CIS 522

...

Convnets

Today no admin

What we are doing next week

Implement Alexnet

Testing

Evaluation

Ethics

Think more about projects

What we learned about convnets

The nature of convolutions

What they are useful for

Intuitions of how they work

Maxpooling

Training loop for convnets

Data augmentation

A convolution exercise

...

A convolution case

Suppose we want to find out whether the following image depicts Cartesian axes.

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & -2 & 0 & -2 \end{bmatrix}$$

We do so by convolving the image with two filters (3*3, no padding, stride of 1)

$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}, \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ 1 & 1 & 1 \\ -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

Compute the output by hand.

A convolution exercise

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & -2 & 0 & -2 \end{bmatrix}$$

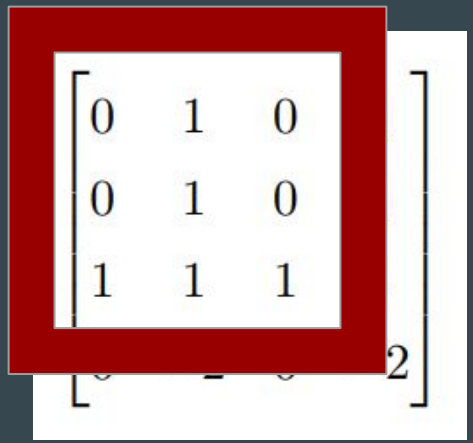
$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

What is the result of the convolution. use [a,b;c,d] format

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

A convolution exercise



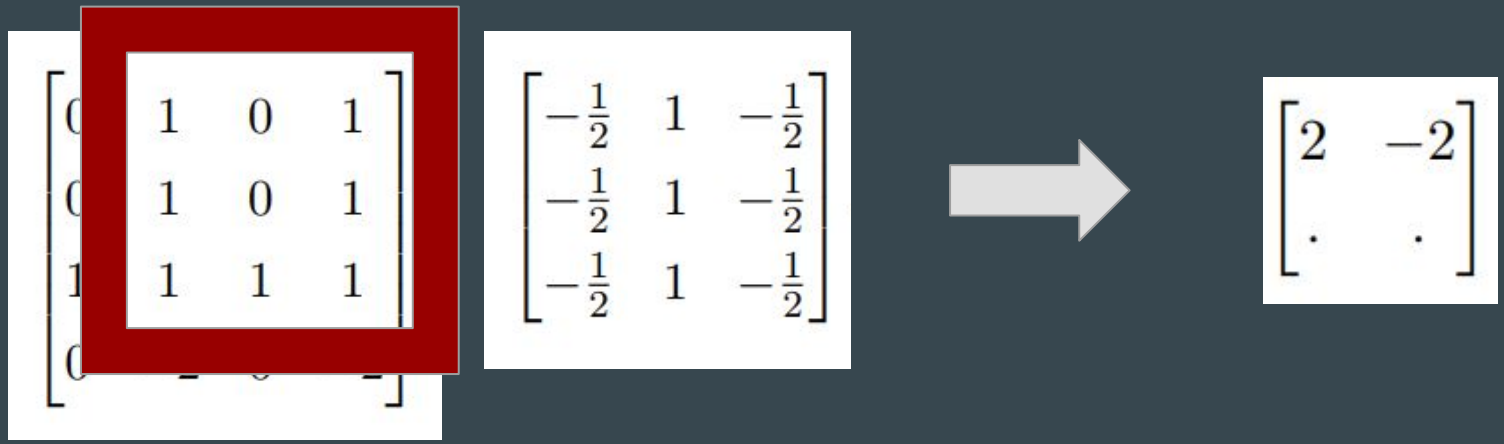
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

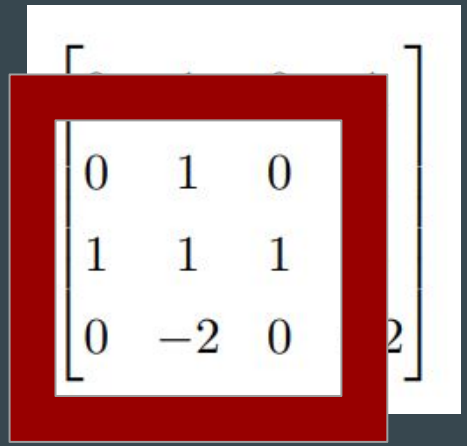
$$\begin{bmatrix} 2 & . \\ . & . \end{bmatrix}$$

$$\left(0 \times \frac{-1}{2}\right) + (1 \times 1) + \left(0 \times \frac{-1}{2}\right)$$
$$\left(0 \times \frac{-1}{2}\right) + (1 \times 1) + \left(0 \times \frac{-1}{2}\right)$$
$$\left(0 \times \frac{-1}{2}\right) + (1 \times 1) + \left(0 \times \frac{-1}{2}\right) = 2$$

A convolution exercise



A convolution exercise

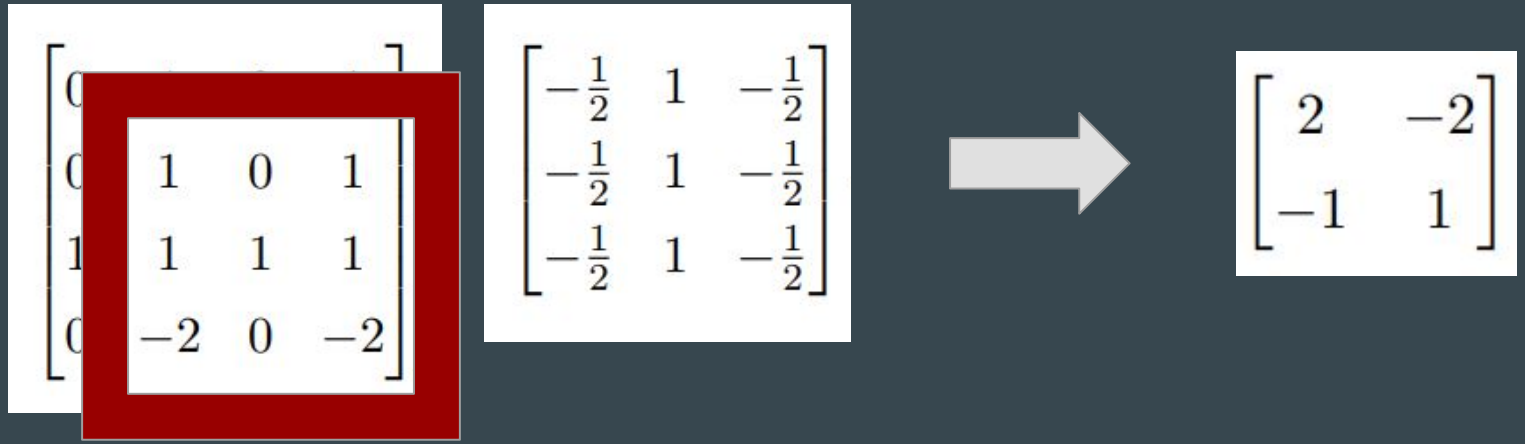


$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & -2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

$$\begin{bmatrix} 2 & -2 \\ -1 & \cdot \end{bmatrix}$$

A convolution exercise

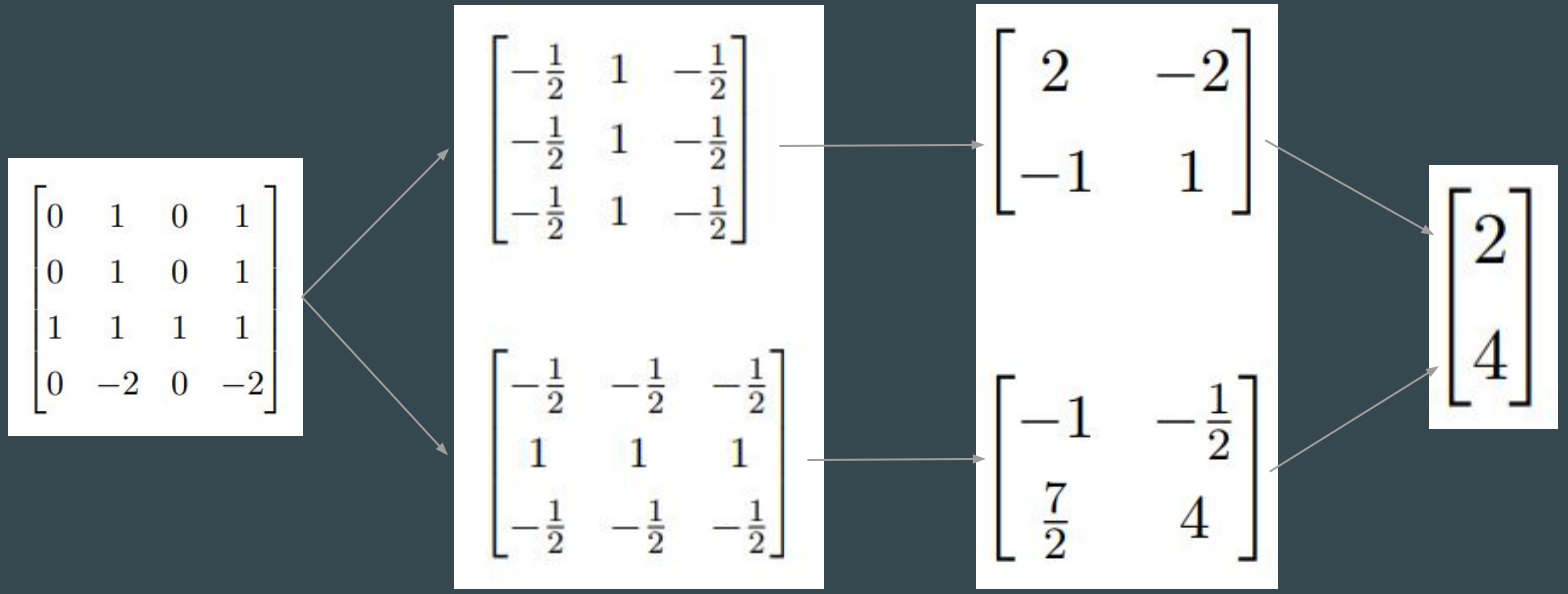


Lets now do some max-pooling

Lets now use both filters and

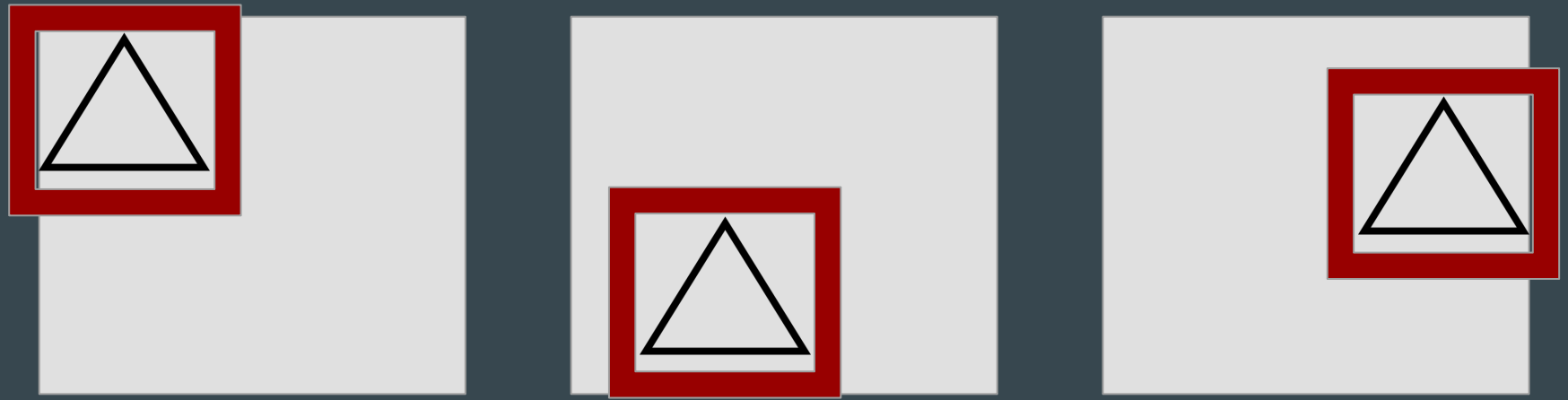
apply a max-pool operation with kernel of (2*2).

Convolution exercise solution



Grayscale image | Filters | Output of filters | Output of max-pool

CNNs are translation-invariant by design.

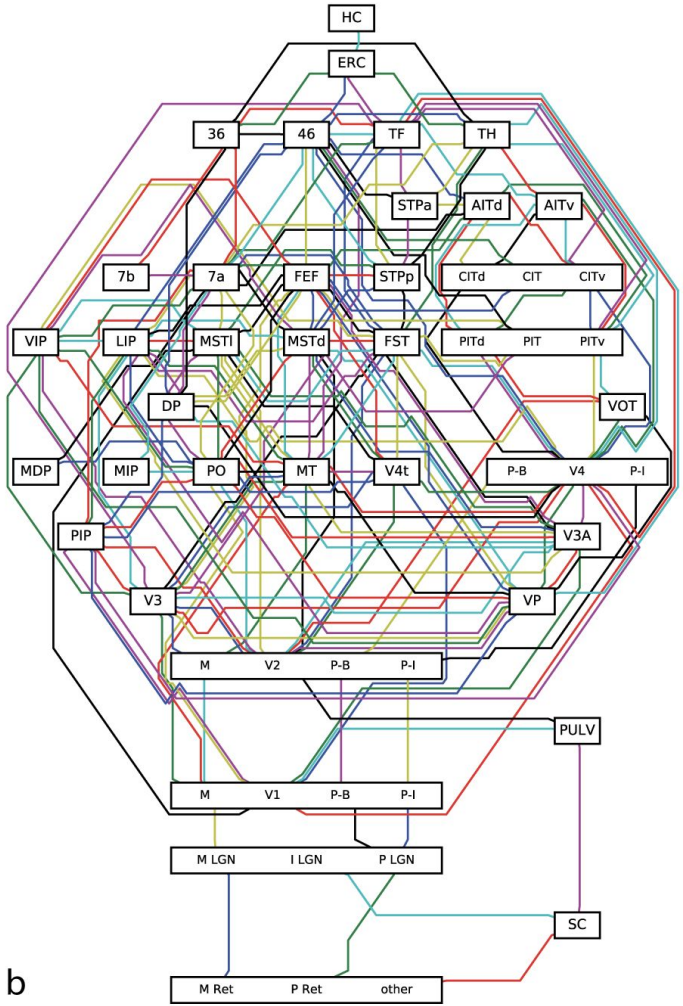


Although we would like many other kinds of invariance,
none are baked into the architecture of CNNs.

Some classic CNNs

...

Architectures as inductive biases



CNNs perform extremely well on CV datasets.

CNNs perform extremely well on MNIST.

Type	Classifier	Distortion	Preprocessing	Error rate (%)
Linear classifier	Pairwise linear classifier	None	Deskewing	7.6[9]
K-Nearest Neighbors	K-NN with non-linear deformation (P2DHMDM)	None	Shiftable edges	0.52[18]
Boosted Stumps	Product of stumps on Haar features	None	Haar features	0.87[19]
Non-linear classifier	40 PCA + quadratic classifier	None	None	3.3[9]
Support vector machine	Virtual SVM, deg-9 poly, 2-pixel jittered	None	Deskewing	0.56[20]
Deep Neural network	2-layer 784-800-10	None	None	1.6[21]
Deep Neural network	2-layer 784-800-10	elastic distortions	None	0.7[21]
Deep neural network	6-layer 784-2500-2000-1500-1000-500-10	elastic distortions	None	0.35[22]
Convolutional neural network	6-layer 784-40-80-500-1000-2000-10	None	Expansion of the training data	0.31[15]
Convolutional neural network	6-layer 784-50-100-500-1000-10-10	None	Expansion of the training data	0.27[23]
Convolutional neural network	Committee of 35 CNNs, 1-20-P-40-P-150-10	elastic distortions	Width normalizations	0.23[8]
Convolutional neural network	Committee of 5 CNNs, 6-layer 784-50-100-500-1000-10-10	None	Expansion of the training data	0.21[17]

CNNs perform reasonably well on CIFAR-10

Research Paper	Error rate (%)	Publication Date
Convolutional Deep Belief Networks on CIFAR-10[6]	21.1	August, 2010
Densely Connected Convolutional Networks[7]	5.19	August 24, 2016
Wide Residual Networks[8]	4.0	May 23, 2016
Neural Architecture Search with Reinforcement Learning[9]	3.65	November 4, 2016

LeNet (1998) -- Background

- Developed by Yann LeCun
  - Worked as a postdoc at Geoffrey Hinton's lab
  - Chief AI scientist at Facebook AI Research
  - Wrote a whitepaper discovering backprop.
  - Co-founded ICLR
- Problem: classify 7x12 bit images of 80 classes of handwritten characters.

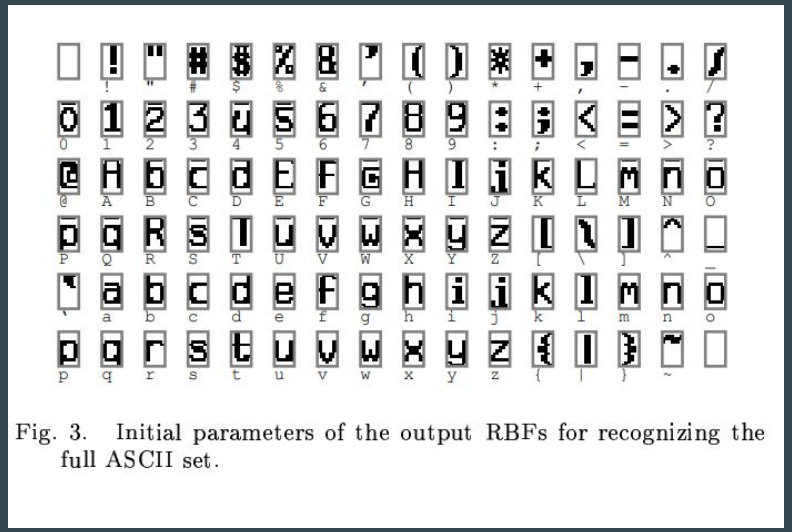


Fig. 3. Initial parameters of the output RBFs for recognizing the full ASCII set.

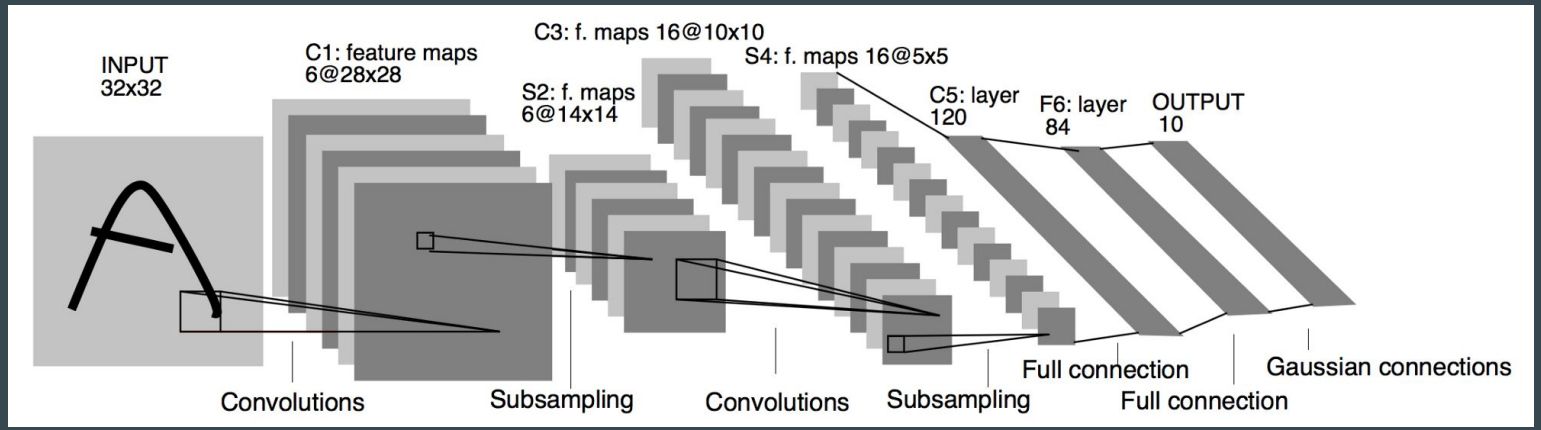
LeNet (1998) -- Architecture

- Convolution filter size: 5x5.
- Subsampling (pooling) kernel size: 2x2.
- Semi-sparse connections.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.



LeNet (1998) -- Results

- Successfully trained a 60K parameter neural network without GPU acceleration!
- Solved handwriting for banks -- pioneered automated check-reading.
- 0.8% error on MNIST; near state-of-the-art at the time.
    - Virtual SVM, kernelized by degree 9 polynomials, also achieves 0.8% error.

LeNet: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

SVM: http://yann.lecun.com/exdb/publis/index.html#lecun-98

LeNet in PyTorch

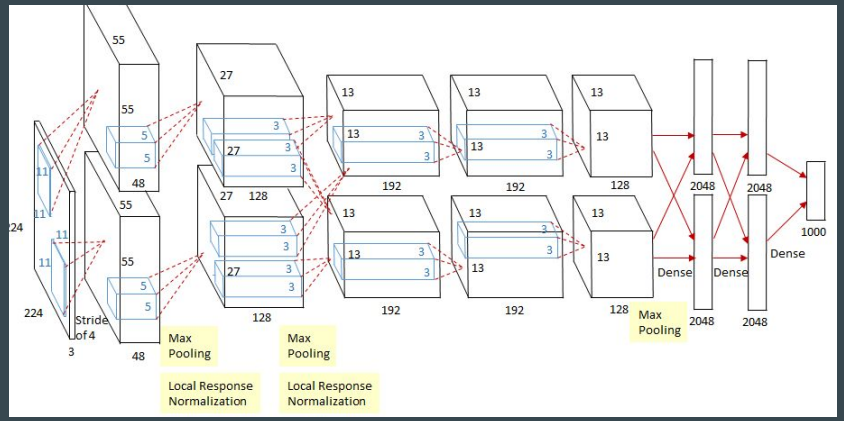
```python
'''LeNet in PyTorch.'''
import torch.nn as nn
import torch.nn.functional as F

class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1   = nn.Linear(16*5*5, 120)
        self.fc2   = nn.Linear(120, 84)
        self.fc3   = nn.Linear(84, 10)

    def forward(self, x):
        out = F.relu(self.conv1(x))
        out = F.max_pool2d(out, 2)
        out = F.relu(self.conv2(out))
        out = F.max_pool2d(out, 2)
        out = out.view(out.size(0), -1)
        out = F.relu(self.fc1(out))
        out = F.relu(self.fc2(out))
        out = self.fc3(out)
        return out
```

AlexNet (2012) -- Background

- Developed by
  - Alex Krizhevsky
  - Ilya Sutskever
    - Chief scientist, OpenAI
    - Inventor of seq2seq learning.
  - Geoffrey Hinton, Alex Krizhevsky's PhD adviser
    - Co-invented Boltzmann machines
- Problem: compete on ImageNet, Fei-Fei Li's dataset of 14 million images with more than 20,000 categories (e.g. strawberry, balloon).

AlexNet (2012) -- Architecture

- Input of 224x224x3 images.
- 8 weighted layers
  - 5 convolutional layers.
    - Filter width 11, then 5, then 3
    - Stride 4
  - A lot of max-pooling layers
    - Kernel width 3
    - Stride 2
  - 3 fully connected layers.
  - ReLU activation until last layer (which was softmax)
  - Trained with dropout and local response normalization



AlexNet (2012) -- Results

- Smoked the competition with 15.3% top-5 error (runner-up had 26.2%)
- One of the first neural nets trained on a GPU with CUDA.
  - (There had been 4 previous contest-winning CNNs)
  - Trained 60 million parameters
- Cited over 30,000 times: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

AlexNet in PyTorch

```
import torchvision.models as models
alexnet = models.alexnet()
```

AlexNet in PyTorch (source code)

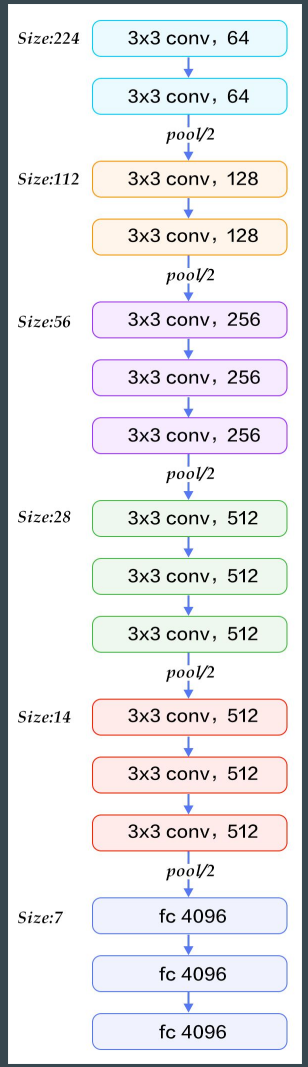
```
class AlexNet(nn.Module):

    def __init__(self, num_classes=1000):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, num_classes),
        )
```

VGG (2014) -- Background

- Developed by the Visual Geometry Group (Oxford)
- Problem: beat AlexNet on ImageNet

VGG (2014) -- Architecture

- The deepest neural net ever.
- 138 million parameters



VGG (2014) -- Results

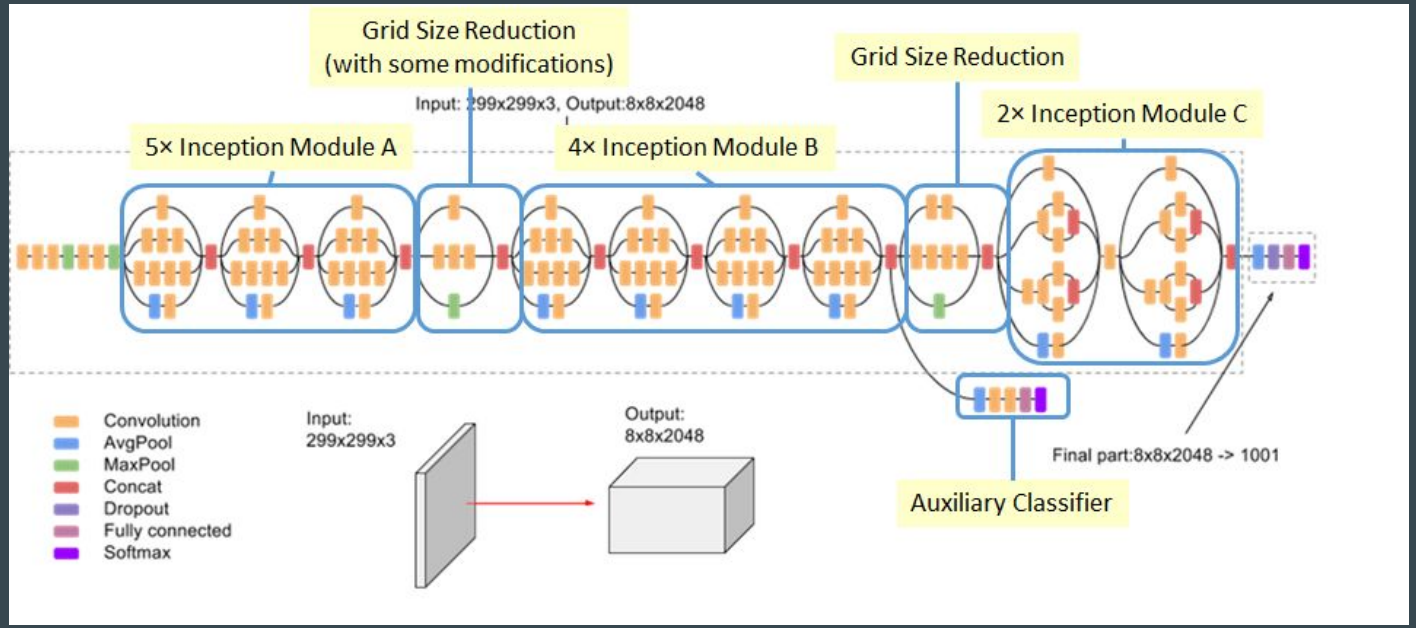
- Configuration E (19 weighted layers) achieved 8.0% top-5 error.
- https://arxiv.org/pdf/1409.1556.pdf

VGG in PyTorch

```
import torchvision.models as models
vgg16 = models.vgg16()
```

No useful source code to speak of; the neural net is so prohibitively large (hence, a pain to train) that the model features are downloaded.

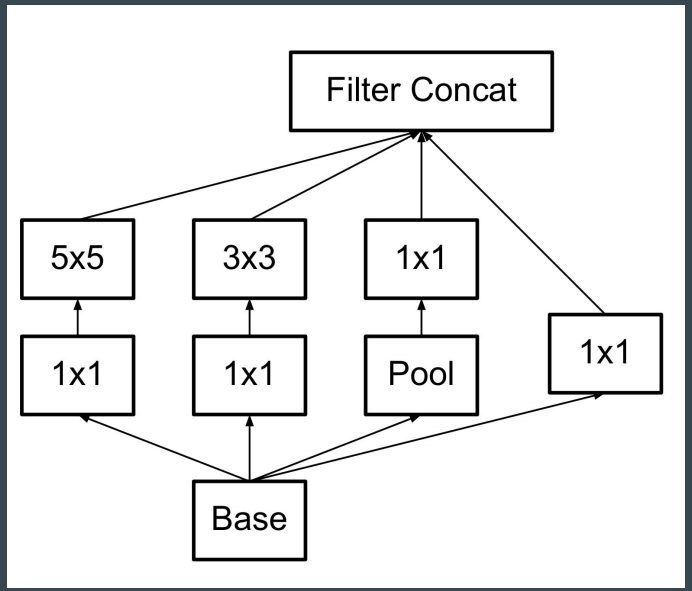
Inception / GoogleNet (2015)



Inception v1: tuning kernel size is hard.

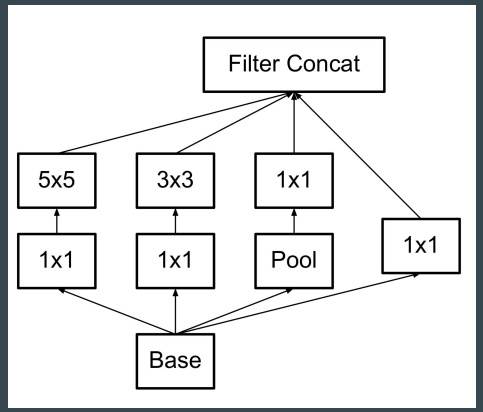
- A computer scientist's solution: toss 'em all together.

A single "Inception Module"

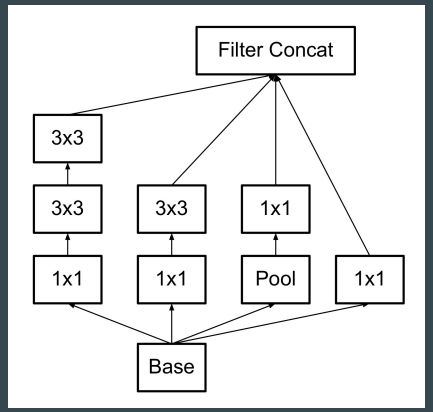


Inception v3: compute, representational bottlenecks

Original



Later version

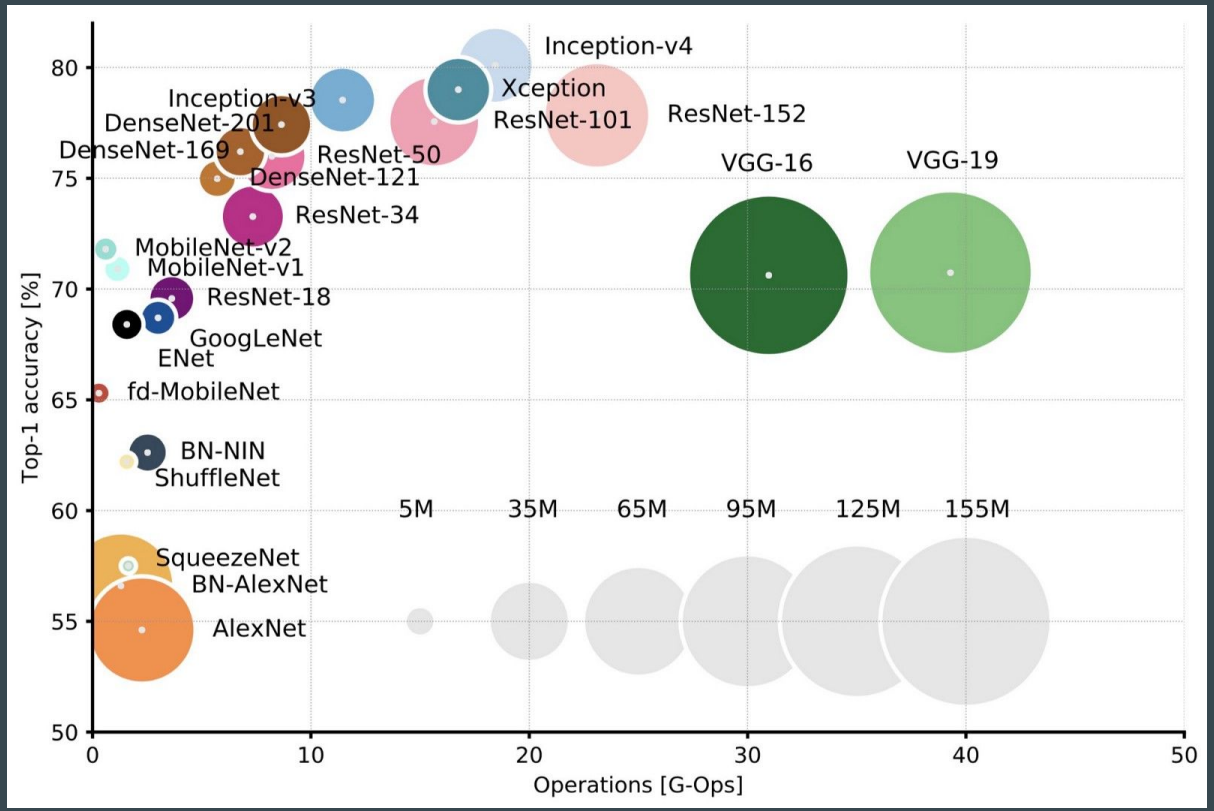


- Representational bottleneck: worse learning properties when the dimensions of the data are drastically changed all at once (more on that next week).

Inception (2014) -- Results

- 6.67% top-5 error rate!
- Andrej Karpathy achieved 5.1% top-5 error rate.

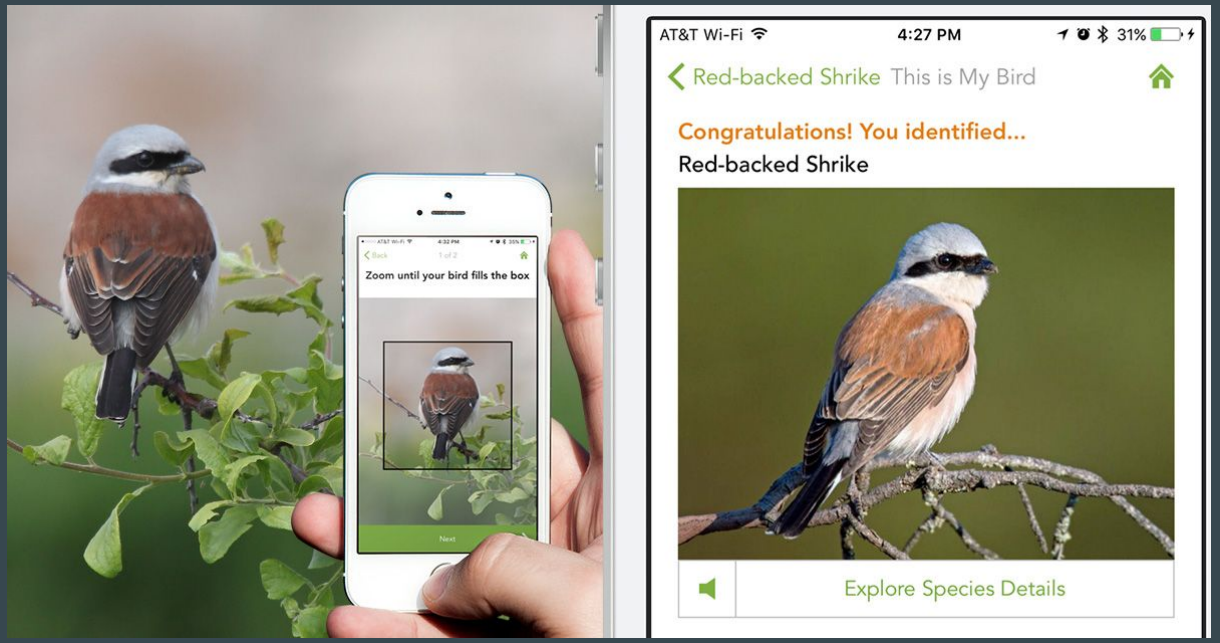
Performance of convolutional architectures on ImageNet



Applications of CNNs

...

Image recognition



Cornell Lab of Ornithology, Merlin bird identification app
(see Van Horn et al. 2015)

Style transfer

- Content is measured by deeper layers.
- Style is measured by the correlations between feature vectors at lower layers.

- Objective 1: Minimize content difference between new image and content template.
- Objective 2: Minimize style difference between new image and style template.

Gatys et al. (2016)

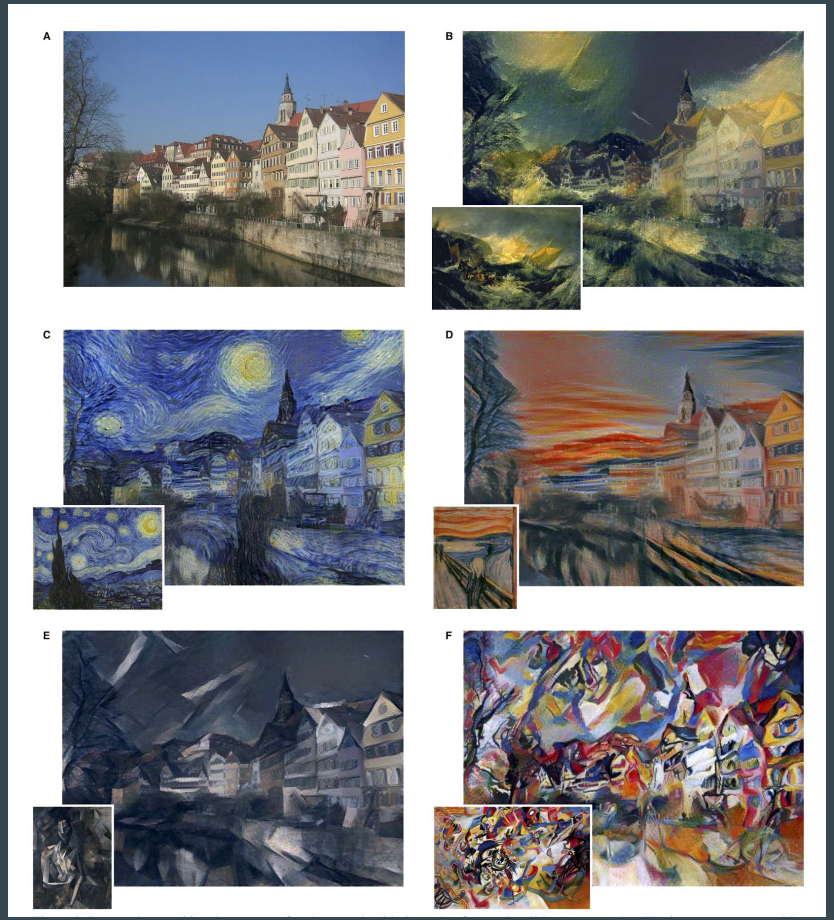
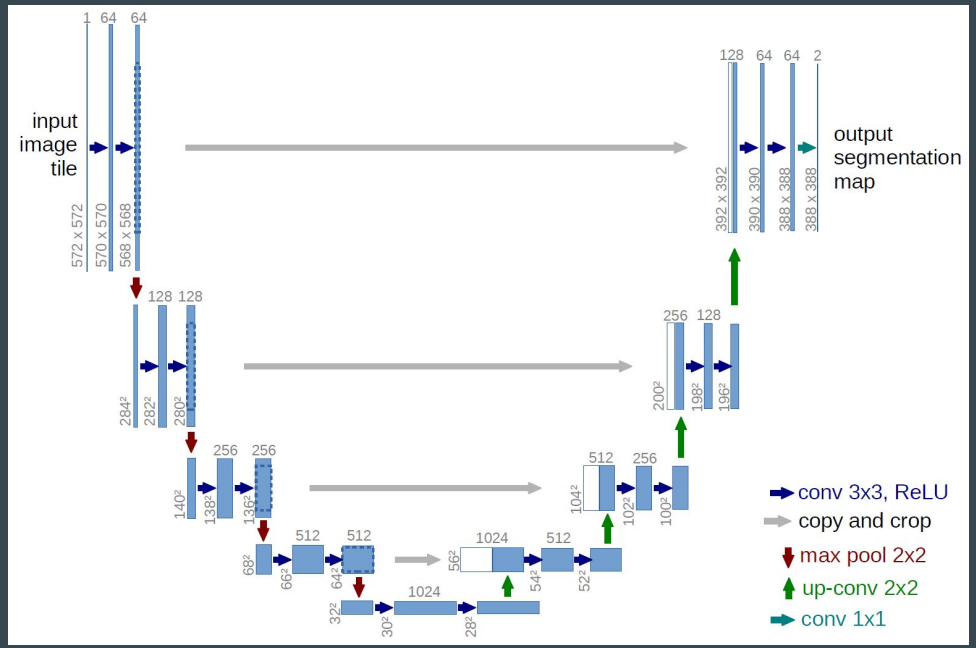
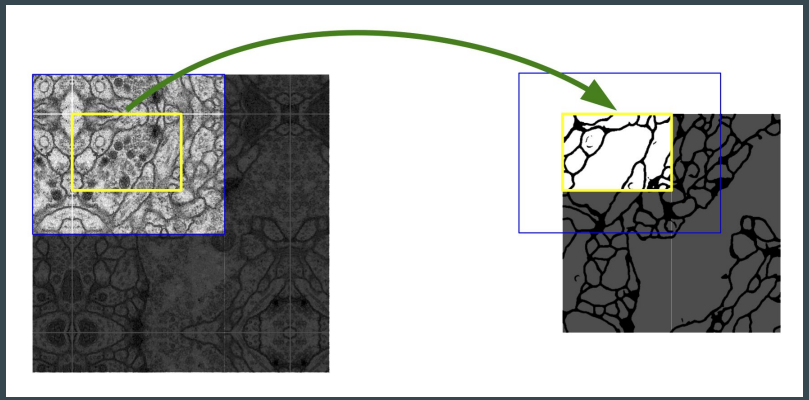


Image segmentation

U-Net (Ronneberger et al. 2015)



Segmenting biological cell membranes

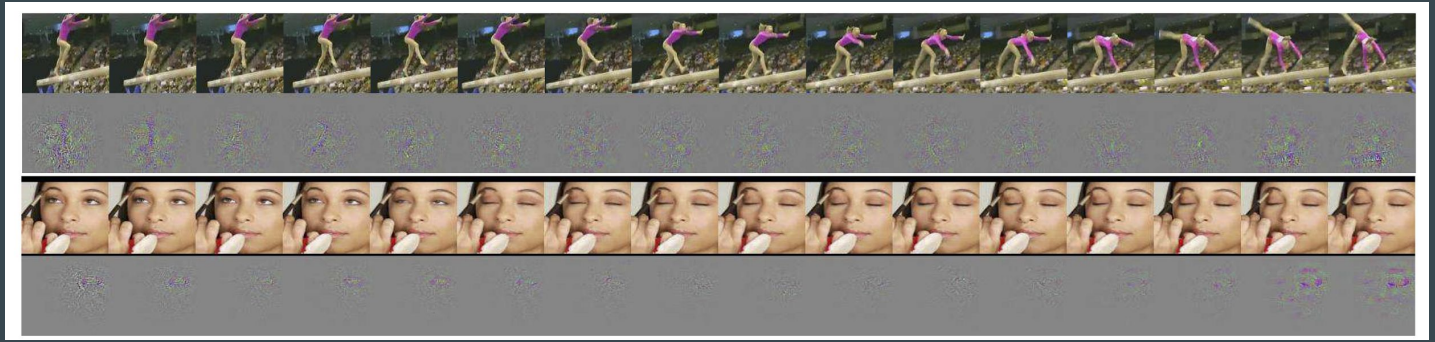


Insights of U-Net

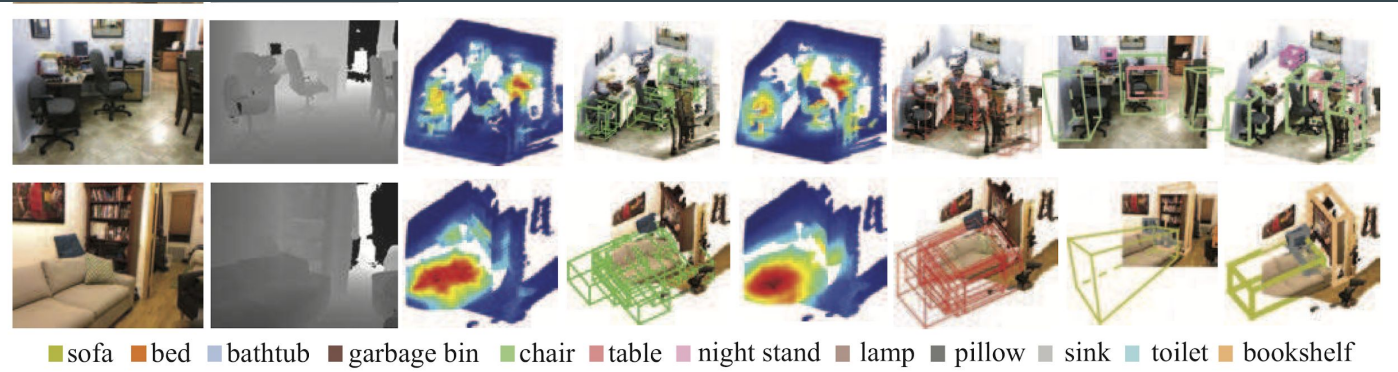
- “Up-convolution”
  - Fixes the problem of shrinking images with CNNs
  - One example of how to make “fully convolutional” nets: pixels to pixels
  - “Up-convolution” is just upsampling, then convolution
  - Allows for refinement of the upsample by learned weights
  - Goes along with decreasing the number of feature channels
  - Not the same as “de-convolution”.
- Connections across the “U” in the architecture

3D convolution

Motion in videos - Tran et al. (2015)

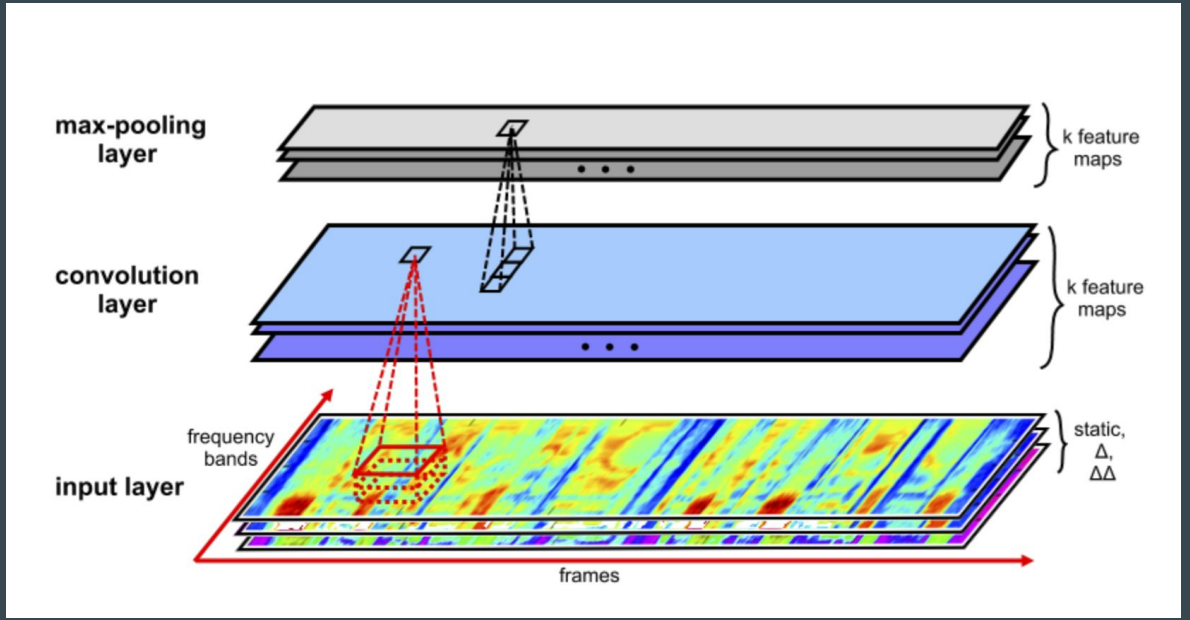


Objects from images w/ depths - Song & Xiao (2016)



Speech recognition

Zhang et al. 2017



- CNN competitive with RNNs (e.g. LSTMs)
- 10 layers, 3x5 conv, 3x1 pooling - deep enough for temporal dependencies
- TIMIT task, classifying phonemes

Graph convolution

- For a full intro: https://tkipf.github.io/graph-convolutional-networks/
- Basic idea:
  - A convolution for matrices is a “local” combination of entries in the matrix
  - Can do the same thing for a graph
  - Here, “local” is whatever nodes a node is adjacent to
- Just the tip of an iceberg…
- Used in learning stuff about molecules, citations, social networks, etc.

Myth: CNNs are for computer vision, and RNNs are for NLP.

CNNs relatively recently were SOTA in many NLP tasks

3. Text Classification

Research Paper	Datasets	Metric	Source Code	Year
Learning Structured Text Representations	Yelp	Accuracy: 68.6	• Tensorflow	2017
Attentive Convolution	Yelp	Accuracy: 67.36	• Theano	2017

https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/4%20-%20Convolutional%20Sentiment%20Analysis.ipynb

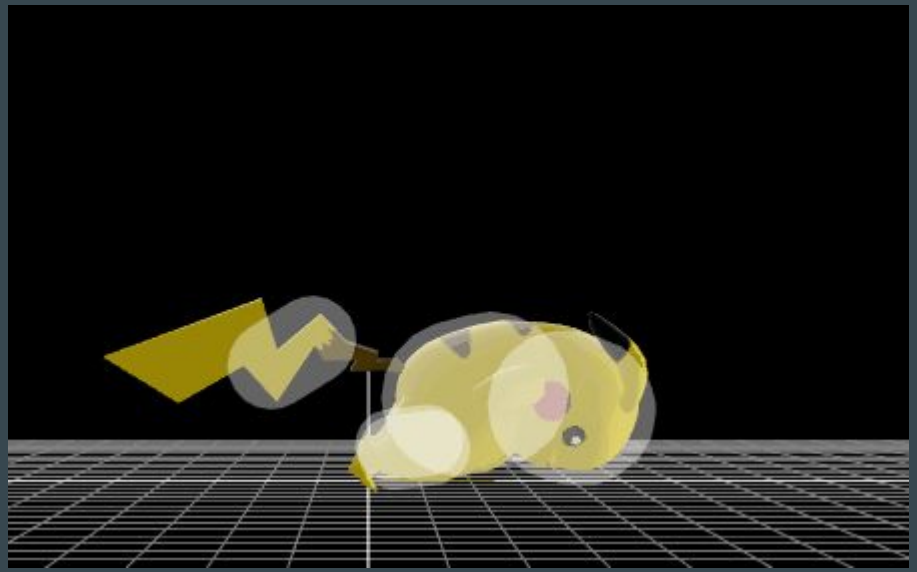
Convolutional neural networks can be a mess.

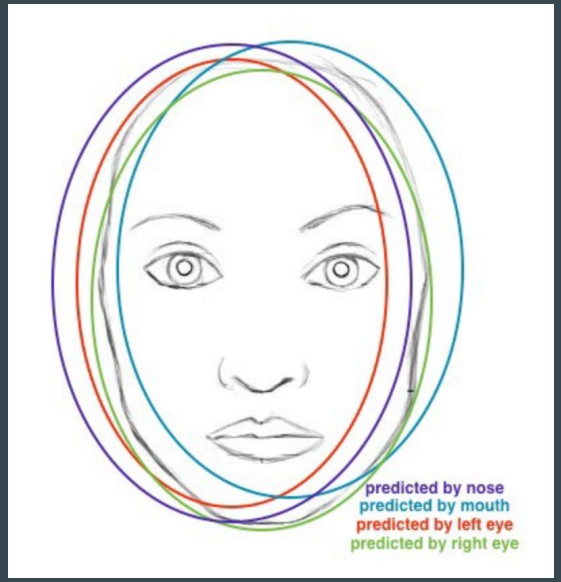
1. Solves the wrong problem: we want equivariance, not invariance.
2. Bad fit to the psychology of shape perception.
3. Does not use underlying linear structure of the universe.
4. Fails to route information intelligently: max pooling sucks.

1. Translational invariance is the wrong problem

The Picasso problem: convolutional neural networks detect parts with no sense of a whole.

2, 3. Bad fit to human perception and structure of motion





https://youtu.be/rTawFwUvnLE?t=835
https://youtu.be/rTawFwUvnLE?t=1195

4. Pooling sucks

- Does not account for the offset in its computation.
- Ensure that information about localization is **erased.**
- Certainly doesn't intelligently send information to the appropriate neurons.
    - E.g. if my data has (age, weight, sex) and my model has a neurons abstracting (sex, age, weight), there should ideally be a way to route the information to the right place.

“The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster.”
-- Geoffrey Hinton

Given all these problems with CNNs, what might we hope for from a superior architecture?

Our wishlist for a new architecture

- Awesome priors
  - **Ontology**: the world is made up of objects that have properties.
  - **Inverse-graphics**: objects move linearly in space (translation) and rotate.
- Information is properly routed to the appropriate neurons.
  - Routes by "agreement" rather than by "volume."
- Super interpretable
  - Clear representation of learned features
  - Visualization of internal representation
- Learns with very few examples (fewer than 5 per class?)
- Outperforms CNNs in accuracy
- Runs blazingly fast

Our wishlist for a new architecture

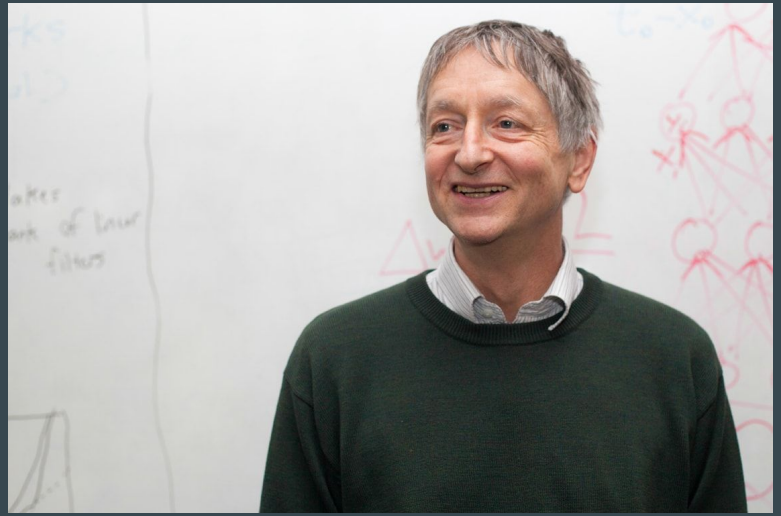
- Awesome priors
  - the world is made up of objects that have properties.
  - objects move linearly in space (translation) and rotate.
- Information is properly routed to the appropriate neurons.
  - Routes by "agreement" rather than by "volume."
- Super interpretable
  - Clear representation of learned features
  - Visualization of internal representation
- Learns with very few examples (fewer than 5 per class?)
- Outperforms CNNs in accuracy
- ~~Runs blazingly fast~~

Today, many of these goals are met by various transformer architectures

Coming your way in a few weeks

Geoffrey Hinton

- English-Canadian cognitive psychologist and computer scientist
- Popularized backpropagation
- The "Godfather of Deep Learning"
- Co-invented Boltzmann machines
- Contributed to AlexNet
- Advised Yann LeCunn, Ilya Sutskever, Radford Neal, Brendan Frey
- Created many of the networks we enjoy these days



Convolution thinking

What is convolution?

Why do we use it?

What would you do if every pixel was different?

What would you do if they were randomly shuffled?

What would you do if there was no spatial organization?

Continued

What would you do if the data had underlying graph structure? E.g. humans in a social network?

What would you do if you had crazy input sensors, say lasers pointing into random directions?