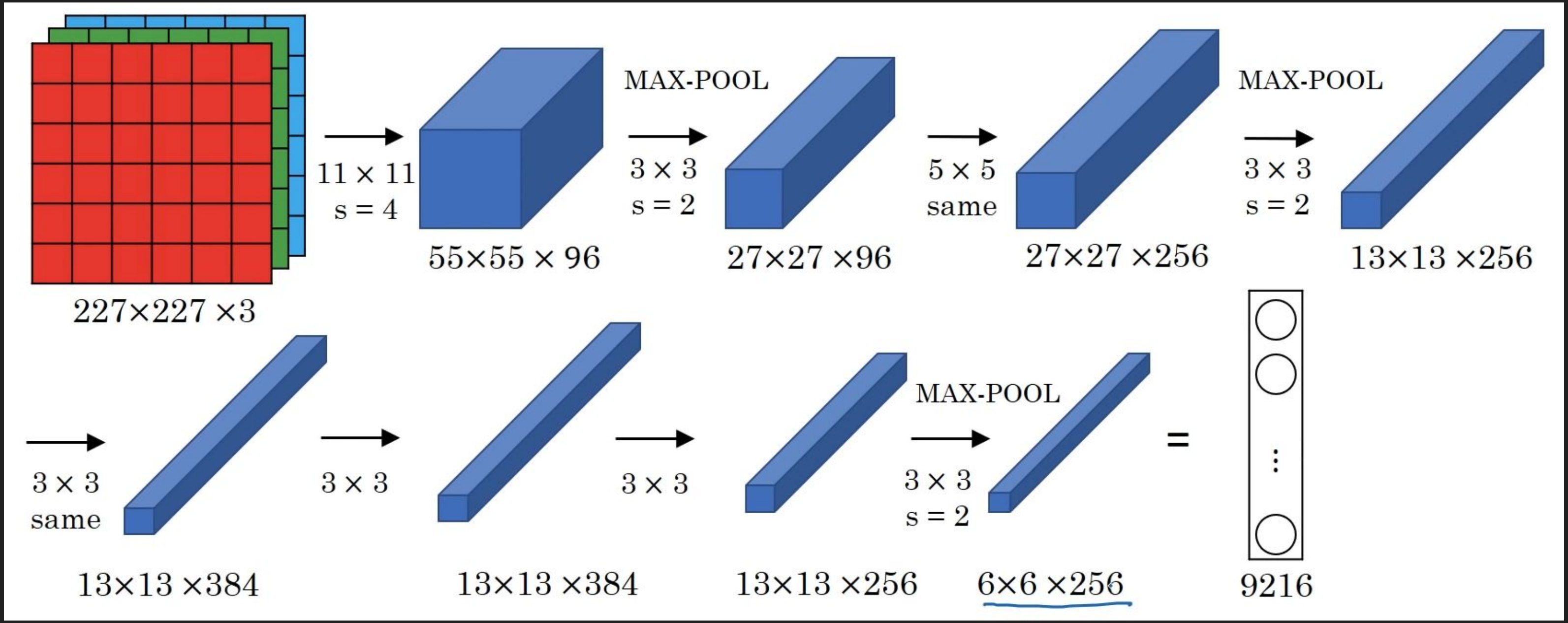
- Original paper by Lecun et al. in 1998
- Paper used architecture to predict handwritten digits
- Small by modern standards: about 60k parameters
- Layer shape:
  - Size of H x W goes down further in network
  - Channels get larger



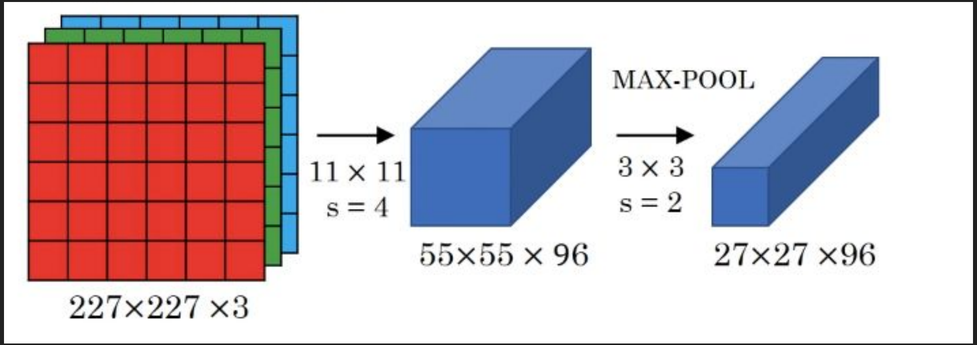
Classic architectures: Alexnet

- Original paper by Krizhefsky et al. in 2012
- Used to win 2012 Imagenet competition (y_train=1000 image cats)
- Similar structure to Lenet but bigger:
  - about 3m parameters
  - Also used Relu, which LeNet didn’t use.
  - First, CNN based winner of competition
  - Made more people take CNN seriously
- Let’s look at it layer by layer...

Classic architectures:
Alexnet full architecture up to first FC layer

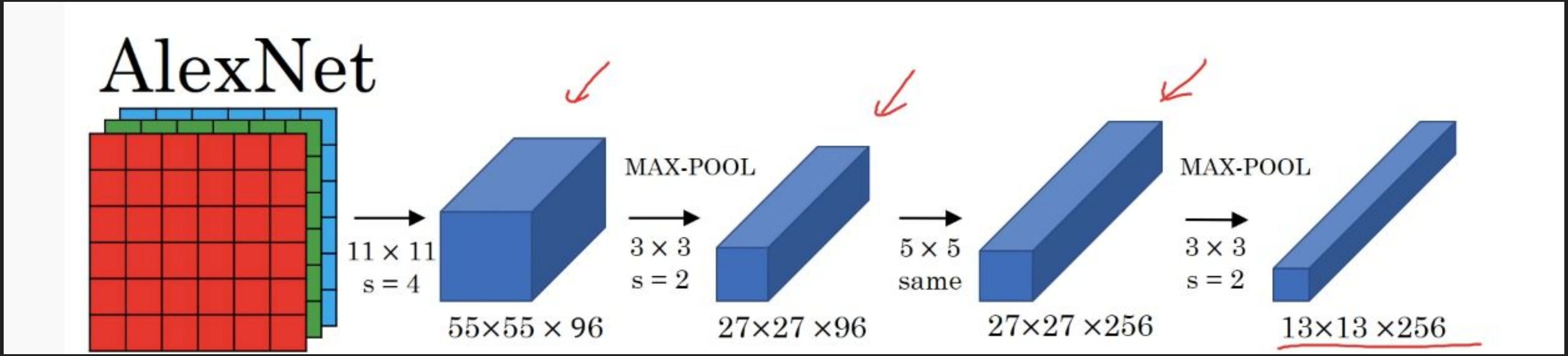


Classic architectures:
Alexnet partial architecture

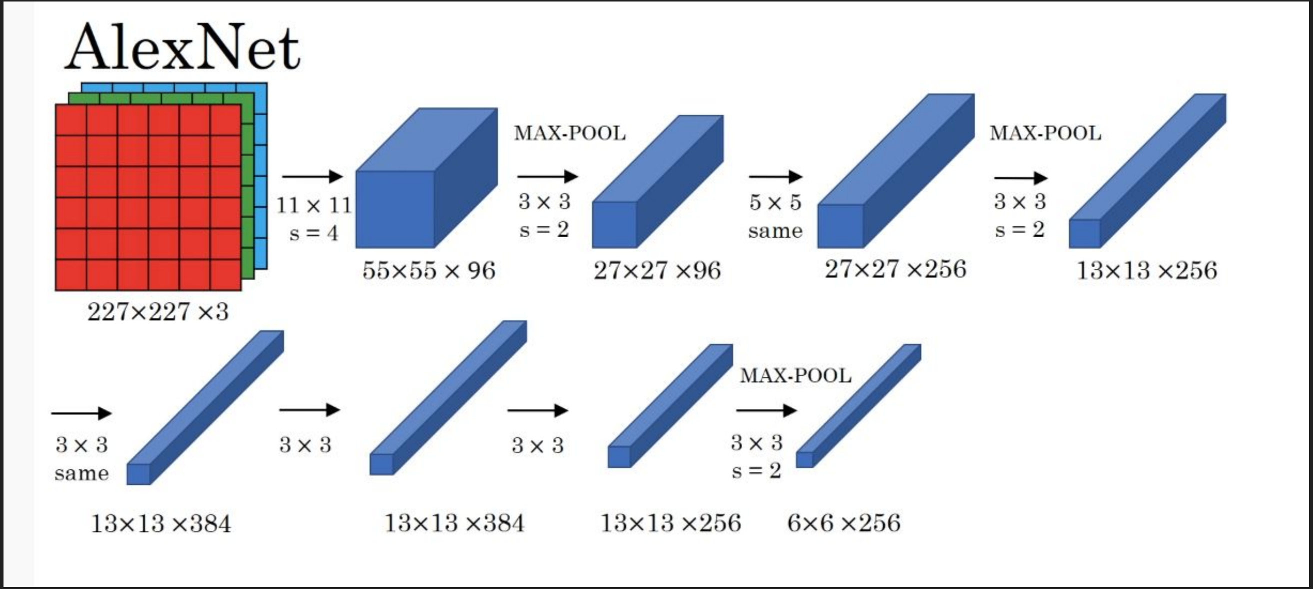


- Input: RGB
- Conv filter size = 11 x 11
  - 96 filters with h x w reduced to 55 after convolution
- Relu transf of filter weights and biases before stacking
- Max pool = 3 x 3 with stride of 2 (with equal channels to filters)

Classic architectures:
Alexnet partial architecture



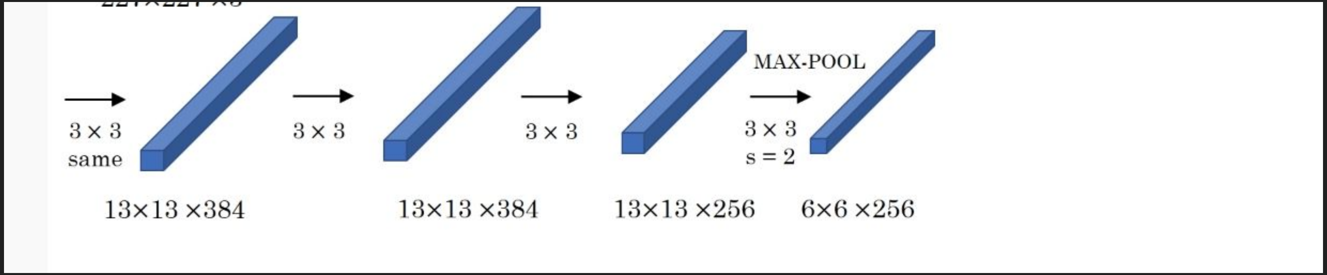
Classic architectures: Alexnet partial architecture



This is different! >>>>>>

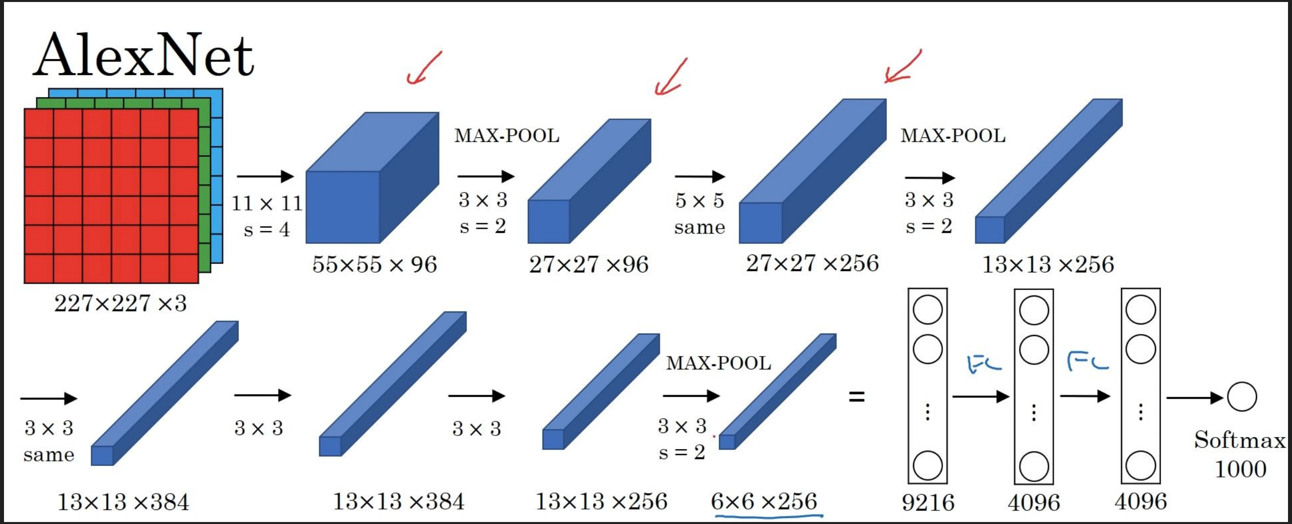
Classic architectures:
Alexnet partial architecture

This is different, so let's take a closer look...



- Multiple 3 x 3 convolutions with "same" padding in a row before adding max pooling
  - 384 3 x 3 filters with same padding then
  - 384 3 x 3 filters with same padding then
  - 256 3 x 3 filters with same padding
  - Then max pool with 3 x 3 filters with stride of 2.

Classic architectures: Alexnet full architecture



Notice addition of:

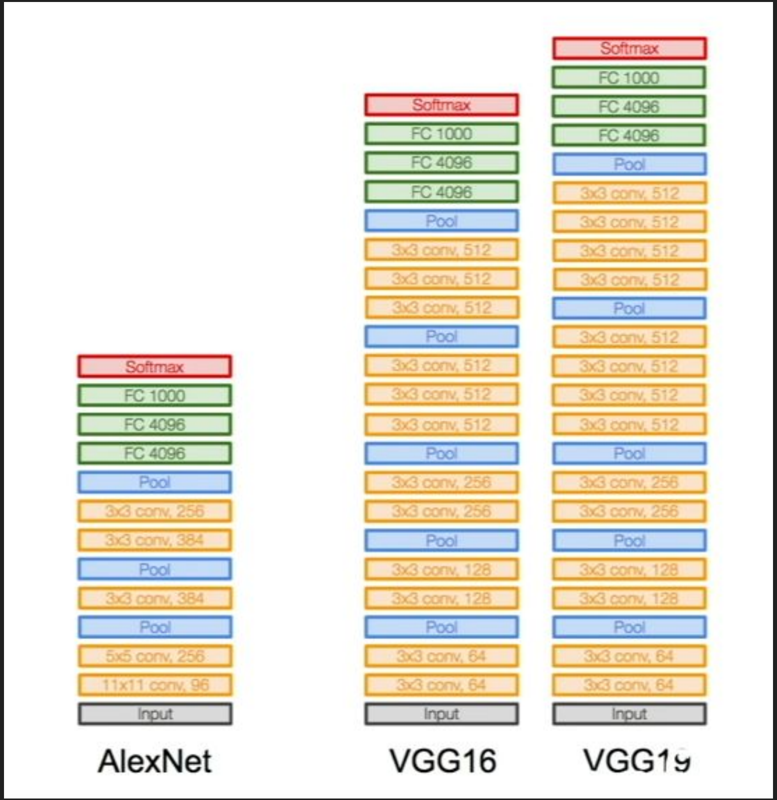
1. Flattened layer (6 times 6 times 256 flattens to 9216 input values) and…
2. Two hidden FC layers with 4096 nodes each and…
3. 1000 output nodes with a softmax transformation

Classic architectures: VGGnet (or VGG16)

- Paper by Samonyon and Zisserman published in 2015
- Used to win 2014 Imagenet competition (y_train=1000 image cats)
- New idea: Create “very deep network” with a bunch of similarly structured layers:
  - about 138m parameters
  - Relu transformations.
  - But very deep (more layers!!)
  - Uses simple uniform filter size throughout network
- Let’s look at it layer by layer...

Classic architectures: VGGnet (or VGG16 or 19)

- Notice
  - the 3 x 3 filter size with “same padding” and stride of one used throughout
  - Never pools immediately after a single convolution filter (conv filters repeat)
  - Doubles number of filters after each pool
  - Pooling uses 2 by 2 filters and stride of 2
- H x W not shown in visual, but it shrinks by half after each pool from:
  - input= 224 x 224 to
  - 112 to 56 to 28 to 14 to 7 before FC layer
- Again channels increase while h x w decreases.

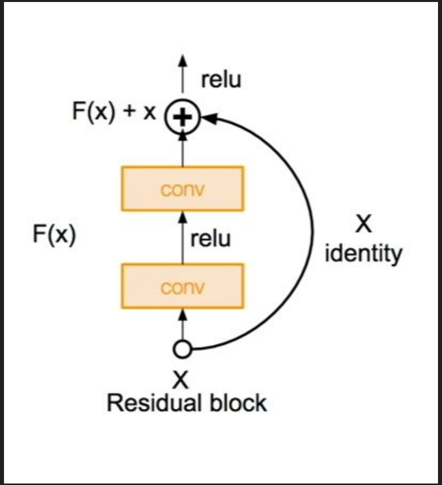


High performance architectures: ResNet

- Deep conv networks can have issues with optimizing and updating parameters
  - In other words, forward/back propagation becomes less effective.
    - Gradient adjustments become less effective.
  - Need new approaches that correct for these problems.
- Enter Residual Networks:
  - New idea: Use skip connections in residual blocks
- One of the current best approaches
  - Won 2015 ImageNet over 96% accuracy

High performance architectures: ResNet, What is a Residual block??

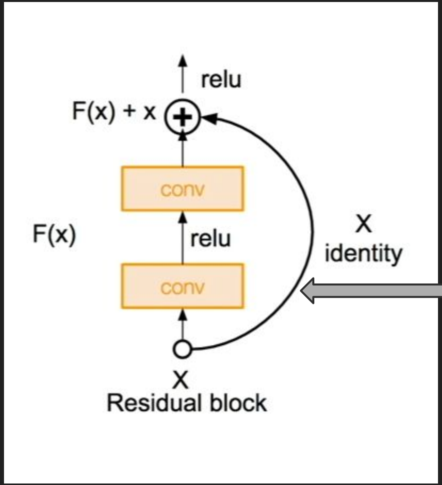
1. X is input to two normal conv layers with relu transformations
2. But we take X input data and add it to pre relu transformed output data after the second convolution.



This represents a single residual block!

So we add X input to the output of the convolved output from the second convolved layer.

High performance architectures: ResNet

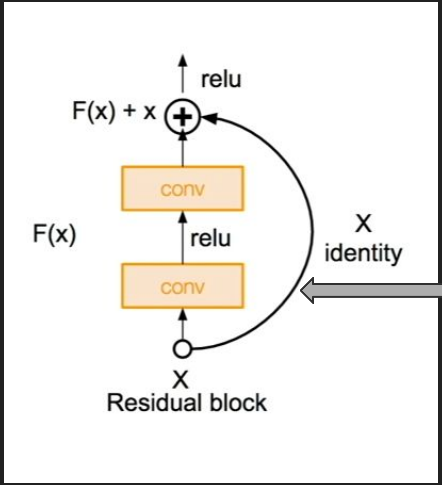


This is called a <u>skip connection.</u>

If output is zero as a result of weight and bias transformation within conv. layers, this will add previous input to carry information forward in network.

Allows us to train really deep networks!

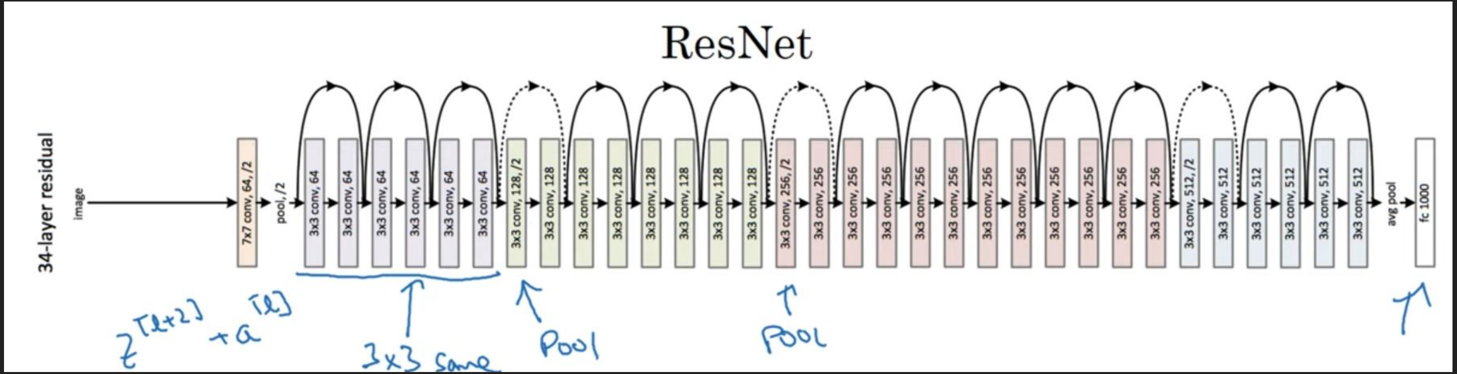
High performance architectures: ResNet



This is called a skip connection.

Also note that for skip connections we sometimes transform X to ensure it is the same size as the output data from conv layer #2. This transformation can take the form of W*X data

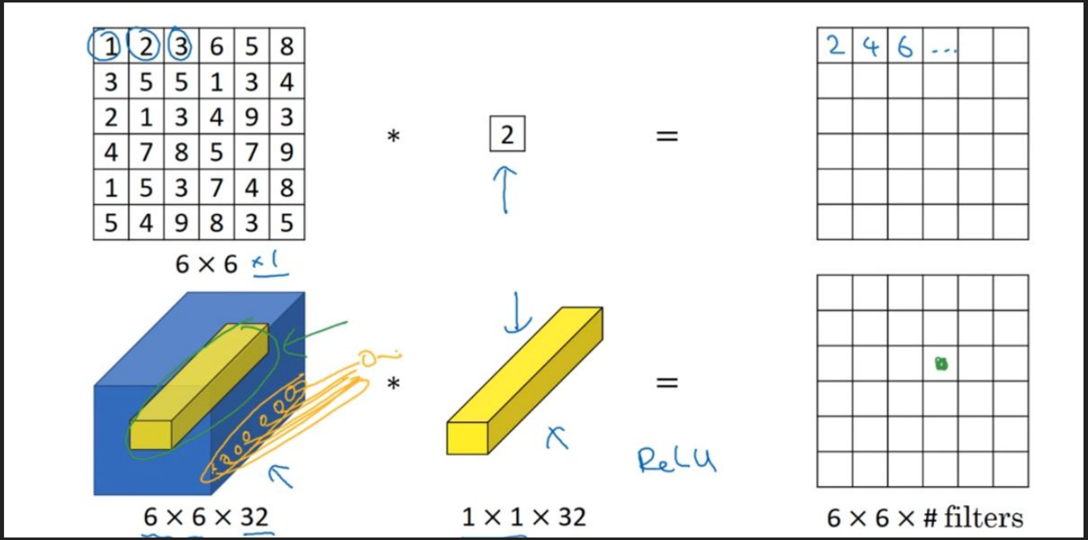
Resnet architecture overview



- Prior to dotted lines pooling is used.
    - Note that after pooling we double number of conv filters in next series of residual blocks.

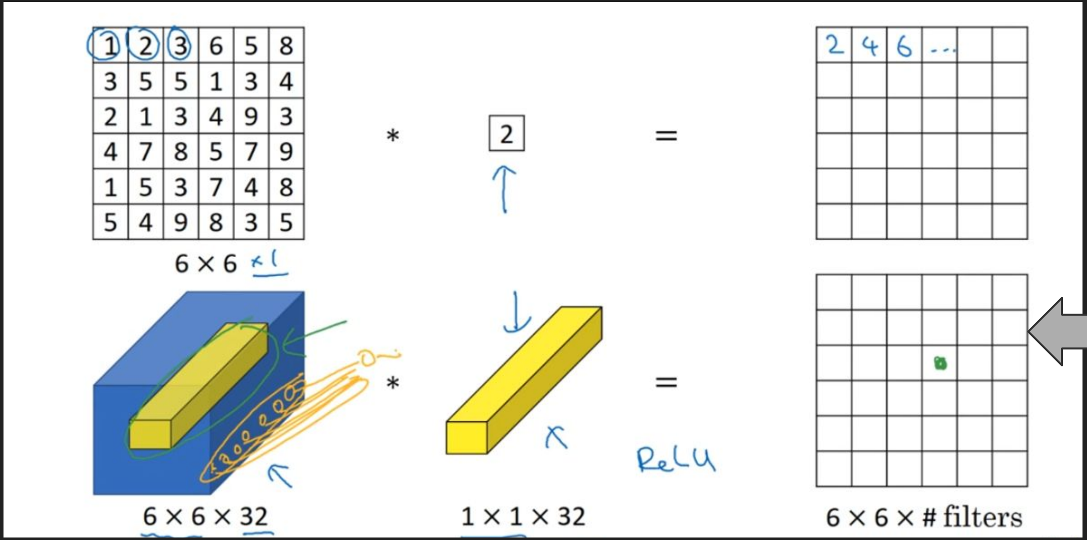
Other important extensions:
1D Convolutions

- Allows us to adjust # of filters (and lower # of network parameters)



Other important extensions: 1D Convolutions

- Allows us to adjust # of filters (and lower # of network parameters)

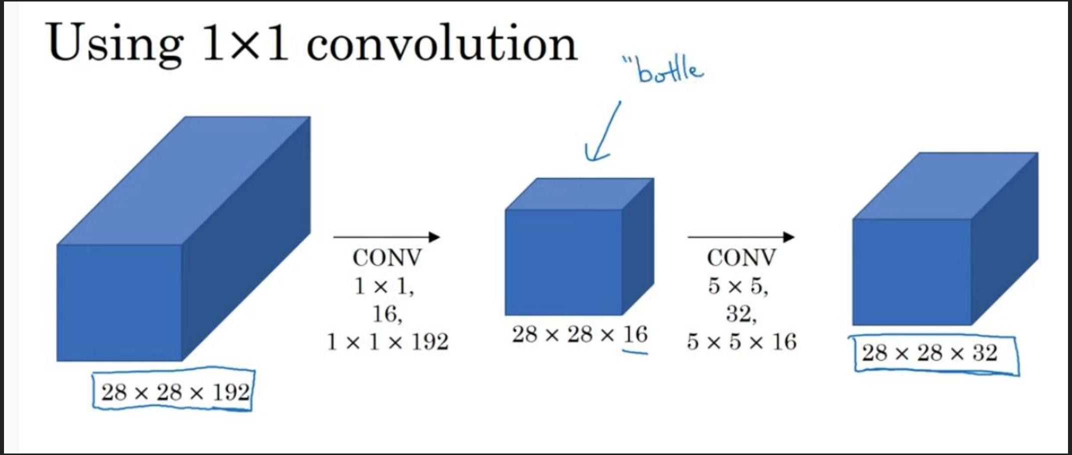


With a single 1d convolution filter we can reduce 32 channels to 1 channel!!

Add filters for # of channels you want in result.

Inception: “We need to go deeper!”

- Example of conv1d utility:
- With conv1d operations we can control # of parameters. Implement a “bottleneck layer”



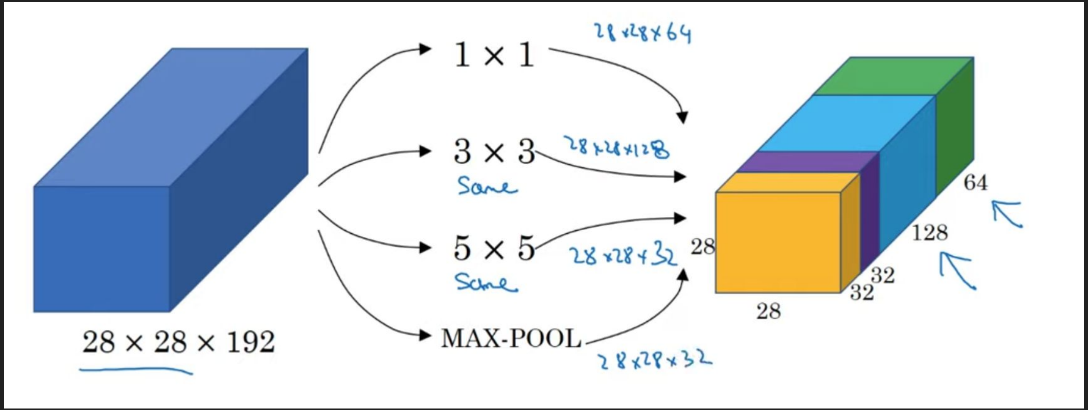
Shrinking 192 input channels to 16 using 16 conv1d filters.

Then we expand size again with 32 conv2d layers.

Shrinks # of parameters dramatically without hurting model performance!

Inception: “We need to go deeper!”

- What if we built a single network layer that combines several layer types into one concatenated output?

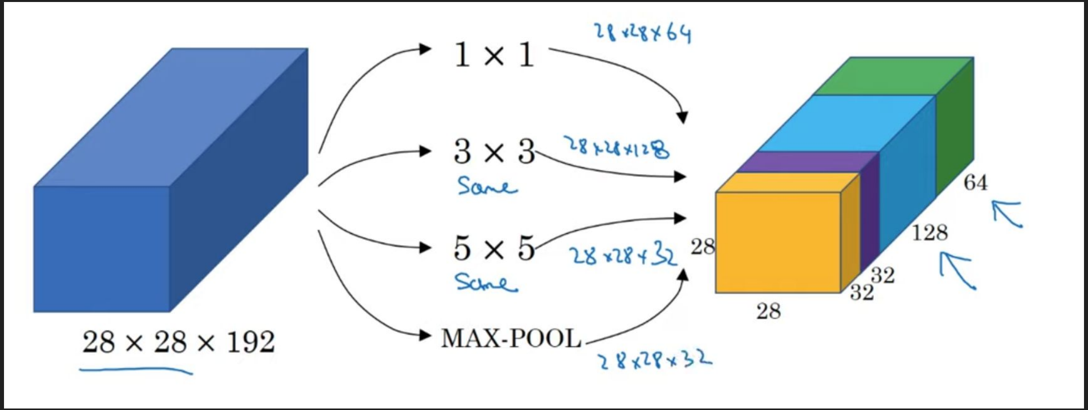


This is the logic behind an inception module/layer.

Note that Max pooled input will also use same padding such that output h x w will have stackable dimensions.

Inception: “We need to go deeper!”

- What if we built a single network layer that combines several layer types into one concatenated output?

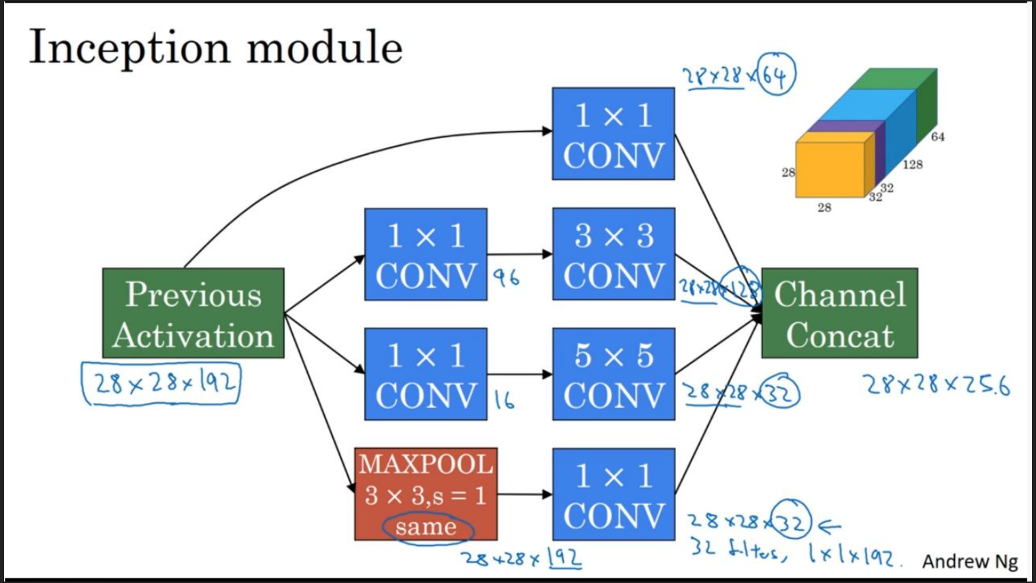


Problem: Results in a large number of parameters.

With conv1d operations we can control # of parameters.

Inception: “We need to go deeper!”

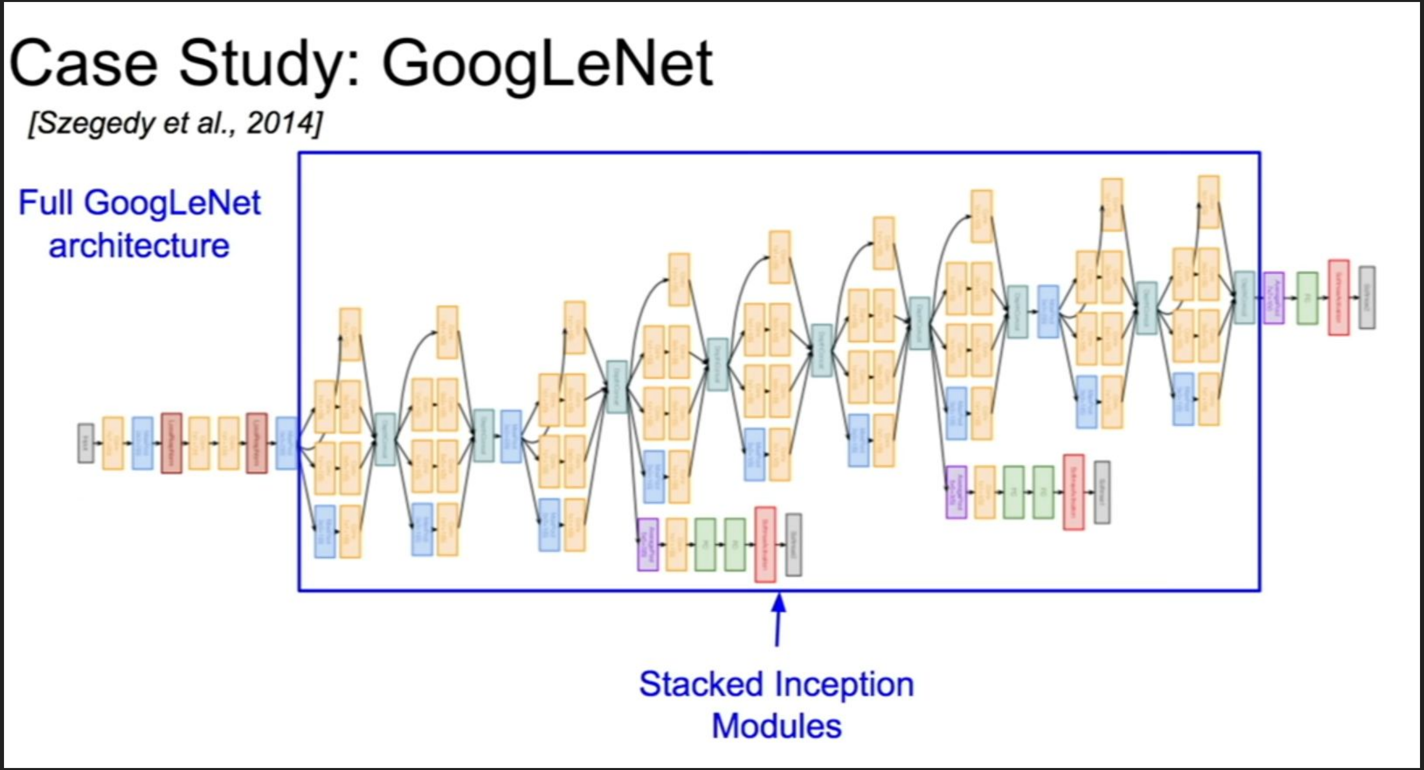
- Need to use conv1d to implement inception module successfully



1. Takes 28 x 28 x 192
2. In sublayer 1
   a. Feeds input to
      i. Conv1d with 96 filters
      ii. Conv1d with 16 filters
      iii. 3 x 3 stride 1 maxpool filter w/ same padding, 192 ouput channels
   b. Notice all have same h x w for output stacking
3. In sublayer 2
   a. Feeds input to:
      i. Conv1d with 64 filters
      ii. Conv2d with 128 filters
      iii. Conv2d with 32 filters
      iv. Conv1d with 32 filters
   b. Notice all but one ingest input from sublayer 1

Inception: “We need to go deeper!”

- Full architecture GoogLeNet, won Imagenet competition in 2014



Inception: “We need to go deeper!”

- Full architecture GoogLeNet, won Imagenet competition in 2014

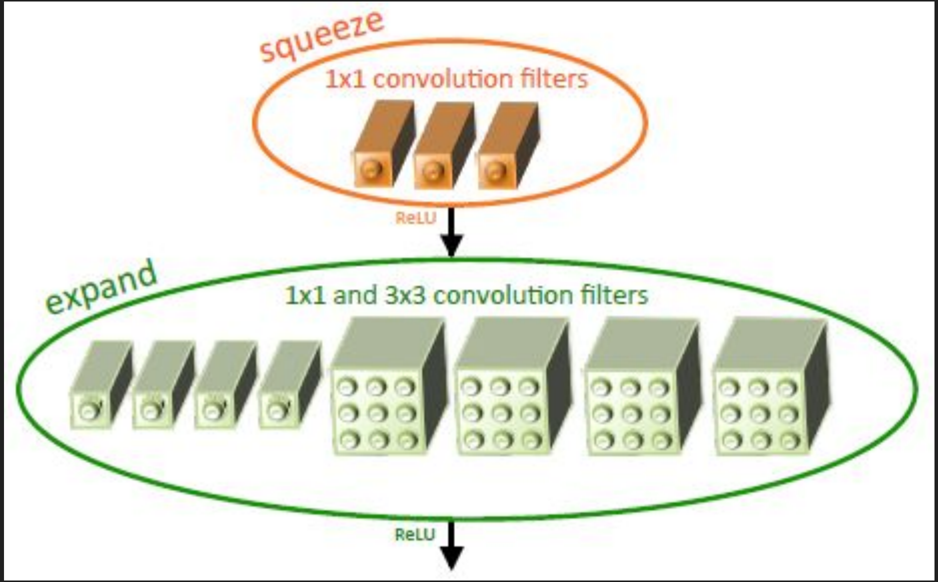
- Notice that side branches are also a part of this architecture.
  - Adds a mini network that sends output to intermediate softmax output
  - Appears to have regularizing effect on inception network
- Key is to understand new inception modules. If you do you understand inception networks like the original GoogLenet approach.

Squeezenet: Models that are made up of “fire modules”

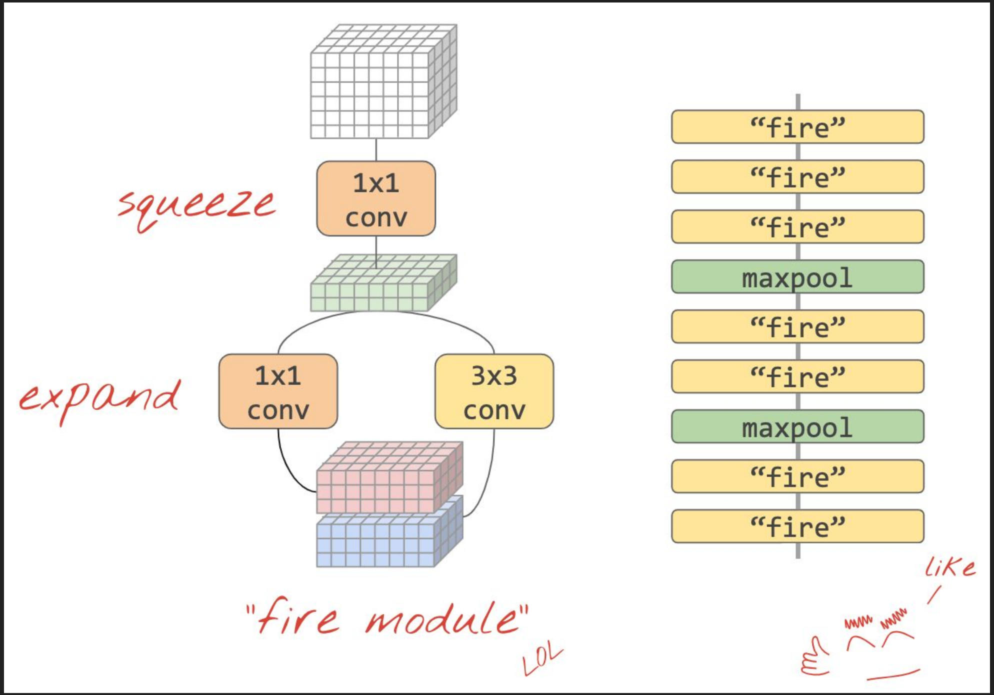
Fire modules minimize parameters and shrink total model size….

We shrink parameters with conv1d and then expand them again with conv2d!

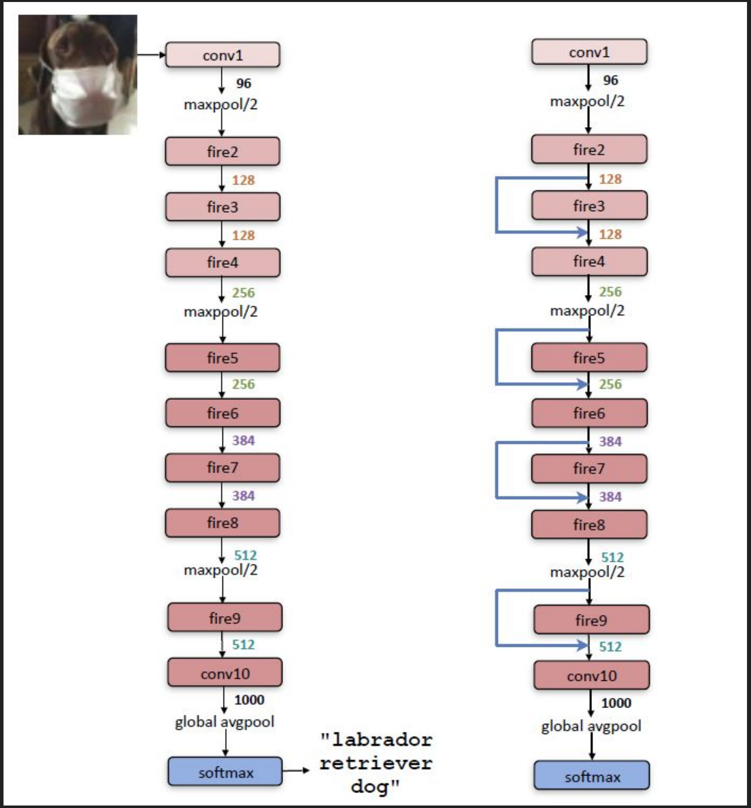
But they can predict as well as AlexNet architectures!!



Squeezenet: Made up of “fire modules”



Squeezenet: Made up of "fire modules"



Note that we can also add skip connections to this architecture!