Machine Learning - Deep Learning fundamentals
(69152) **CNNs**

Master in Robotics, Graphics and Computer Vision

Ana C. Murillo

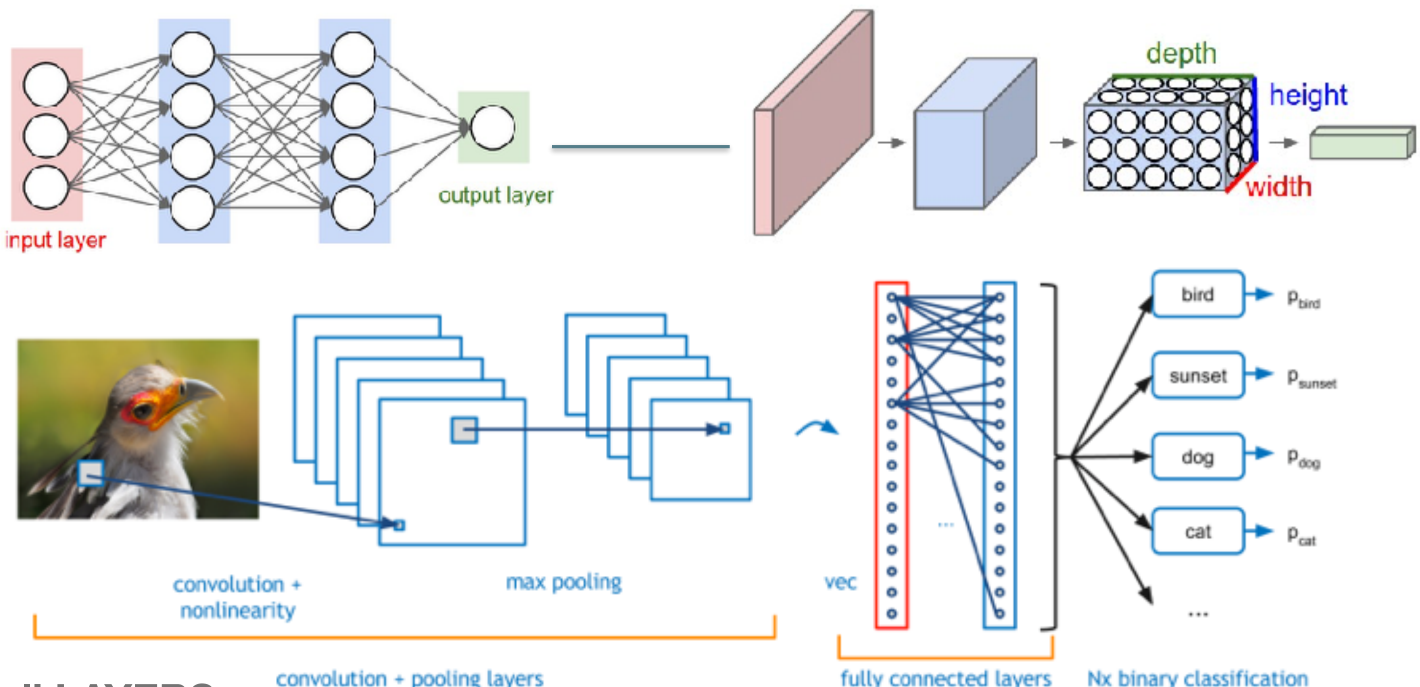
Today

- What’s a CNN?
- Well-known CNN arquitectures

CNNs

- Convolutional (deep) Neural Networks (CNN)

[DEMO online, CNN]()



NOT all LAYERS are the same

Fei-Fei, Karpathy, Johnson. Convolutional Neural Networks for Visual Recognition (http://cs231n.stanford.edu)
Evan Shelhamer, Jeff Donahue, Jon Long, Yangqing Jia, and Ross Girshick. *Deep Learning for Vision:* a Hands-On Tutorial

CNNs - Basic Layers

- **Input**: raw pixel values. RGB image 32x32, volume [32x32x3]

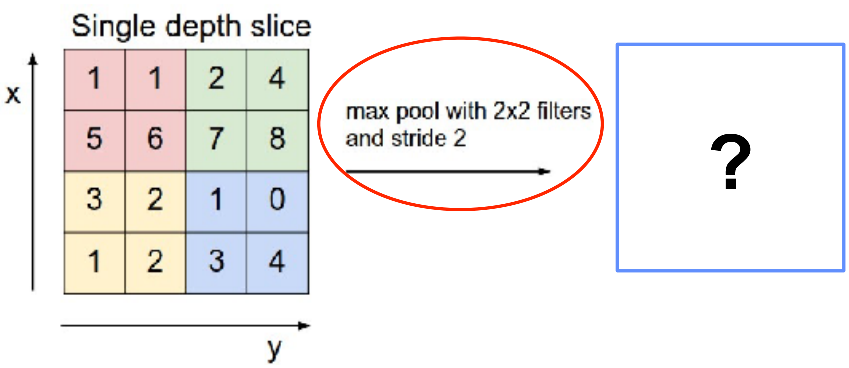
- **Convolutional**: compute output of neurons connected to local input regions (each computes dot product between their weights and the region). Larger volume [32x32x12].

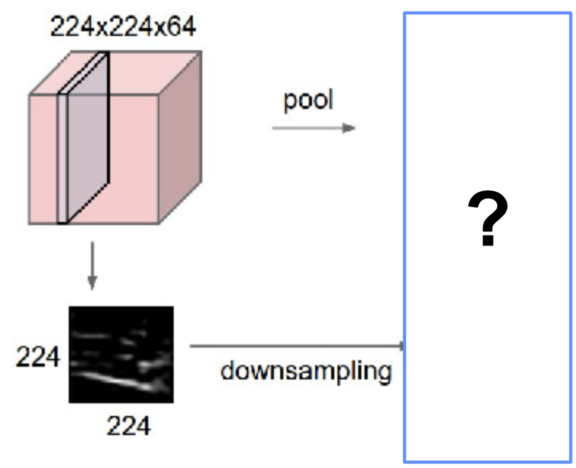
- **RELU**: element-wise activation function (e.g. thresholding at zero). Volume unchanged ([32x32x12]).

- **Pooling**: downsampling operation. e.g. reduce volume to [16x16x12].

- **Fully-connected**: compute class scores. Each neuron in this layer will be connected to all the numbers in the previous volume. Volume of size [1x1x10].

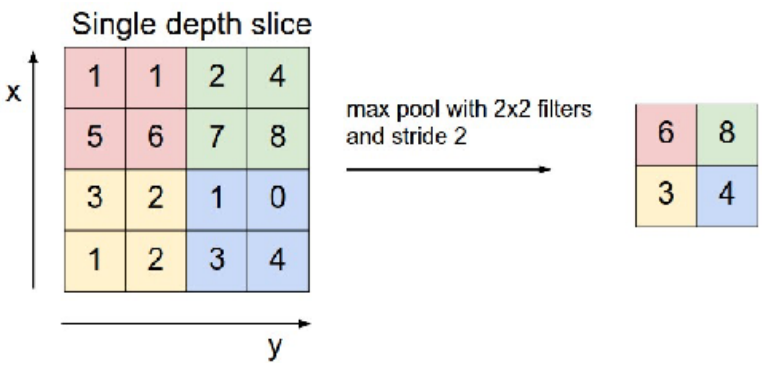
CNNs - pooling example

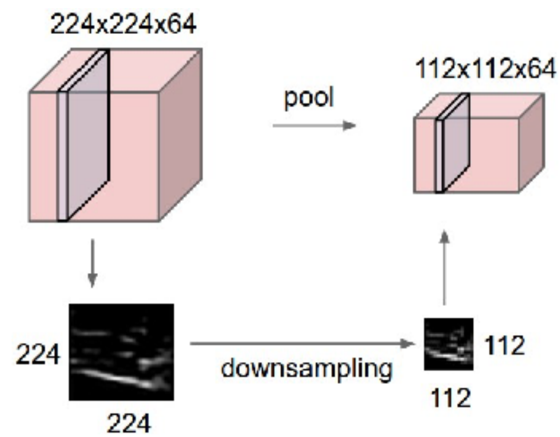




Fei-Fei, Karpathy, Johnson. Convolutional Neural Networks for Visual Recognition (http://cs231n.stanford.edu)
Evan Shelhamer, Jeff Donahue, Jon Long, Yangqing Jia, and Ross Girshick. *Deep Learning for Vision:* a Hands-On Tutorial

CNNs - pooling example





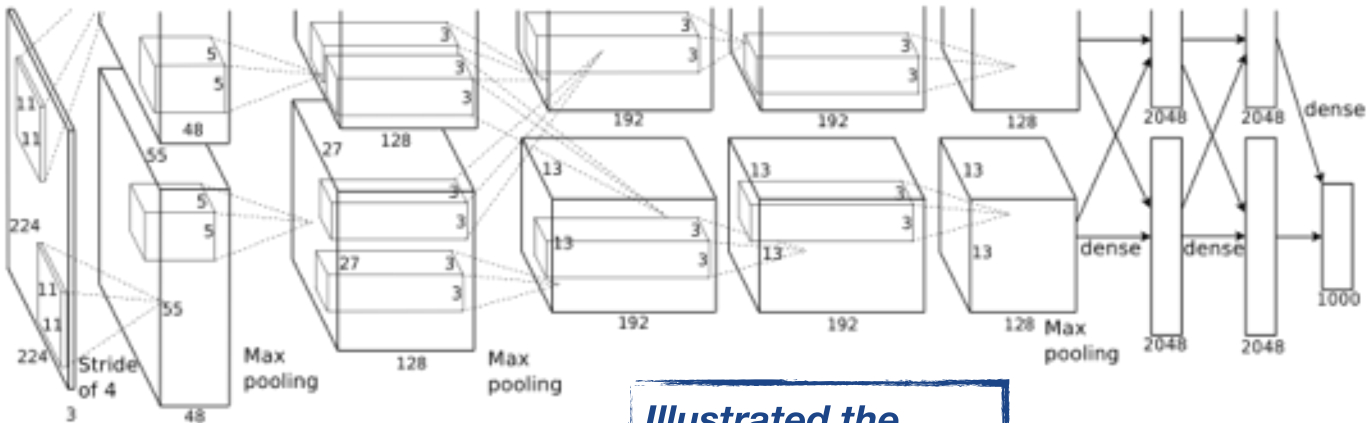
Fei-Fei, Karpathy, Johnson. Convolutional Neural Networks for Visual Recognition (http://cs231n.stanford.edu)
Evan Shelhamer, Jeff Donahue, Jon Long, Yangqing Jia, and Ross Girshick. *Deep Learning for Vision:* a Hands-On Tutorial

CNNs

- A bit of “history” about CNNs

CNNs - Image Classification

CNNs *basics*: AlexNet (Beginning of the *new* CNN *boom):*
a layered model composed of convolution, subsampling, and further operations followed by a holistic representation.



Illustrated the benefits of CNNs

- 15 million annotated images
- over 22000 classes
- **ReLU, data augmentation, dropout**

IMAGENET

ImageNet Classification with Deep Convolutional Neural Networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. NIPS **2012**

CNNs

CNNs *basics*: DeConvNet

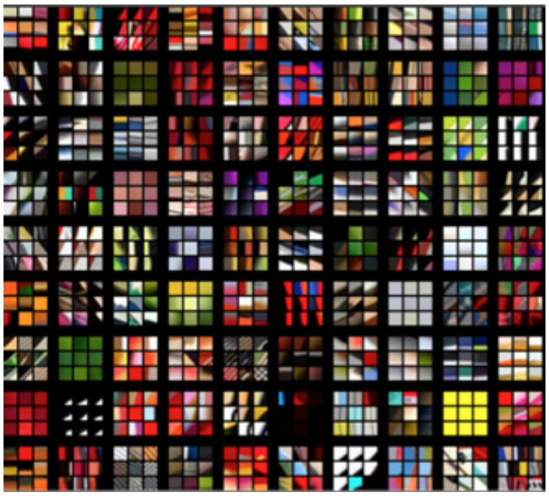
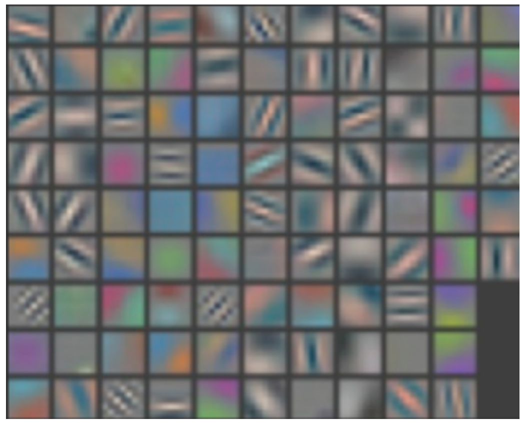
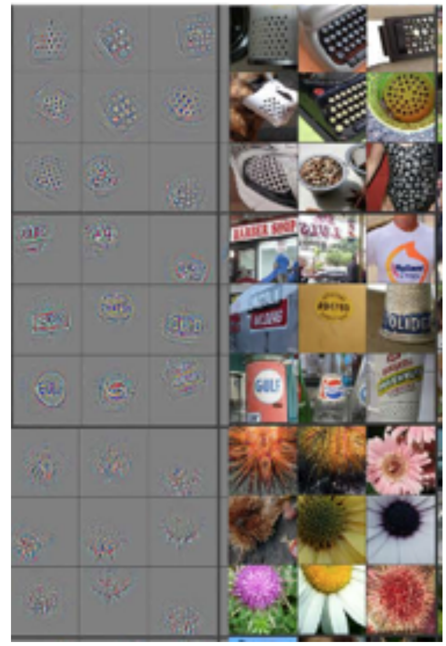


image patches that strongly activate 1st layer filters



1st layer filters

better understanding: earlier vs later layers



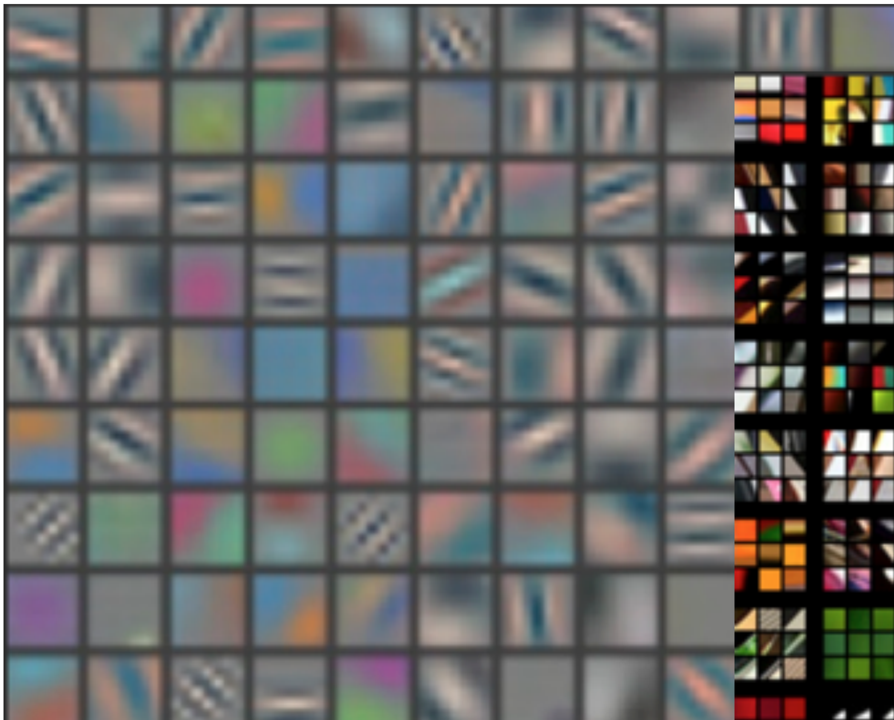
conv_5 DeConv visualization
[Zeiler-Fergus]

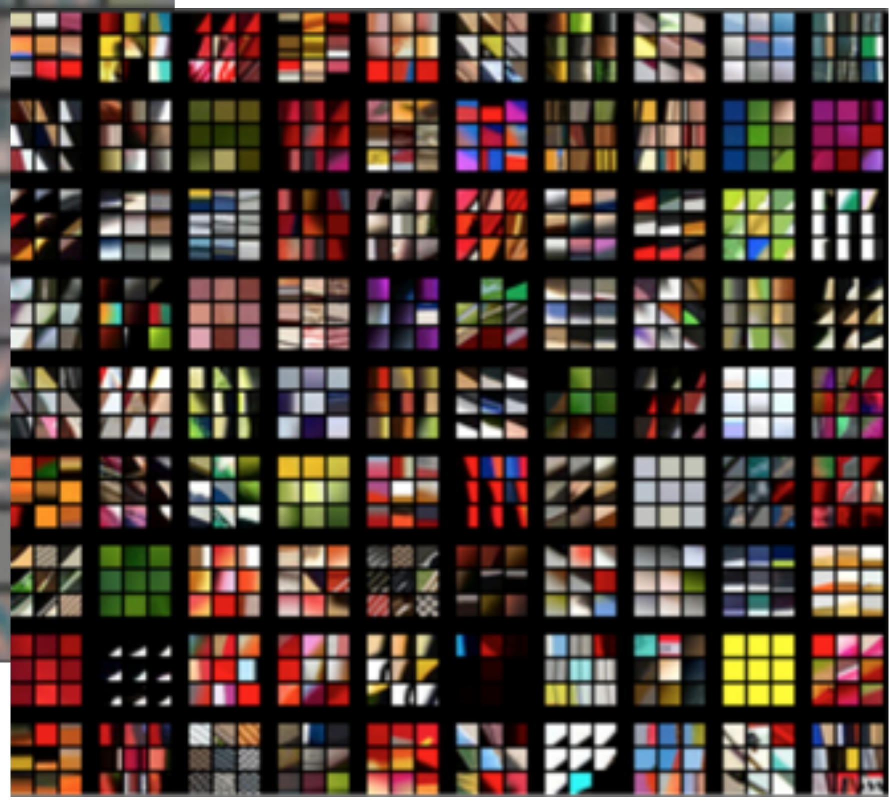
- Similar architecture to AlexNet
- less training data - smaller filters
- Deconvolutional Network - **visualization**

Visualizing and Understanding Convolutional Neural Networks
Matthew Zeiler and Rob Fergus. **2013**

http://people.csail.mit.edu/torralba/research/drawCNN/drawNet.html

CNNs *basics*: DeConvNet

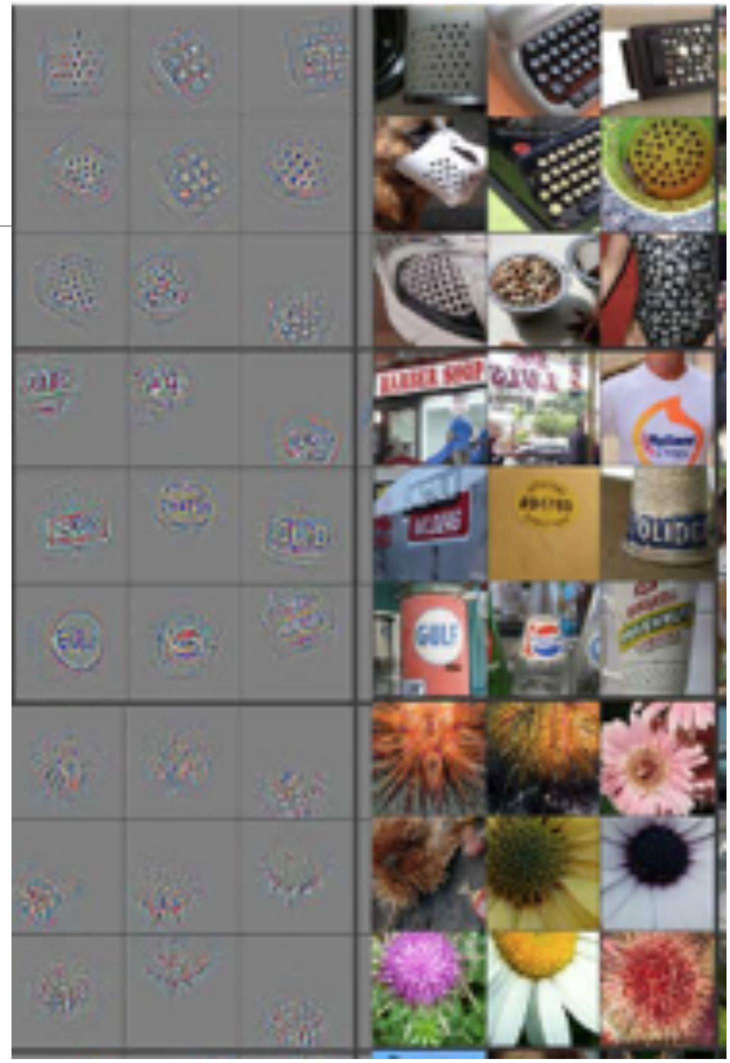




1st layer filters

Visualizing and Understanding Convolutional Neural Networks
Matthew Zeiler and Rob Fergus. **2013**

CNNs *basics*: DeConvNet



top 9 activations for a few feature maps
(projection to pixel space using DeConv
and actual image patches)

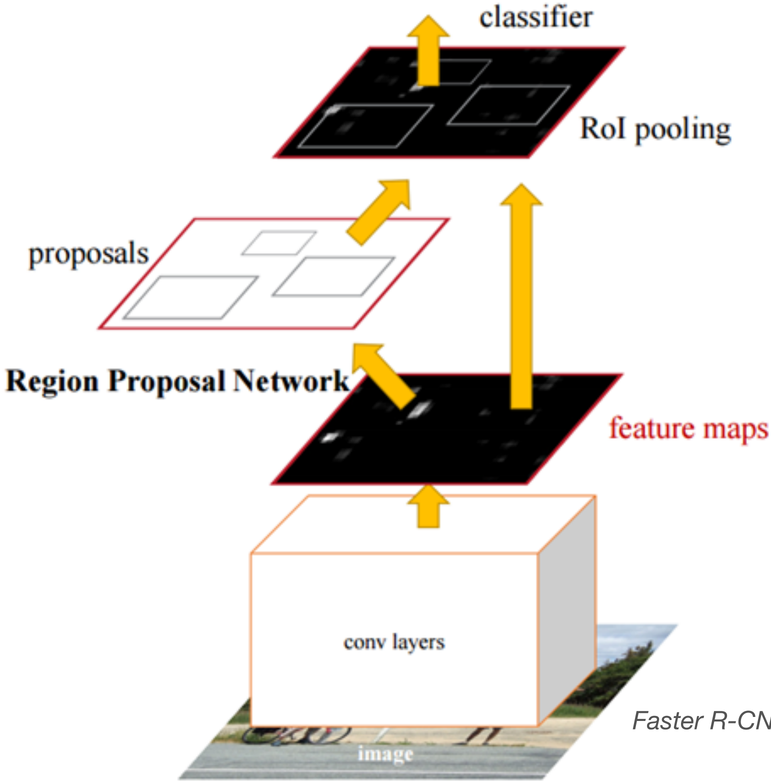
conv$_5$ DeConv visualization
[Zeiler-Fergus]

Visualizing and Understanding Convolutional Neural Networks
Matthew Zeiler and Rob Fergus. **2013**

CNNs - Image Classification + Detection

- **Analyze the *feature maps*: Region - CNN**

(R-CNN - 2013, Fast R-CNN - 2015, Faster R-CNN - 2016)



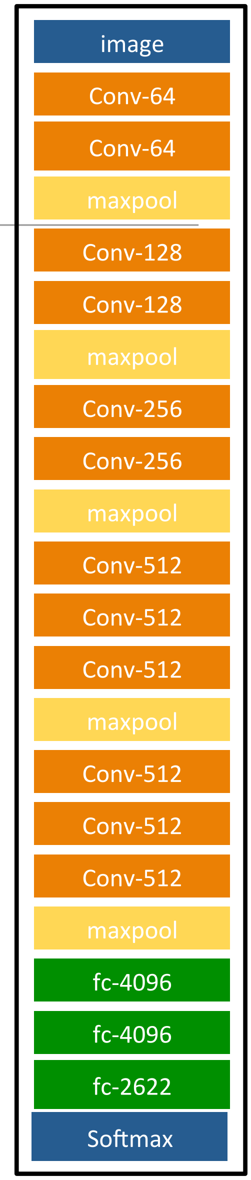
Analyze feature maps (activations) to learn where the objects are

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
Shaoqing Ren, Kaiming He, **Ross Girshick**, and Jian Sun. 2016

CNNs - Image classification

- *Deeper* classification: VGG - ***very deep*** ConvNets
  - Smaller filters - less params
  - more depth! —> always good (?)
  - Consecutive conv. layers (smaller filters).
  - Compare to AlexNet 5 conv. layers

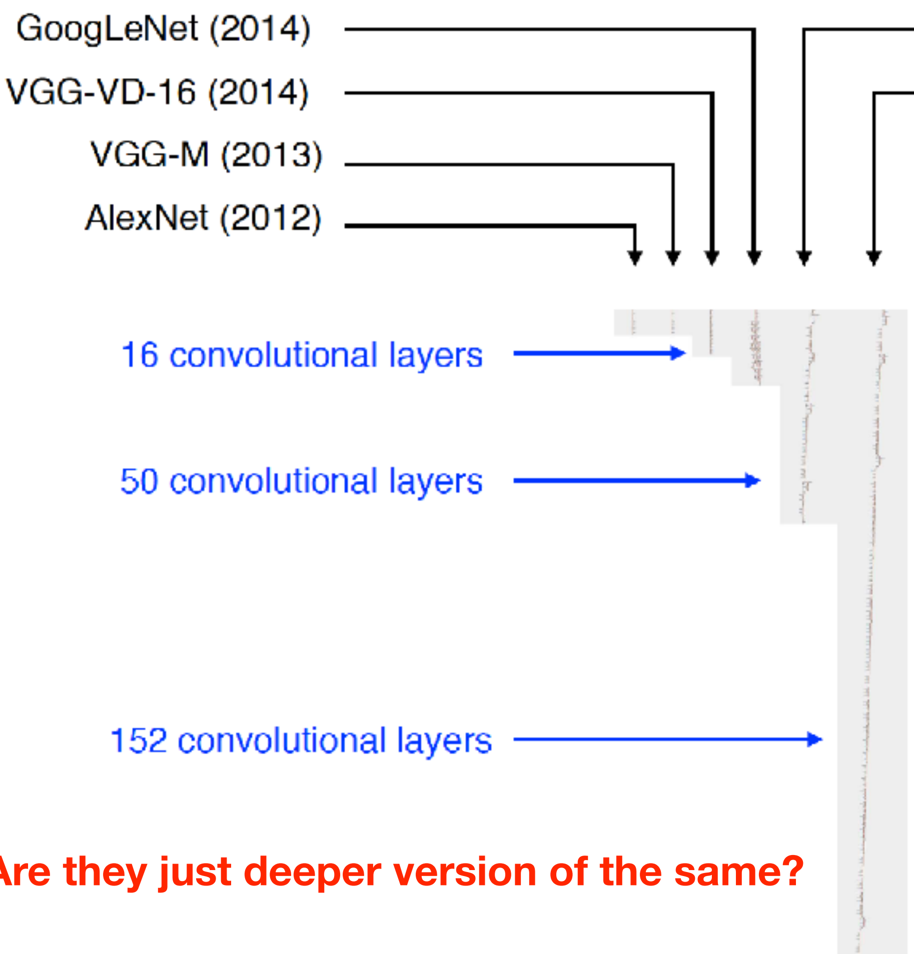
How deep?



Very Deep Convolutional Networks for Large-Scale Image Recognition. K. Simonyan, A. Zisserman. ***2014***
Deep Face Recognition: O. M. Parkhi, A. Vedaldi, A. Zisserman. BMVC. **2015**

slide from R. Zemel. DLSS. 2017

Image classification (deeper)



Are they just deeper version of the same?

Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet classification with deep convolutional neural networks*. In Proc. NIPS, 2012.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proc. CVPR, 2015.

K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In Proc. ICLR, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In Proc. CVPR, 2016.

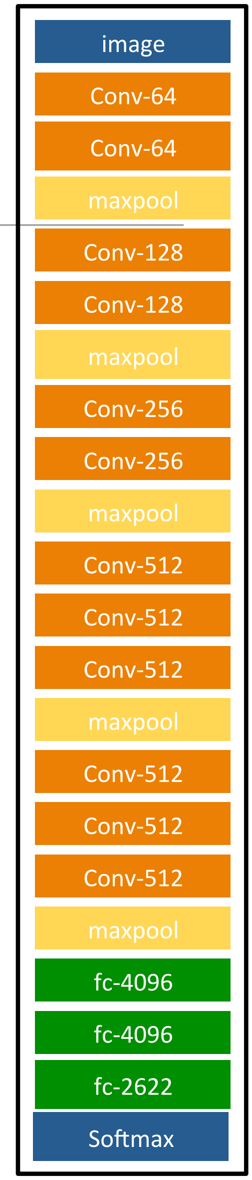
slide from R. Zemel. DLSS. 2017

Deeper: How does memory get affected?

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters



Very Deep Convolutional Networks for Large-Scale Image Recognition.
K. Simonyan, A. Zisserman. 2014

Deeper: How does memory get affected?

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Note:

Most memory is in early CONV

Most params are in late FC

slide from
Fei-Fei Li & Justin Johnson & Serena Yeung
http://cs231n.stanford.edu

What is more relevant for batch size design decisions?

Very Deep Convolutional Networks for Large-Scale Image Recognition.
K. Simonyan, A. Zisserman. 2014

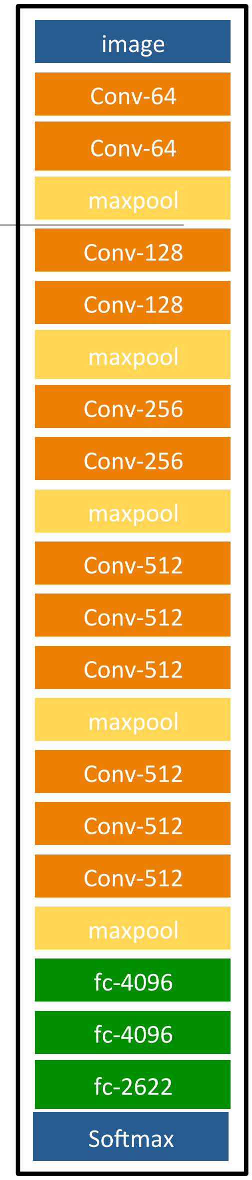
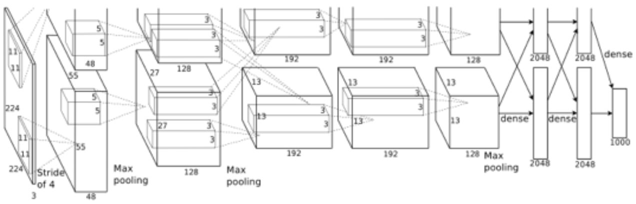
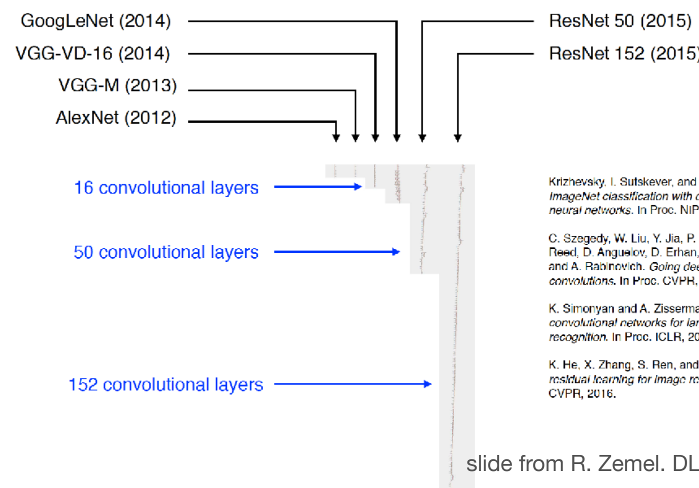


Image classification (deeper)

- Not only more layers! **it's getting hard(er) to train …**



ImageNet Classification with Deep Convolutional Neural Networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. NIPS 2012



Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet classification with deep convolutional neural networks*. In Proc. NIPS, 2012.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proc. CVPR, 2015.

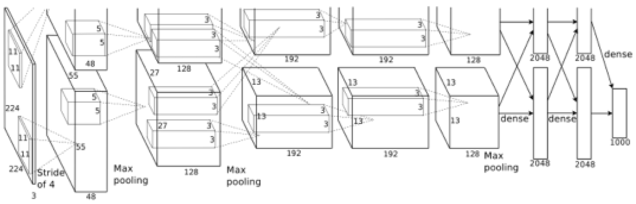
K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In Proc. ICLR, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In Proc. CVPR, 2016.

slide from R. Zemel. DLSS. 2017

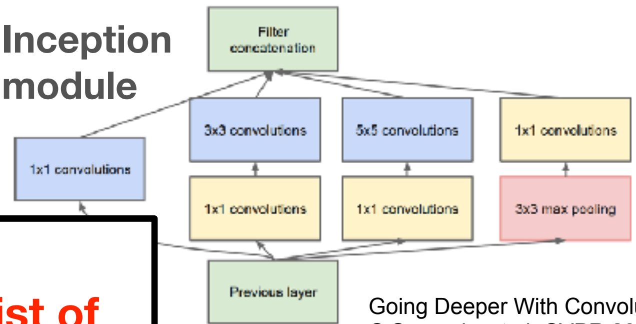
Some *reference modules* for CNN architectures …

- to train better: GoogLeNet, ResNet, DenseNet, …



ImageNet Classification with Deep Convolutional Neural Ne…
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. NIPS 2…

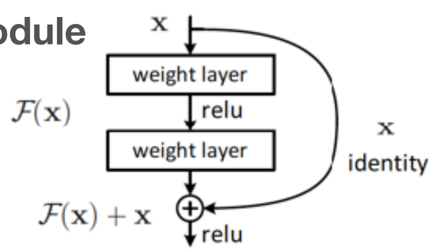
Inception module

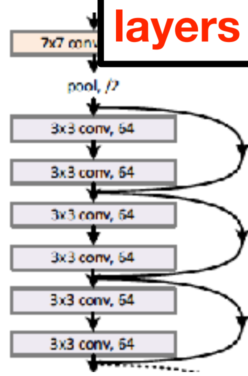


Going Deeper With Convolutions.
C.Szegedy, at al. CVPR 2015.

Not just “linear” list of layers anymore

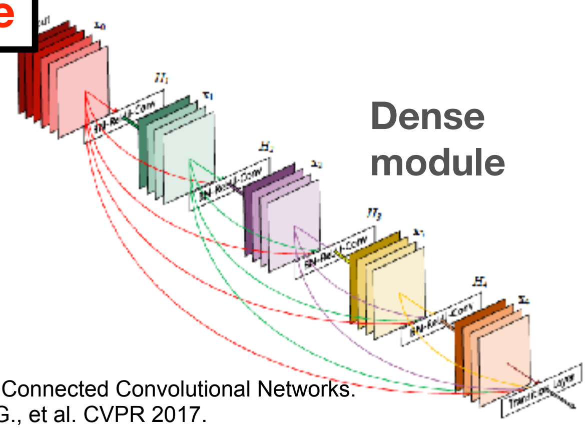
Residual module





Deep residual learning for image recognition.
He, Kaiming, et al. CVPR 2016

Dense module



Densely Connected Convolutional Networks.
Huang, G., et al. CVPR 2017.

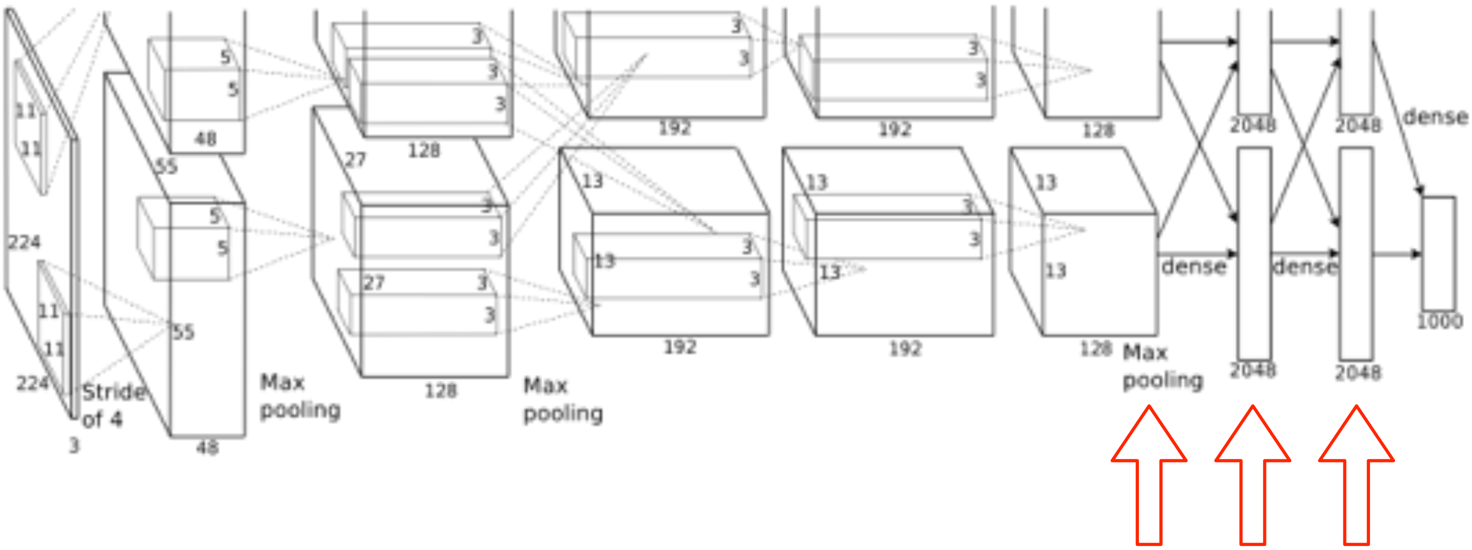
Open or interesting problems related to deep learning?

Not enough data to train? (or resources)

CNNs & Transfer Learning

- CNNs are able to generalize well!
  - great **features**
  - **fine-tuning**

Lab 2
you'll practice
some of this



ImageNet Classification with Deep Convolutional Neural Networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. NIPS 2012

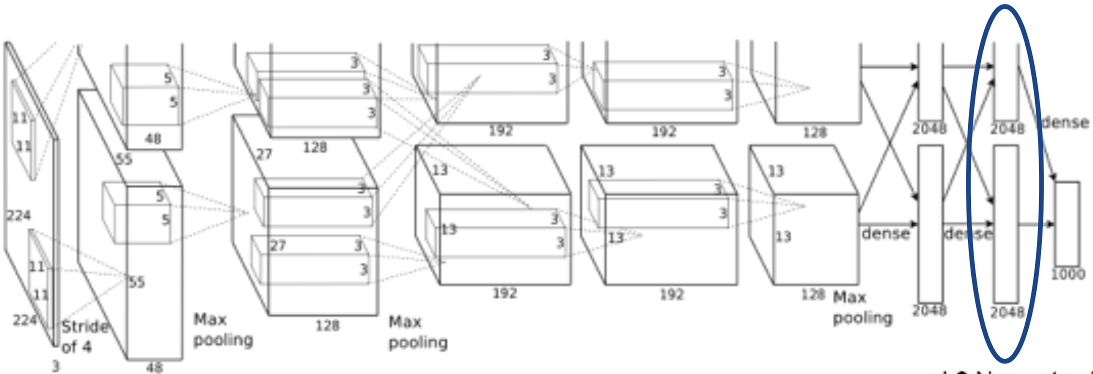
deep features

Transfer Learning

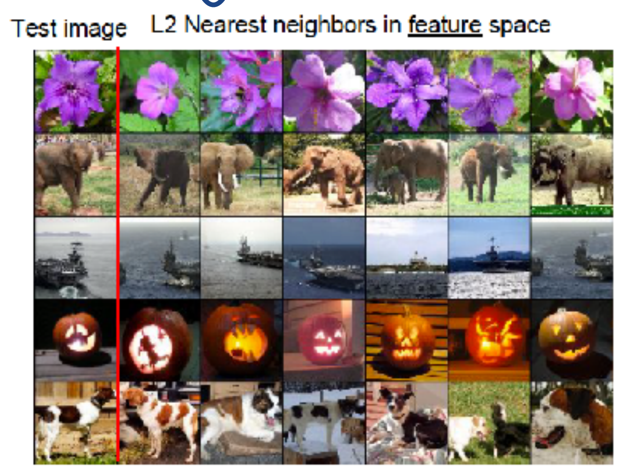
- What is it?
- When would you use transfer learning?
- Types of transfer learning?

Transfer Learning: features

- E.g., features from AlexNet last layer: 4096 dims. vector



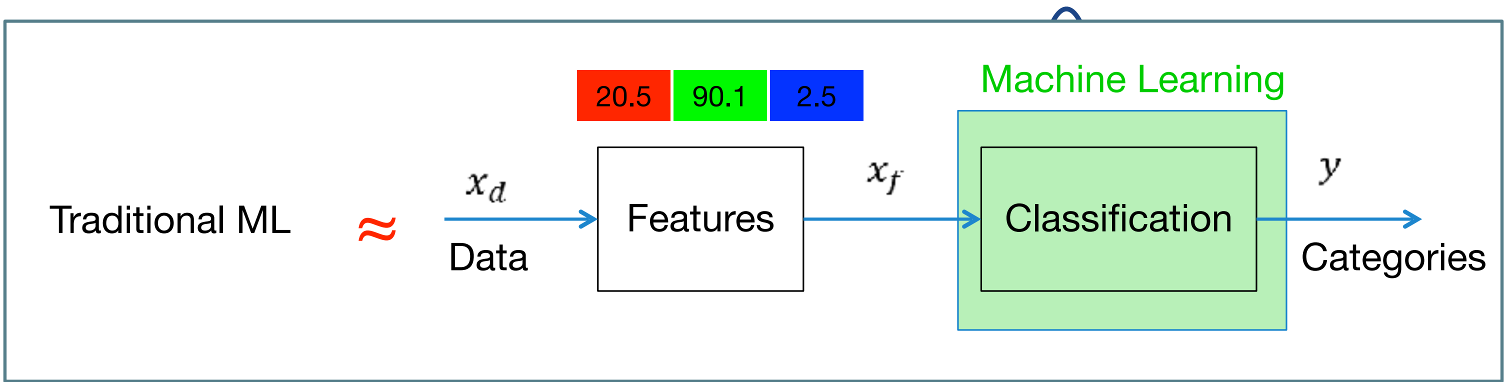
How do we get these features?



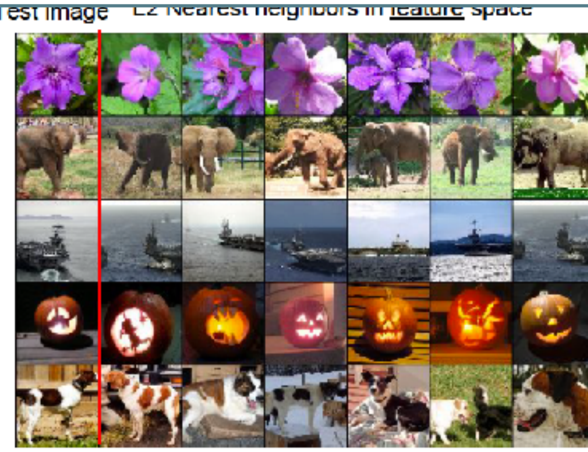
ImageNet Classification with Deep Convolutional Neural Networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. NIPS 2012

Transfer Learning: features

- E.g., features from AlexNet last layer: 4096 dims. vector



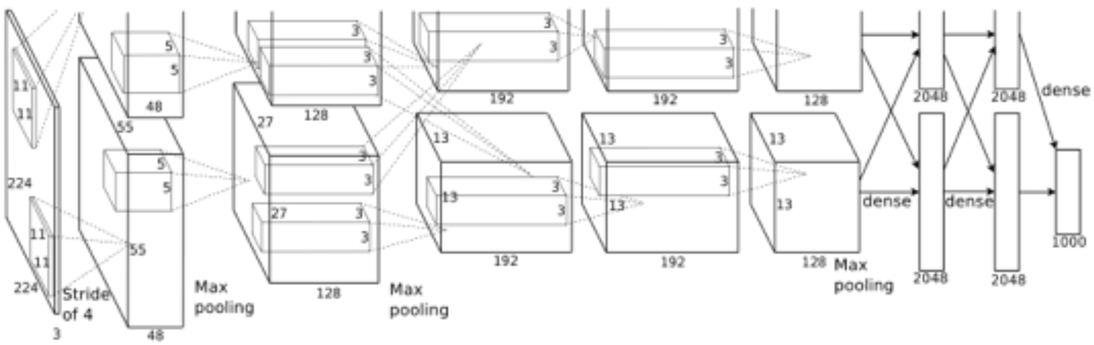
How do we use these features?



ImageNet Classification with Deep Convolutional Neural Networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. NIPS 2012

Transfer Learning: fine-tune

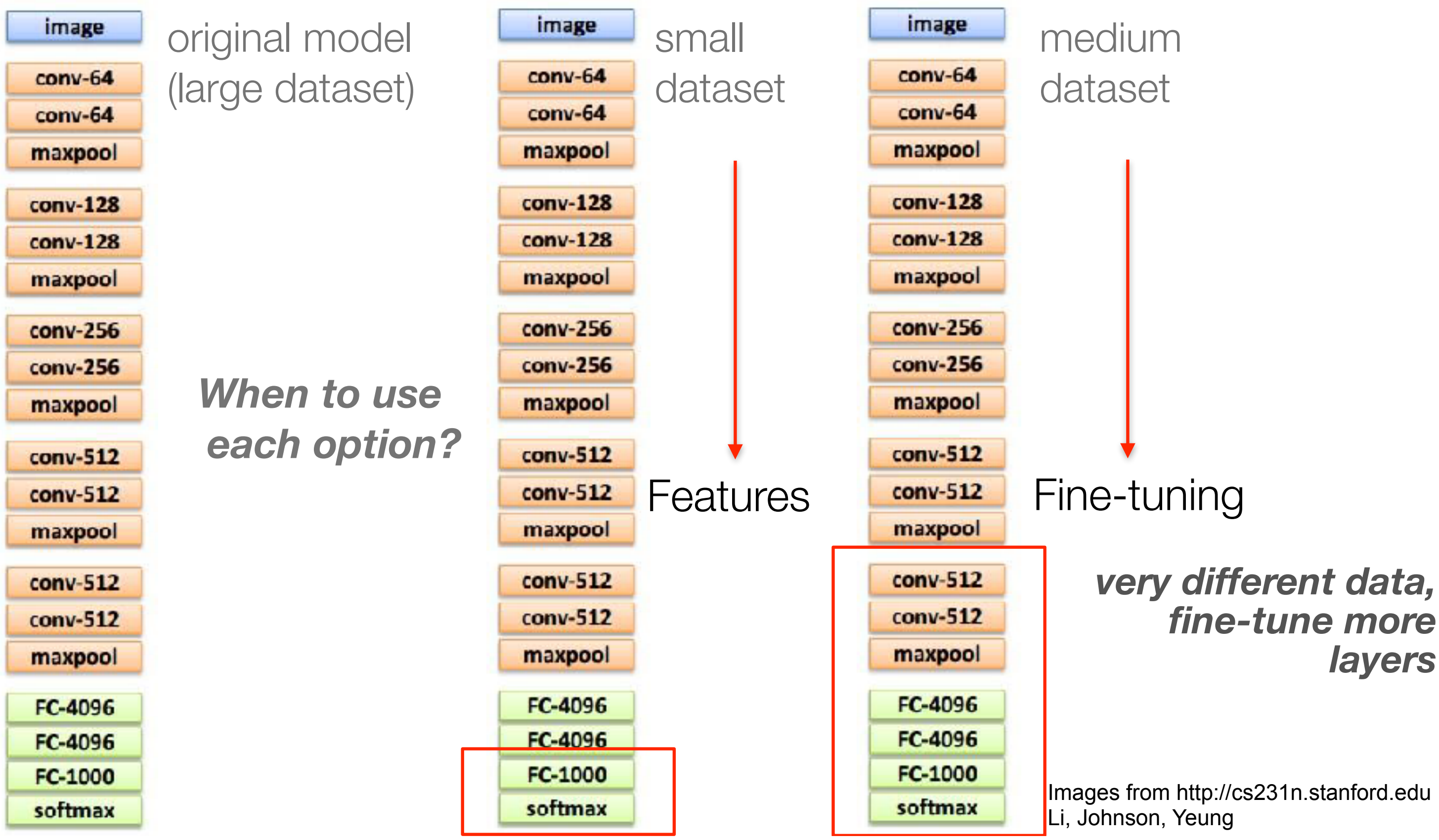
- For example, fine-tune ImageNet AlexNet for non Image-Net classes
- Basically, initialise weighs to something more “*interesting*” than random (careful also with hyperparameters!)



ImageNet Classification with Deep Convolutional Neural Networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. NIPS 2012

Transfer Learning

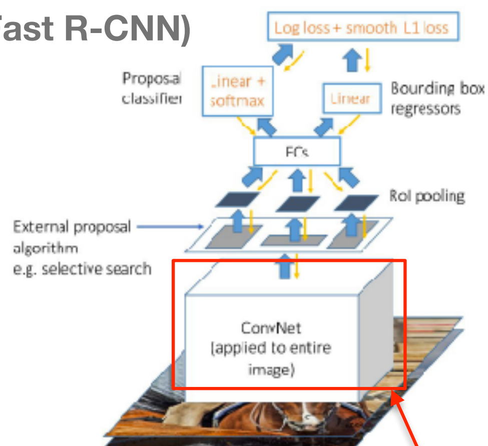
… when not enough resources to train from scratch



Transfer Learning: fine-tuning

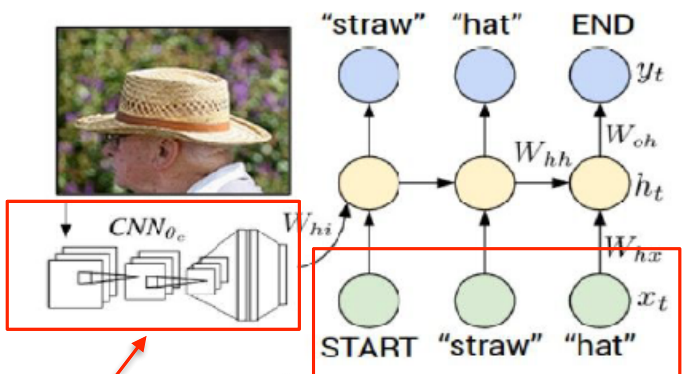
- not just that! very very spread out strategy:

Object Detection (Fast R-CNN)



Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Image captioning: CNN + RNN



Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

Pre-trained models to initialize certain blocks

Images from http://cs231n.stanford.edu . Li, Johnson, Yeung

Lab 2: NN and CNNs applied in Computer Vision

- Understand a **DNN implemented from scratch**
- Implement a toy CNN with Keras
- Fine-tune a well-known CNN architecture with Keras

All this is applied to Image Classification

- Optional tasks:
  - implement variations of the classification networks
  - explore object recognition model
  - explore semantic segmentation model

Available in Moodle.

Please check the prior-work task (prepare your set up and data)

TO-DO …

Have you all used COLAB?

- Lab 2 - **TOMORROW (18 OCT , 20 OCT) —> WED. @ A07 / FRIDAY @ L0.06a ADA BYRON**
- PLEASE **have your computer ready with Tensorflow2+Keras (ideally with GPU available) AND/OR we will use Google COLAB**
- Recommended COLAB tutorial if you have not used it much before:
  https://colab.research.google.com/notebooks/intro.ipynb
  https://colab.research.google.com/notebooks/basic_features_overview.ipynb
  Loading data: Drive, Sheets, and Google Cloud Storage

***ASSIGNMENT* BEFORE YOUR LAB:**

1. Pick 5 to 10 classes from one of these datasets (do not take all images from each class if you don't have space)
   - https://www.kaggle.com/kmader/food41/version/5#
   - http://www.robots.ox.ac.uk/~vgg/data/pets/
   - http://www.robots.ox.ac.uk/~vgg/data/flowers/
   - **any other dataset you have?**
2. Put them in folders like ——————————————>

```
data/
    dogs/
        dog001.jpg
        dog002.jpg
        ...
    cats/
        cat001.jpg
        cat002.jpg
        ...
```

3. Upload to Google Drive if you plan to use COLAB

Lab 2: NN and CNNs applied in Computer Vision

- Frameworks
  - Caffe, **Tensorflow+Keras**, **Pytorch**, …
- Model **zoos**
  - in every framework *(you should always start by looking into existing models. “rare” to “require” to implement a network from scratch)*

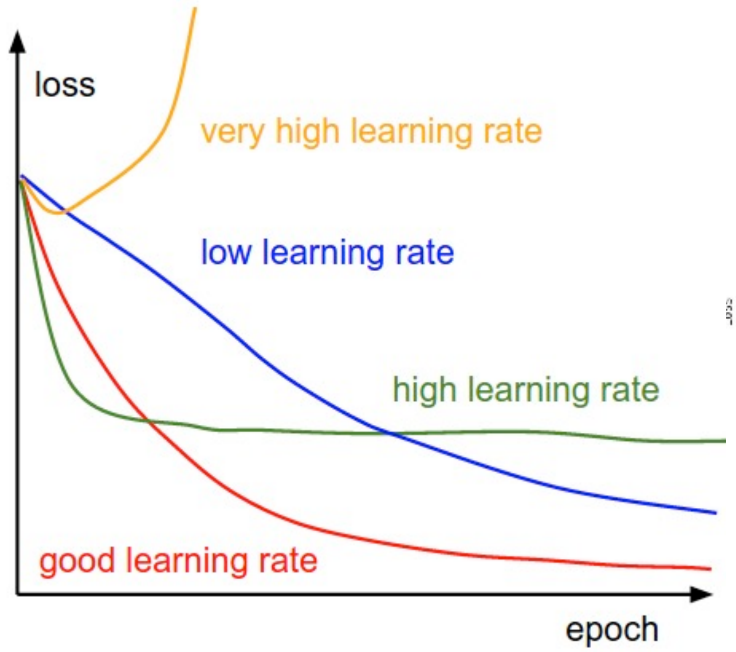
Understanding the training process

Initial sanity checks (after a few “toy” iterations)

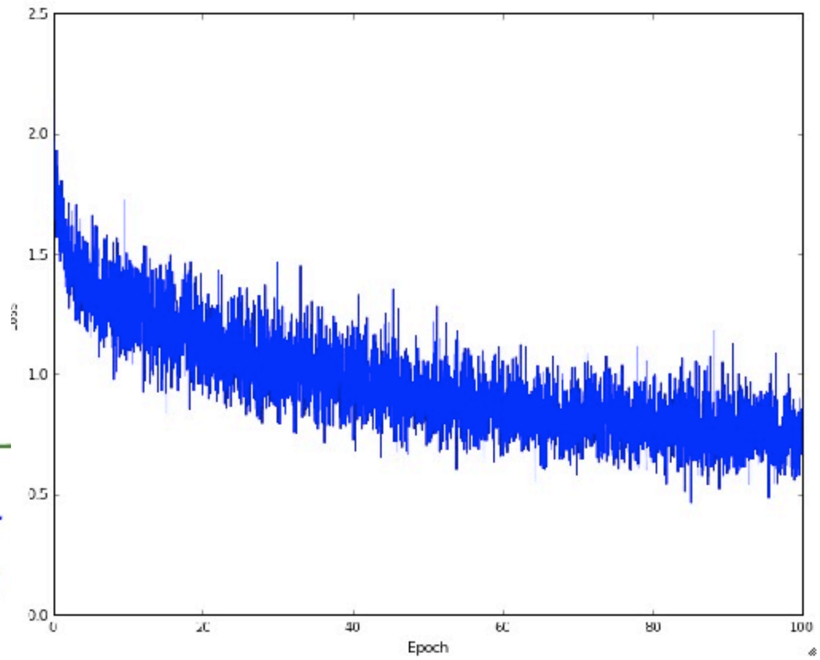
- *Correct* loss at chance
- Increasing regularisation strength increases loss
- Overfit a tiny training set (you can set regularisation strength to zero to better see this)

Understanding the training process

- Monitor the loss function



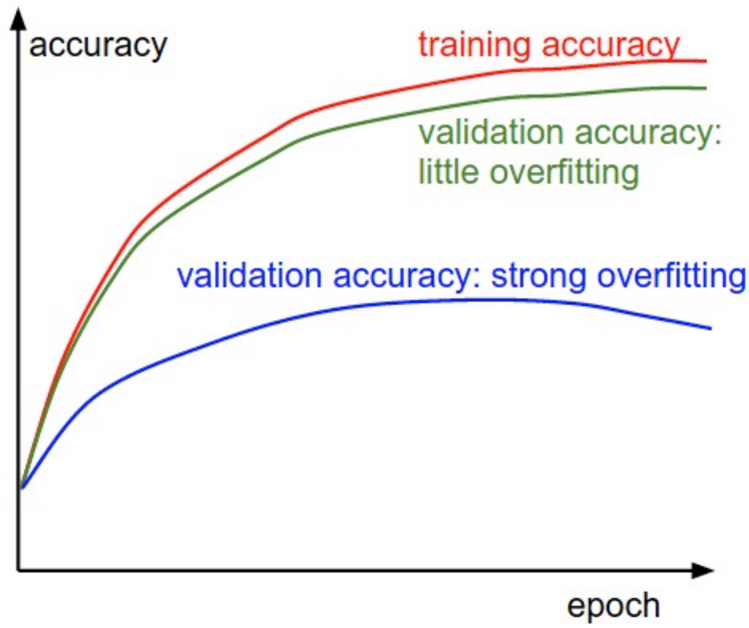
Typical loss function.
Batch size a little low? (noisy plot)



http://cs231n.stanford.edu Class notes

Understanding the training process

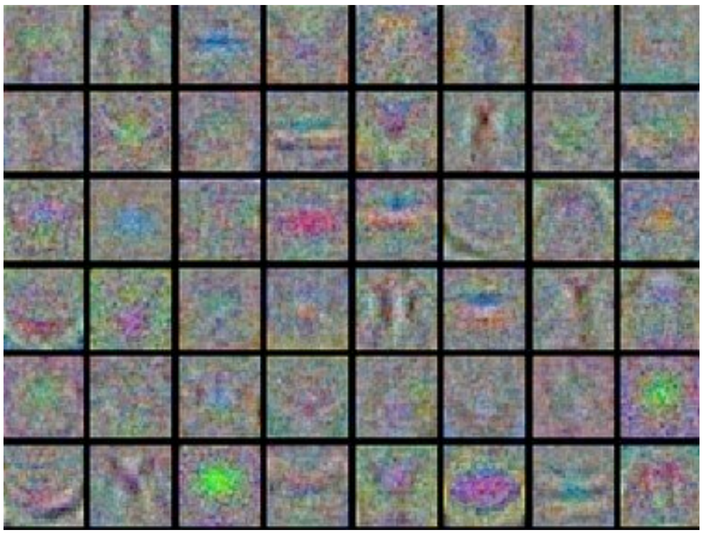
- Monitor train/val accuracy

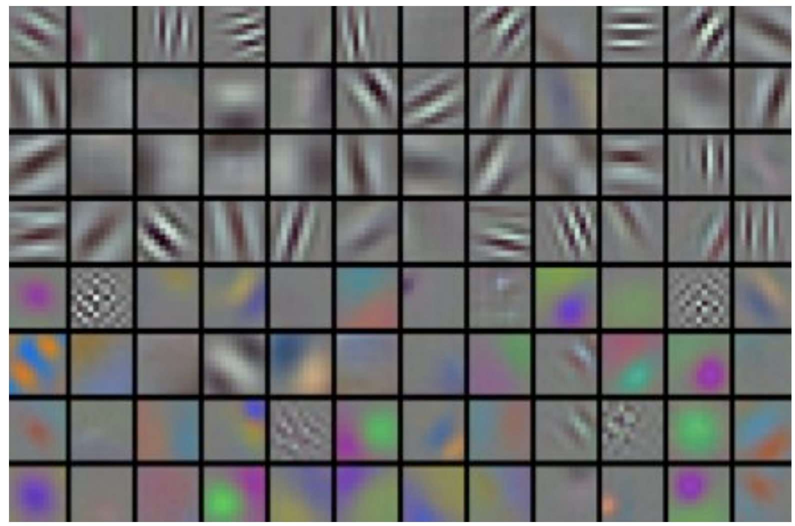


http://cs231n.stanford.edu Class notes

Understanding the training process

- Visualize first conv layer weights





Understanding the training process

Additional tips:

- **Decay your learning rate** during training
- If you can afford it
  - **Search** for good **hyperparameters** with random search (not grid search). Fom coarse to fine
  - Consider **model ensembles** (not in our labs)

Lab 2: NN and CNNs applied in Computer Vision

- Understand a **DNN implemented from scratch**
- Implement a toy CNN with Keras
- Fine-tune a well-known CNN architecture with Keras

All this is applied to Image Classification

- Optional tasks:
  - implement variations of the classification networks
  - explore object recognition model
  - explore semantic segmentation model

Exercise now (start for Lab2)

COLAB