

# Machine Learning (CE 40477)

Fall 2024

Ali Sharifi-Zarchi

CE Department  
Sharif University of Technology

October 20, 2024



## 1 CNNs in Vision

## 2 AlexNet

## 3 ResNet

## 4 Data Preprocessing

## 5 Data Augmentation

## 6 Transfer Learning

## 7 References

## 1 CNNs in Vision

② AlexNet

3 ResNet

## 4 Data Preprocessing

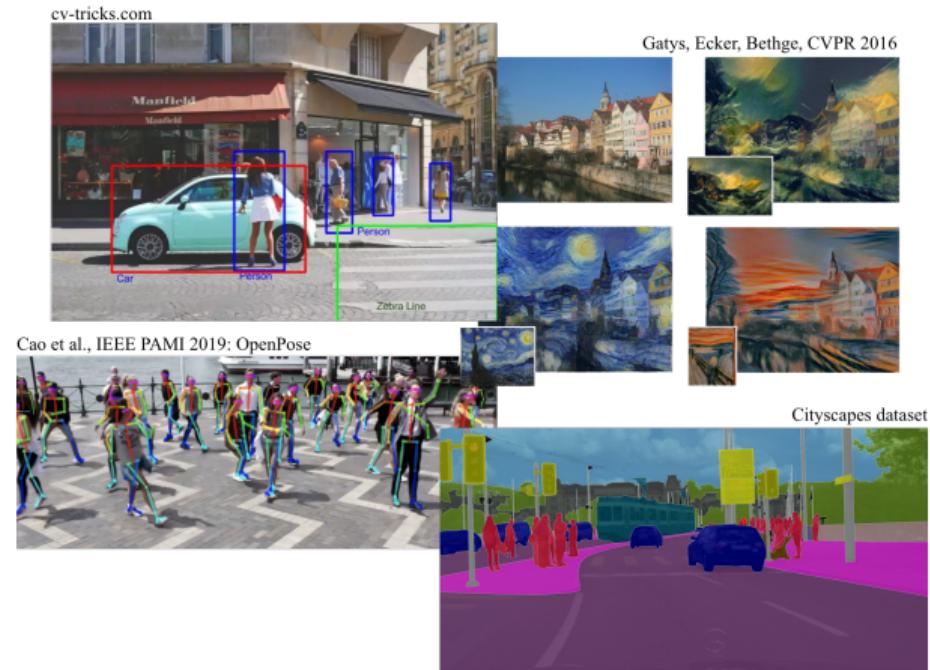
## 5 Data Augmentation

## 6 Transfer Learning

## 7 References

## CNNs are everywhere!

- The success of deep learning for image recognition has been driven by two key factors:
    - Large-scale CNNs
    - Transfer learning



# ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

2009: IMAGENET

- Dataset and benchmark on image classification
  - 1 million images with ground truth class labels for training (hand-annotated) 1000 object categories

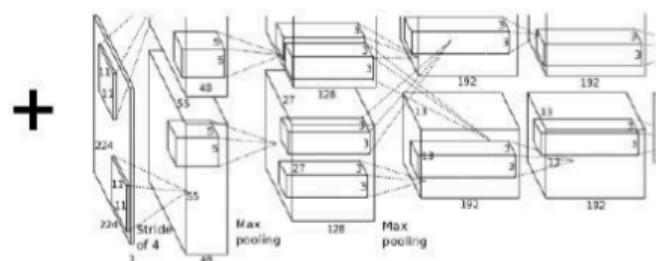


Deng et al., CVPR 2009

CNNs are old. Why did it work eventually?



## Big Data: ImageNet



## Deep Convolutional Neural Network



## Backprop on GPU

**+ A number of small tweaks**



## Image Source

## 1 CNNs in Vision

## 2 AlexNet

## 3 ResNet

## 4 Data Preprocessing

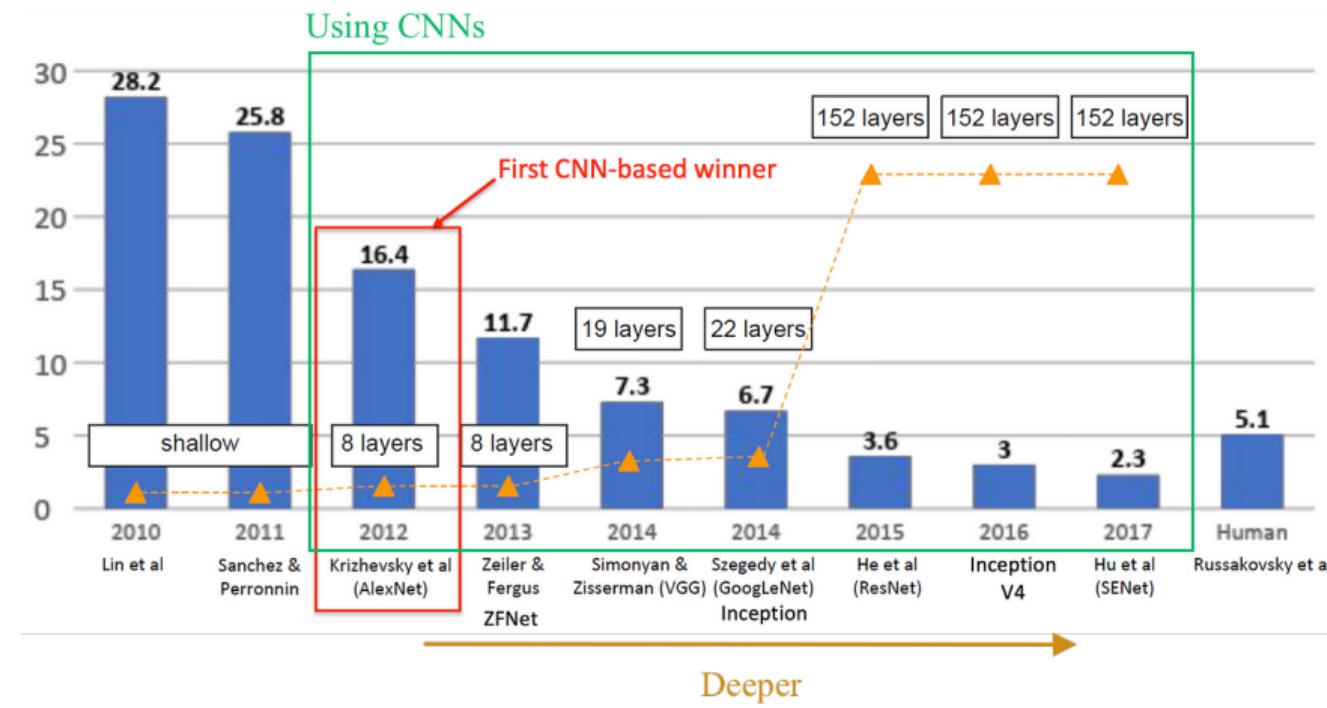
## 5 Data Augmentation

## 6 Transfer Learning

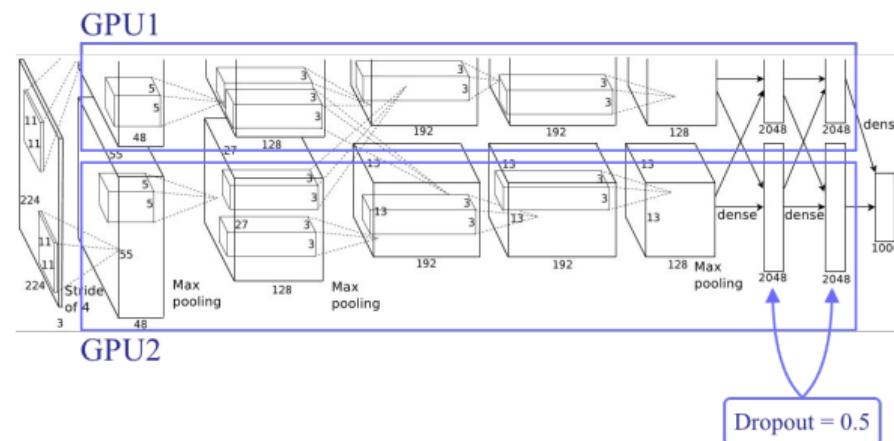
## 7 References

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

- Reduced the error rate by 10 %



AlexNet [Krizhevsky, et al. NIPS (2012)]



- Training: 6 days on 2 NVIDIA GTX 580 GPUs (3 GB RAM)
  - 8 layers / 60 million parameters and 650,000 neurons
  - 5 convolutional layers, some of which are followed by max-pooling layers
  - 3 fully-connected layers
  - Testing: Multi-crop
    - Classify different shifts of the image and vote over the lot!
    - Test-time data augmentation

## AlexNet Architecture

CONV1

## MAX POOL1

NORM1

## CONV2

## MAX POOL 2

NORM2

CONV3

CONV4

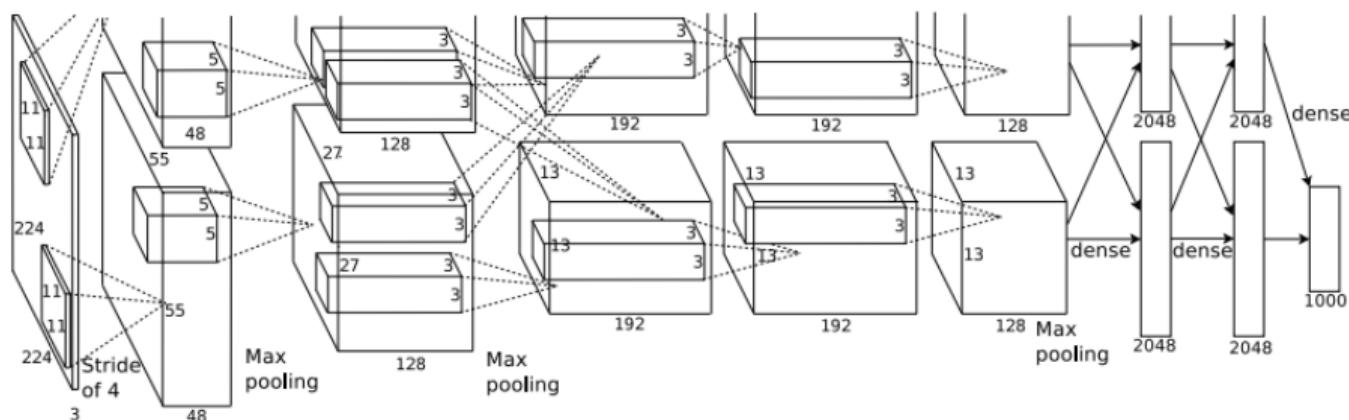
**CONV5**

## MAX POOL 3

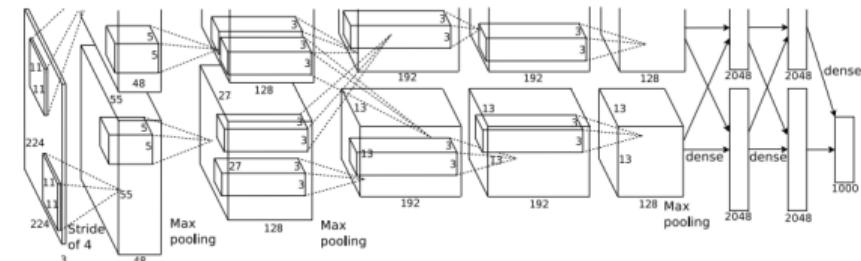
EC6

EC7

EC8



## AlexNet Architecture



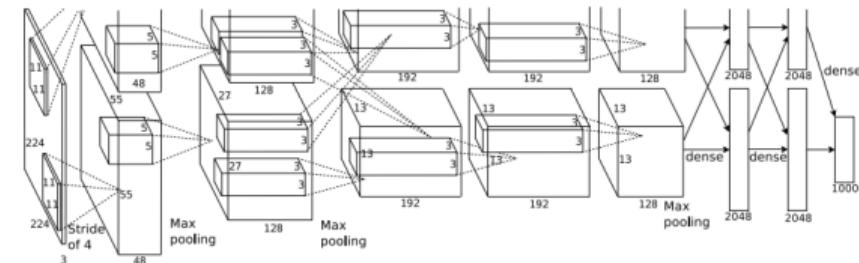
Input:  $227 \times 227 \times 3$  images

**First layer** (CONV1): 96  $11 \times 11$  filters applied at stride 4

$$W' = \frac{W-F+2P}{S+1}$$

⇒ Q: what is the output volume size? Hint:  $(227-11)/4+1 = 55$

## AlexNet Architecture



Input:  $227 \times 227 \times 3$  images

**First layer** (CONV1): 96  $11 \times 11$  filters applied at stride 4

$$W' = \frac{W-F+2F}{S+1}$$

⇒ Output volume [55 × 55 × 96]

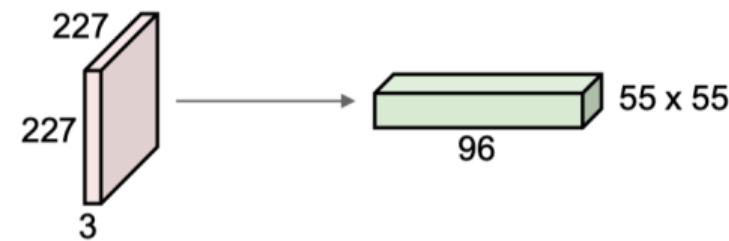
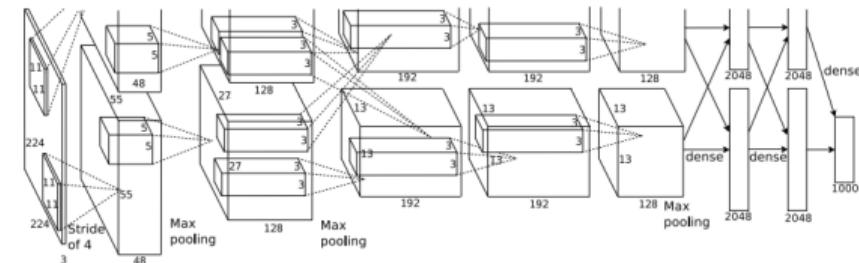


Figure: Alex Krizhevsky, et al., 2012

## AlexNet Architecture



Input:  $227 \times 227 \times 3$  images

**First layer** (CONV1): 96  $11 \times 11$  filters applied at stride 4

⇒ Output volume [55 × 55 × 96]

Q: What is the total number of parameters in this layer?

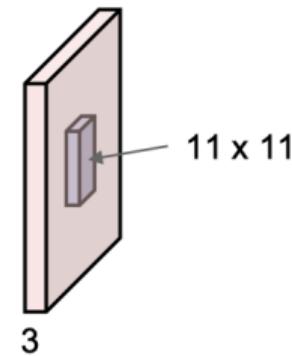
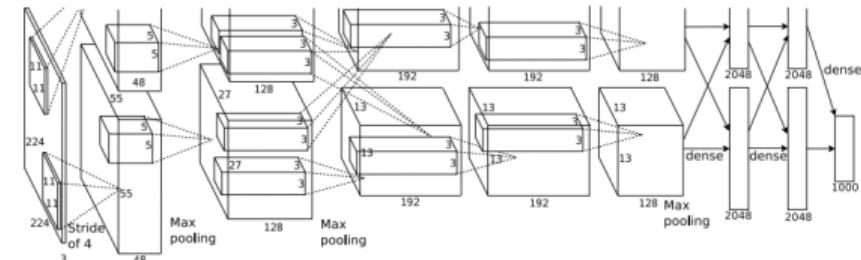


Figure: Alex Krizhevsky, et al., 2012

# AlexNet Architecture



Input:  $227 \times 227 \times 3$  images

**First layer (CONV1):** 96  $11 \times 11$  filters applied at stride 4

⇒ Output volume  $[55 \times 55 \times 96]$

Parameters:  $(11 \times 11 \times 3 + 1) \times 96 = 35K$

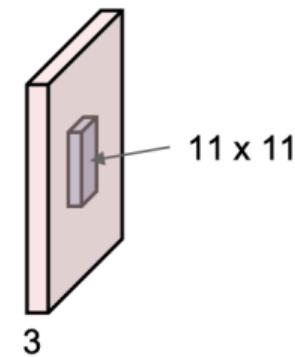
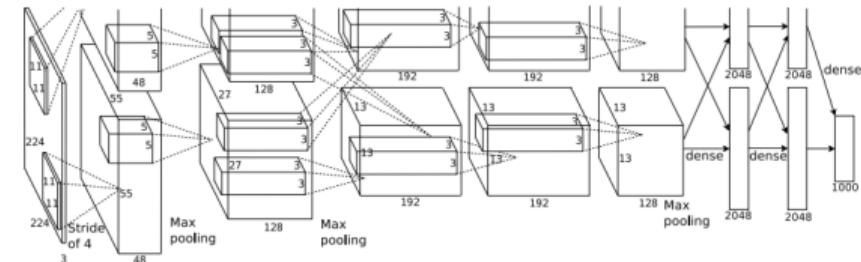


Figure: Alex Krizhevsky, et al., 2012

# AlexNet Architecture



Input:  $227 \times 227 \times 3$  images

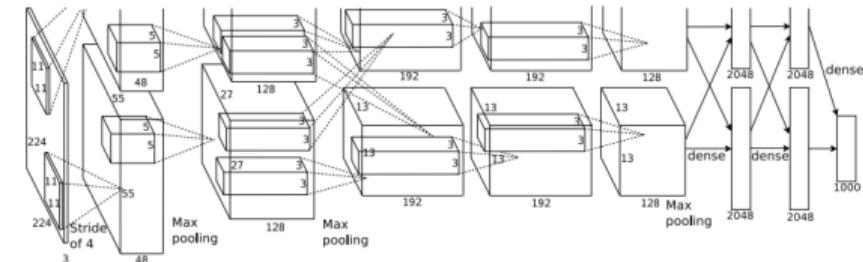
After CONV1:  $55 \times 55 \times 96$

**Second layer (POOL1):**  $3 \times 3$  filters applied at stride 2

Q: What is the output volume size? Hint:  $(55-3)/2+1 = 27$

$$W' = \frac{W-F+2P}{S+1}$$

# AlexNet Architecture



Input:  $227 \times 227 \times 3$  images

After CONV1:  $55 \times 55 \times 96$

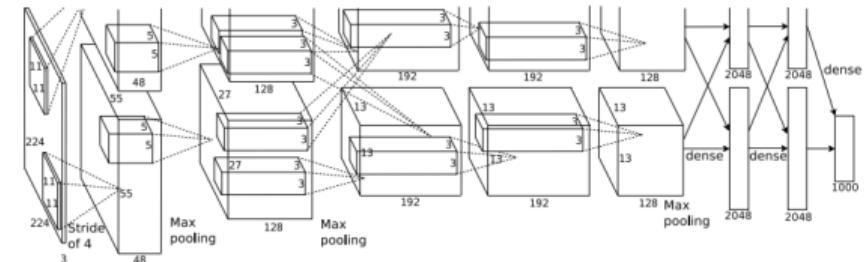
**Second layer (POOL1):**  $3 \times 3$  filters applied at stride 2

Output volume:  $27 \times 27 \times 96$

$$W' = \frac{W-F+2P}{S+1}$$

Q: What is the number of parameters in this layer?

# AlexNet Architecture



Input:  $227 \times 227 \times 3$  images

After CONV1:  $55 \times 55 \times 96$

**Second layer (POOL1):**  $3 \times 3$  filters applied at stride 2

Output volume:  $27 \times 27 \times 96$

Parameters: 0!

## AlexNet Full Architecture

[ $227 \times 227 \times 3$ ] INPUT

[ $55 \times 55 \times 96$ ] CONV1: 96  $11 \times 11$  filters at stride 4, pad 0

[ $27 \times 27 \times 96$ ] **MAX POOL1**: MAX POOL1:  $3 \times 3$  filters at stride 2

[ $27 \times 27 \times 96$ ] NORM1: Normalization layer

[ $27 \times 27 \times 256$ ] **CONV2**: 256  $5 \times 5$  filters at stride 1, pad 2

[ $13 \times 13 \times 256$ ] MAX POOL2:  $3 \times 3$  filters at stride 2

[ $13 \times 13 \times 256$ ] NORM2: Normalization layer

[ $13 \times 13 \times 384$ ] CONV3: 384  $3 \times 3$  filters at stride 1, pad 1

[ $13 \times 13 \times 384$ ] **CONV4**: 384  $3 \times 3$  filters at stride 1, pad 1

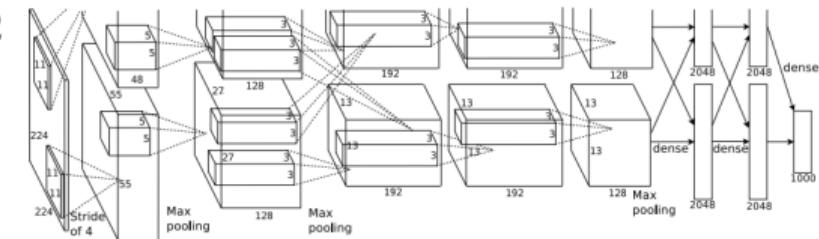
[ $13 \times 13 \times 256$ ] CONV5: 256  $3 \times 3$  filters at stride 1, pad 1

[ $6 \times 6 \times 256$ ] **MAX POOL 3**:  $3 \times 3$  filters at stride 2

[4096] EC6 4096 neurons

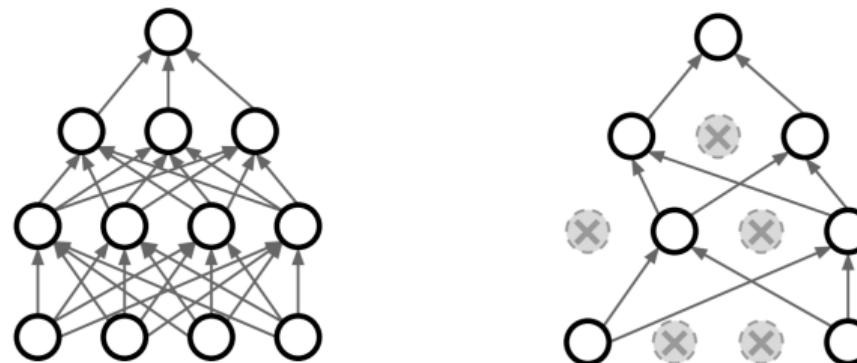
[4096] EC7 4096 neurons

[1000] EC8 1000 neurons (class scores)



# Dropout

- In each forward pass, randomly set some neurons to zero
  - Probability of dropping is a hyperparameter; 0.5 is common

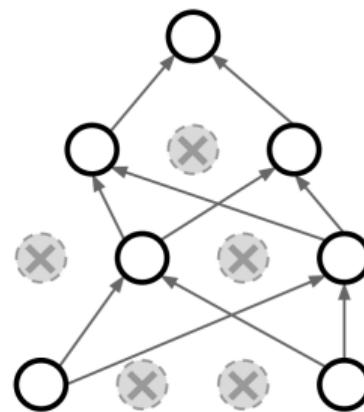


Srivastava et al., JMLR 2014

- Thus, a smaller network is trained for each sample
  - Smaller network provides a regularization effect

# Dropout

- How can dropout be a good idea?



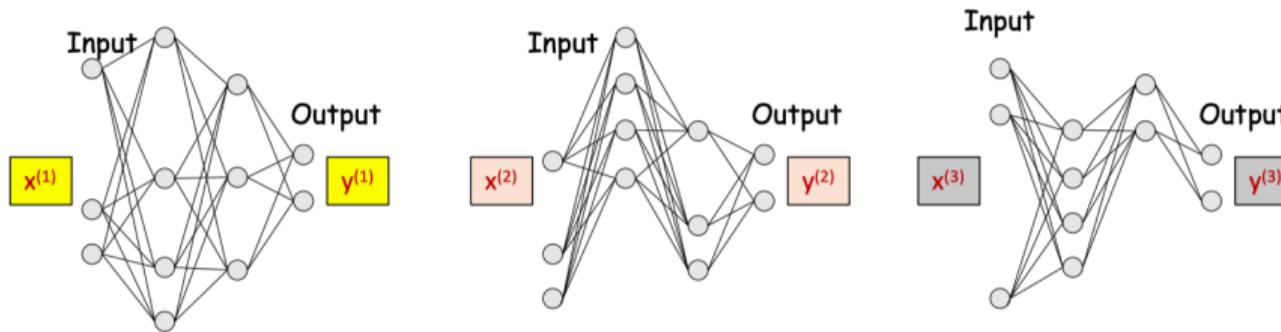
Forces the network to have a redundant representation;  
Prevents co-adaptation of features



Hinton et al., 2012

# Dropout

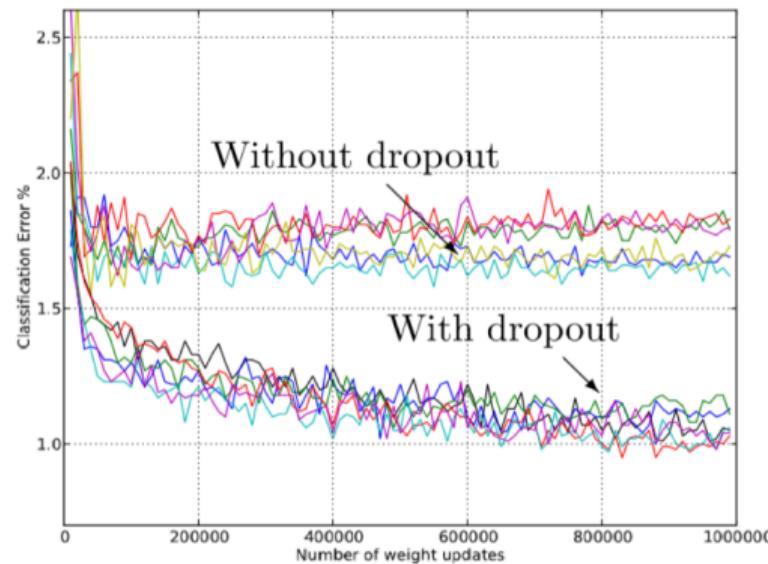
- During training: Backpropagation is effectively performed only over the remaining network
  - The effective network is different for different inputs
  - Gradients are obtained only for the weights and biases from On nodes to On nodes
  - For the remaining, the gradient is just 0



The pattern of dropped nodes  
changes for each input  
i.e. in every pass through the net

# Dropout

- Test error for different architectures on MNIST with and without dropout
  - 2-4 hidden layers with 1024-2048 units

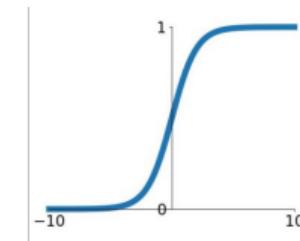


Srivastava et al., 2013

# Flashback To Activation Functions

## Sigmoid:

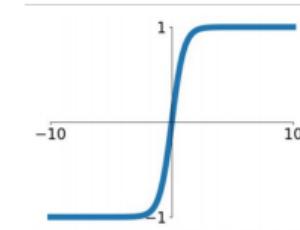
- $\sigma(x) = \frac{1}{1+e^{-x}}$
- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating firing rate of a neuron
- **Problems:**
  - Saturated neurons kill the gradients
  - Sigmoid outputs are not zero-centered



Sigmoid

## Tanh:

- $tanh(x) = \frac{1-e^{-2x}}{1+e^{2x}} = 1 - 2\sigma(2x)$
- Squashes numbers to range [-1,1]
- Zero centered (nice)
- Still kills gradients when saturated

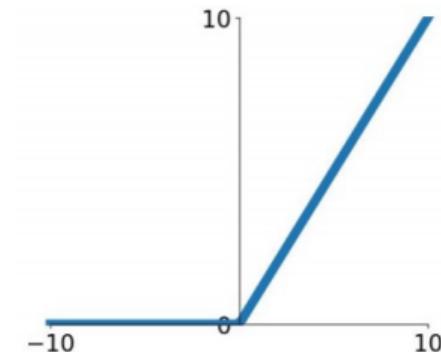


Tanh

# ReLU

## ReLU

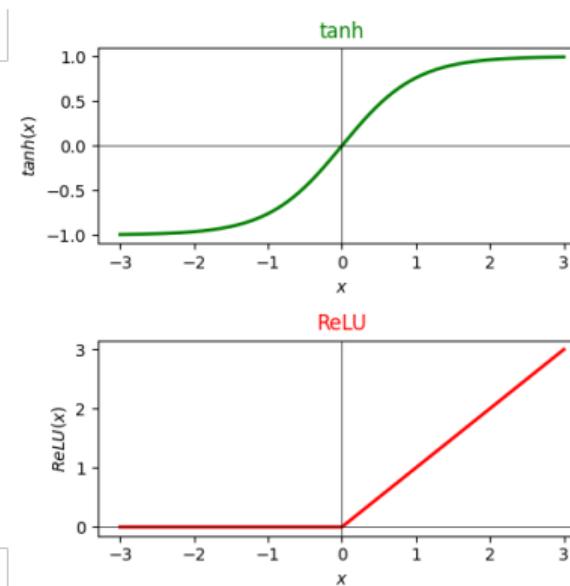
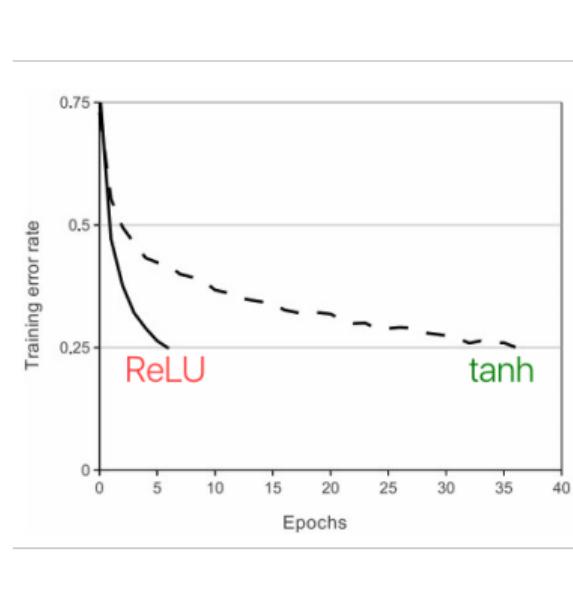
- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)
- Actually more biologically plausible than sigmoid
- **Problems:**
  - Not zero-centered output
  - An annoyance:
    - ★ hint: what is the gradient when  $x < 0$ ?



ReLU

# ReLU

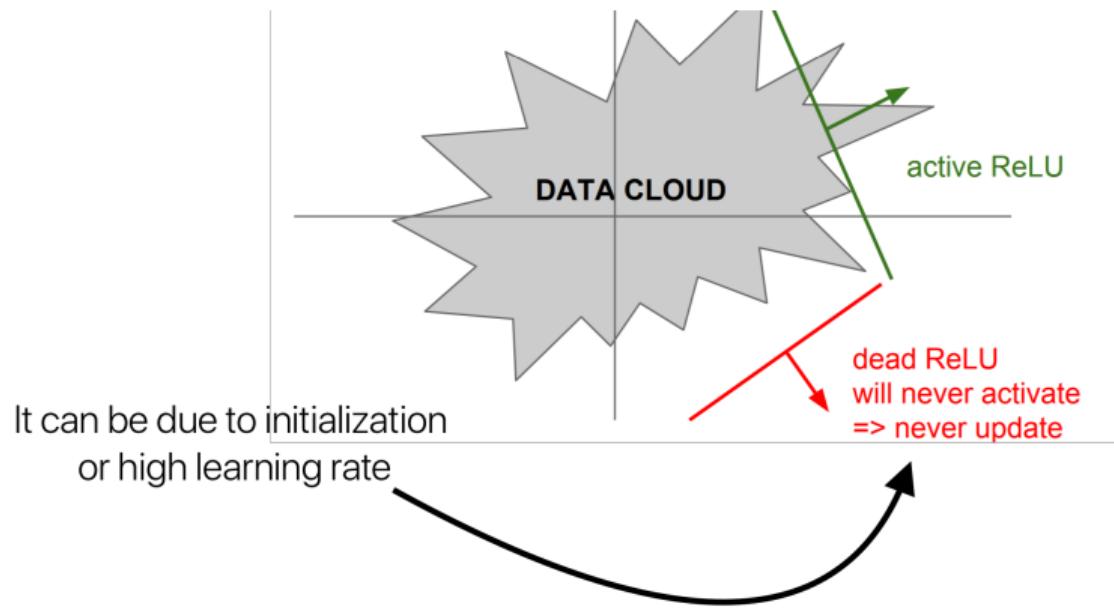
- ReLU trains faster than tanh



Krizhevsky, et al., 2012

# ReLU Initialization

- people like to initialize ReLU neurons with slightly positive biases (e.g. 0.01)
- It can be due initialization or high learning rate



# Learning magic in Alexnet

- First use of ReLU
  - Made a large difference in convergence
- "Dropout" 0.5 (in FC layers only)
- Heavy data augmentation
- SGD Momentum 0.9 with mini batch size 128
- L2 weight decay 5e-4
- Learning rate: 0.01, decreased by 10 every time validation accuracy plateaus
- Evaluated using: Validation accuracy
- **Final top-5 error: 18.2% with a single net, 15.4% using an ensemble of 7 networks**
  - Lowest prior error using conventional classifiers: > 25 %

## 1 CNNs in Vision

## 2 AlexNet

## 3 ResNet

## 4 Data Preprocessing

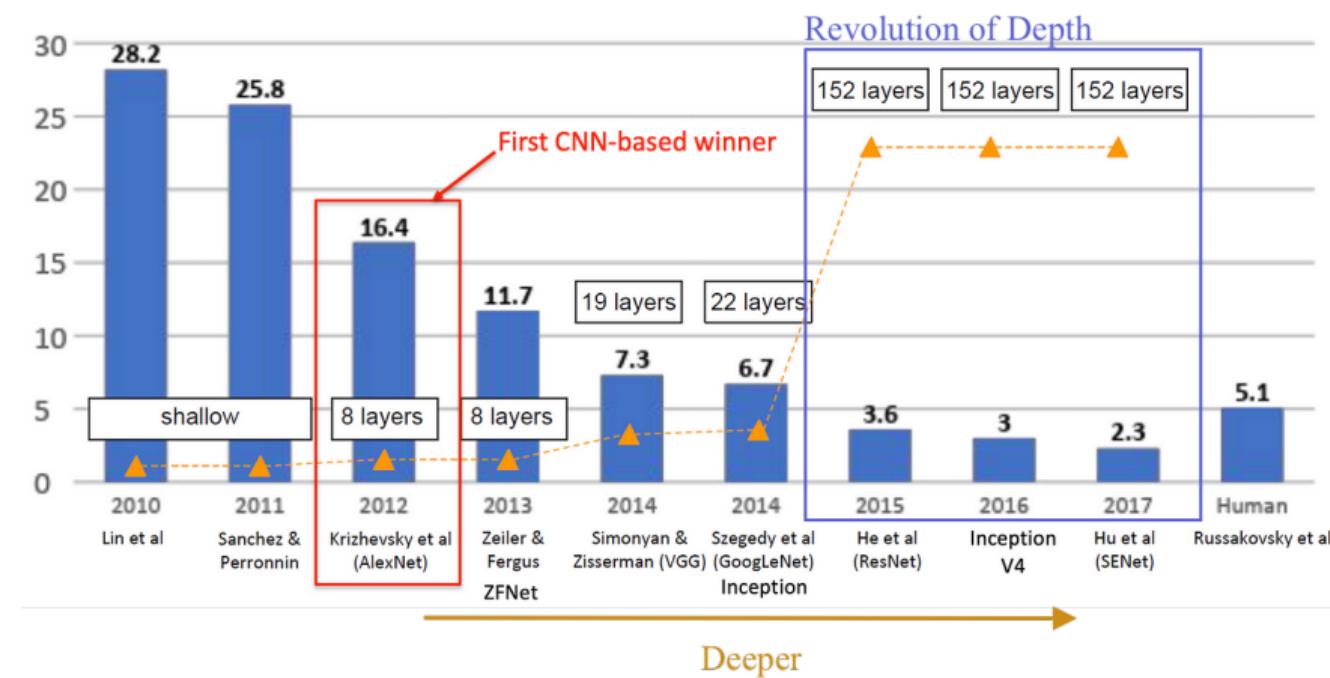
## 5 Data Augmentation

## 6 Transfer Learning

## 7 References

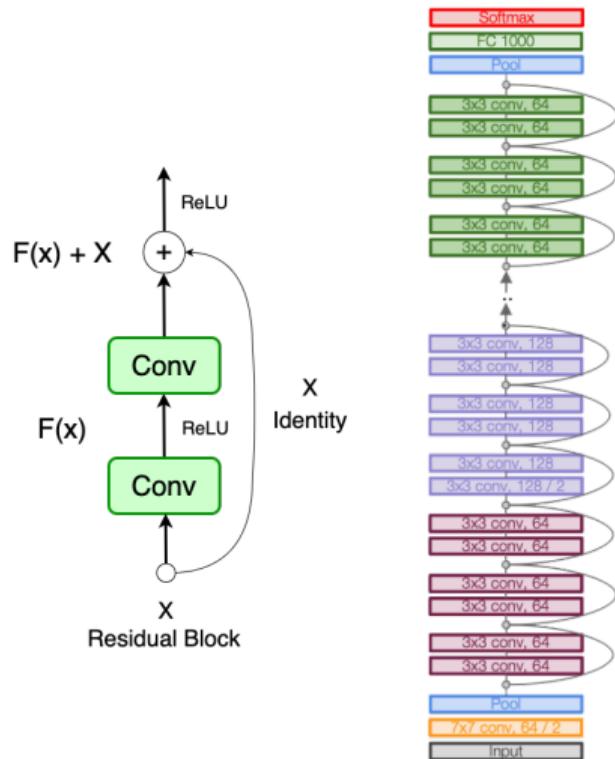


## Revolution of Depth



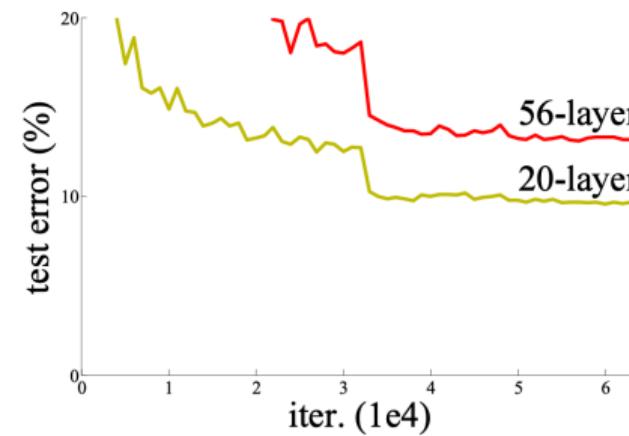
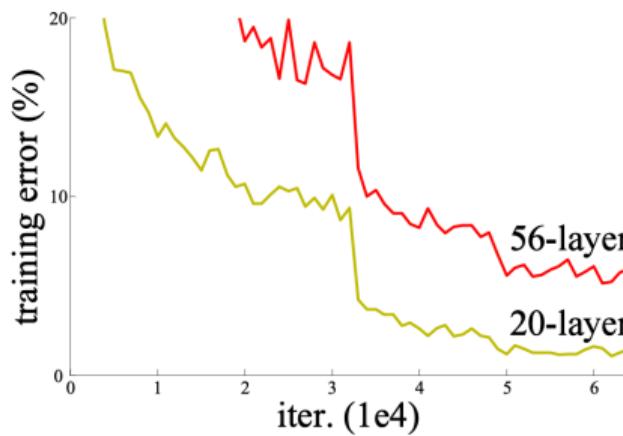
ResNet [He, et al., (2015)]

- Very deep networks using residual connections
  - ResNet introduces the concept of skip connections (or residual connections) that allow the gradient to be directly backpropagated to earlier layers
  - Skip connections help in overcoming the degradation problem, where the accuracy saturates and then degrades rapidly as the network depth increases.
  - 152-layer model for ImageNet
  - ILSVRC15 classification winner (3.57 % top 5 error)
  - Swept all classification and detection competitions in ILSVRC15 and COCO15!



ResNet [He, et al., (2015)]

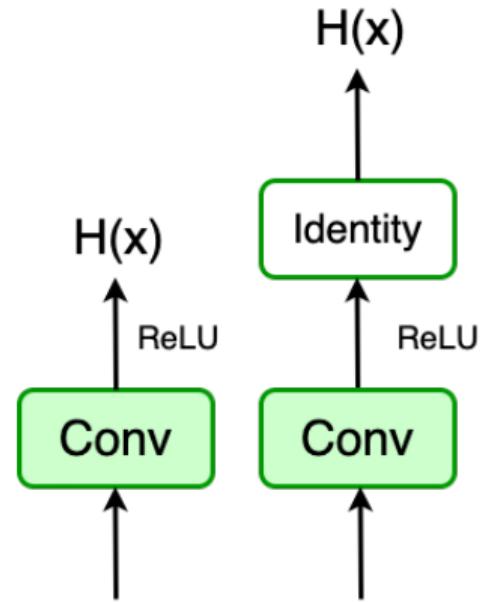
- What happens when we continue stacking deeper layers on a “plain” convolutional neural network?
  - **Question:** What’s strange about these training and test curves?



Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer plain networks

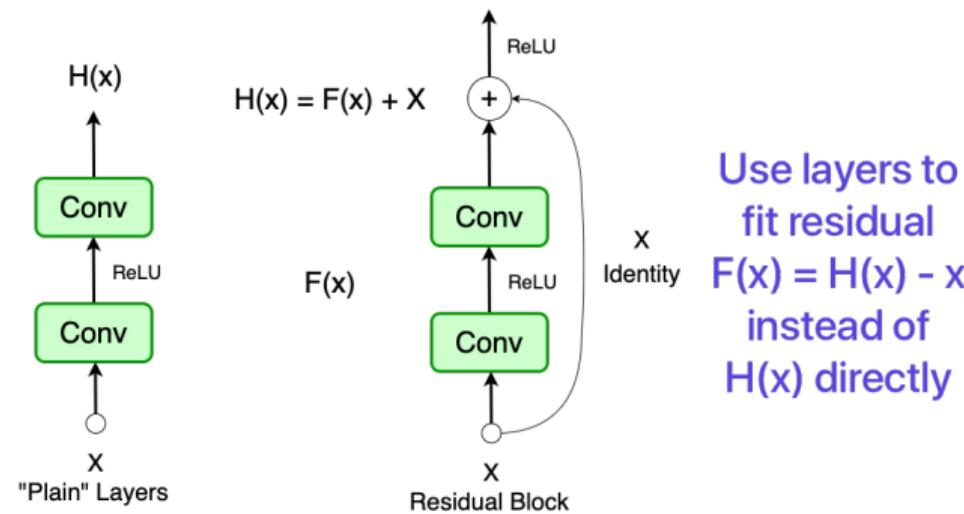
## ResNet [He, et al., (2015)]

- **Fact:** Deeper models have more representation power (more parameters) than shallower models.
- But **56-layer model performs worse** on both training and test error and its not caused by overfitting!
- **Hypothesis:** the problem is an optimization problem, deeper models are harder to optimize
- What should the deeper model learn to be at least as good as the shallower model?
- A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.



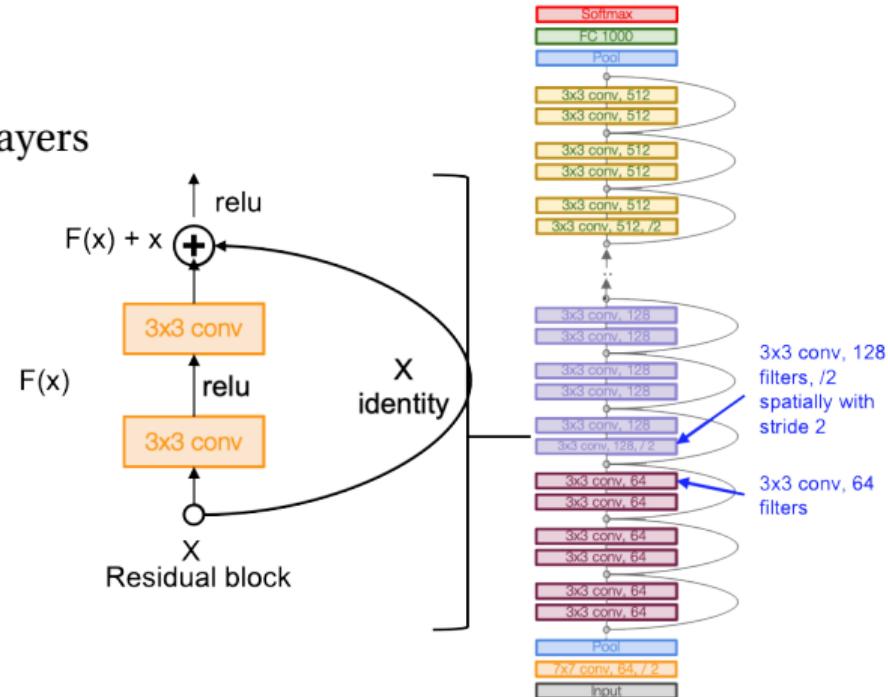
## ResNet [He, et al., (2015)]

- Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping
- Identity mapping:  $H(x) = x$  if  $F(x) = 0$



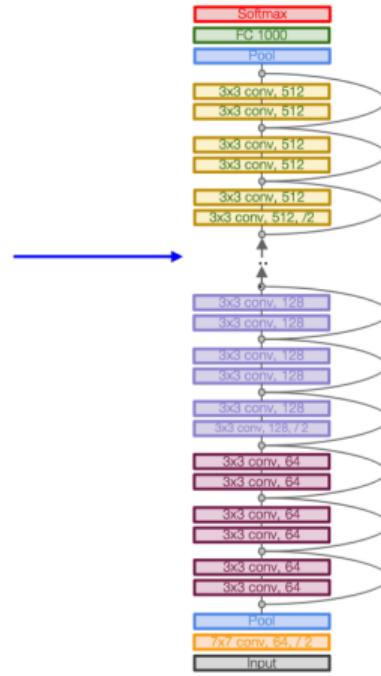
## Full ResNet architecture

- Stack residual blocks
  - Every residual block has two  $3 \times 3$  conv layers
  - Periodically, double number of filters and downsample spatially using stride 2 (/2 in each dimension) Reduce the activation volume by half
  - Additional conv layer at the beginning
  - No FC layers at the end (only FC 1000 to output classes)
  - (In theory, you can train a ResNet with input image of variable sizes)



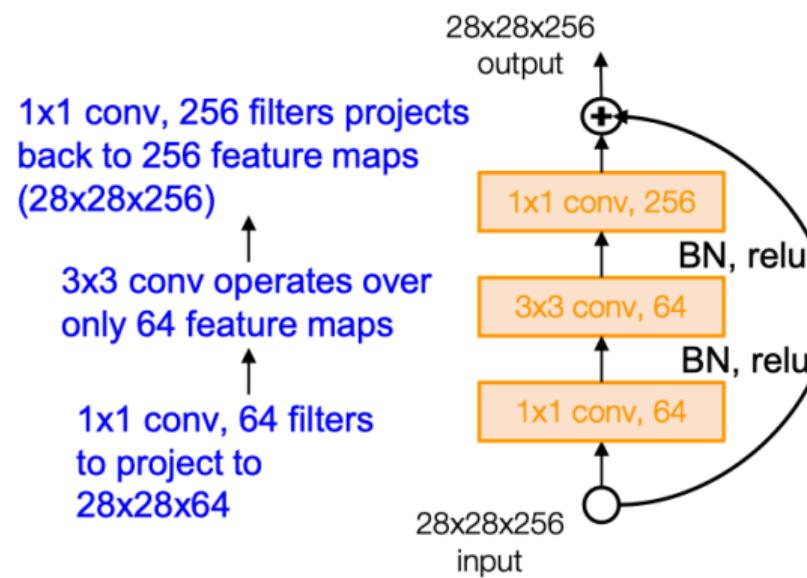
# Full ResNet architecture

- Total depths of 18, 34, 50, 101, or 152 layers for ImageNet



# Full ResNet architecture

- For deeper networks (ResNet- 50+), use bottleneck layer to improve efficiency (similar to GoogLeNet)



# Experimental Results

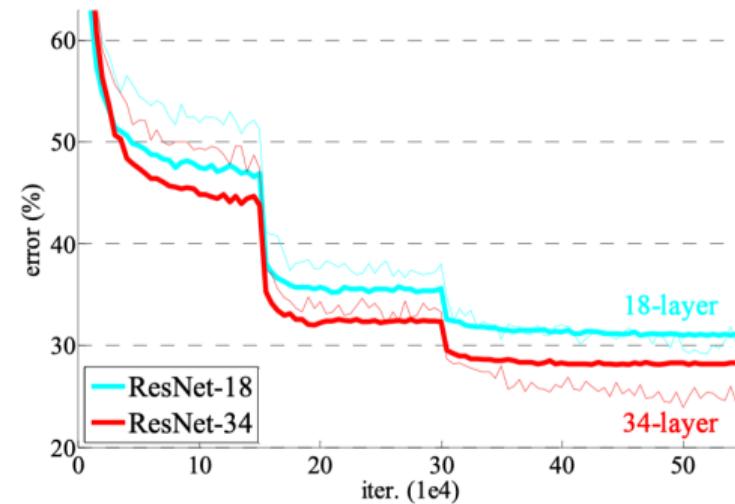
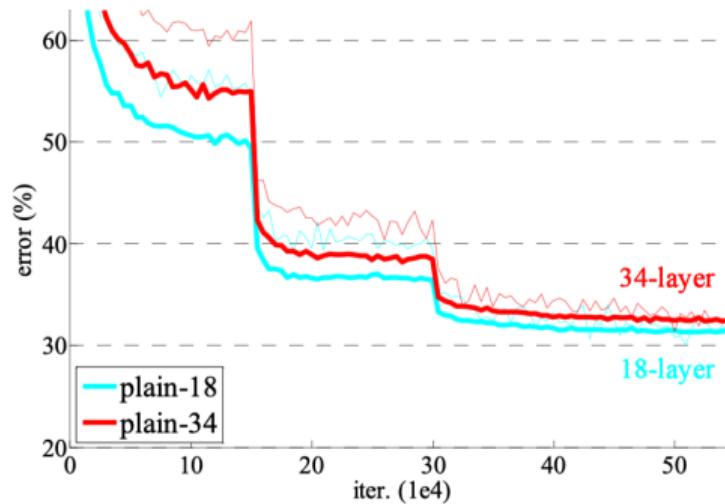
- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lower training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

## MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places** in all five main tracks
  - ImageNet Classification: "*Ultra-deep*" (quote Yann) **152-layer** nets
  - ImageNet Detection: **16%** better than 2nd
  - ImageNet Localization: **27%** better than 2nd
  - COCO Detection: **11%** better than 2nd
  - COCO Segmentation: **12%** better than 2nd

ILSVRC 2015 classification winner (3.6% top 5 error) -- better than “human performance”! (Russakovsky 2014)

## ResNet Experiments



Thin curves denote training error, and bold curves denote validation error [He, et al. 2015].

## Training ResNet in practice:

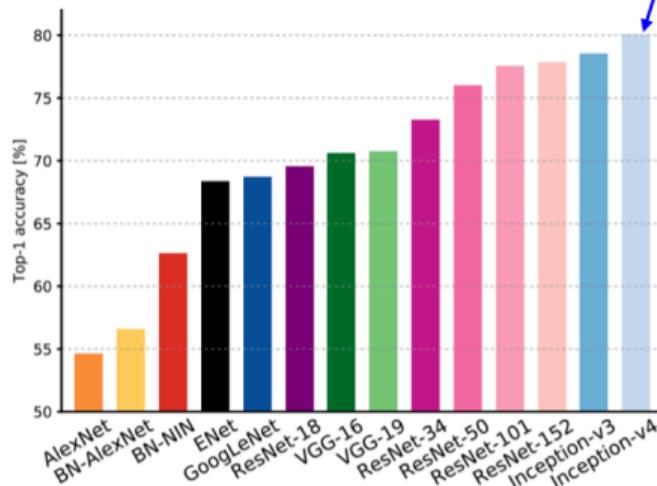
- Batch Normalization after every CONV layer
  - Xavier initialization from He, et al.
  - SGD + Momentum (0.9)
  - Learning rate: 0.1, divided by 10 when validation error plateaus
  - Mini-batch size 256
  - Weight decay of 1e-5
  - No dropout used

# Key Points

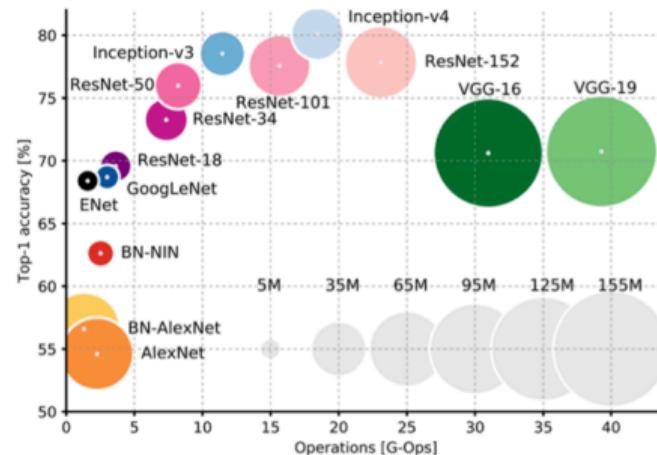
- **AlexNet** showed that you can use CNNs to train Computer Vision models.
- **ResNet** showed us how to train extremely deep networks
  - Limited only by GPU & memory!
  - Showed diminishing returns as networks got bigger
- After ResNet: CNNs were better than the human metric and focus shifted to other topics:
  - ★ Efficient Networks: **MobileNet, ShuffleNet**
  - ★ **Neural Architecture Search** can now automate architecture design

## Model Comparison

## Comparing complexity...



## Inception-v4: Resnet + Inception!



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

## 1 CNNs in Vision

## 2 AlexNet

## 3 ResNet

## 4 Data Preprocessing

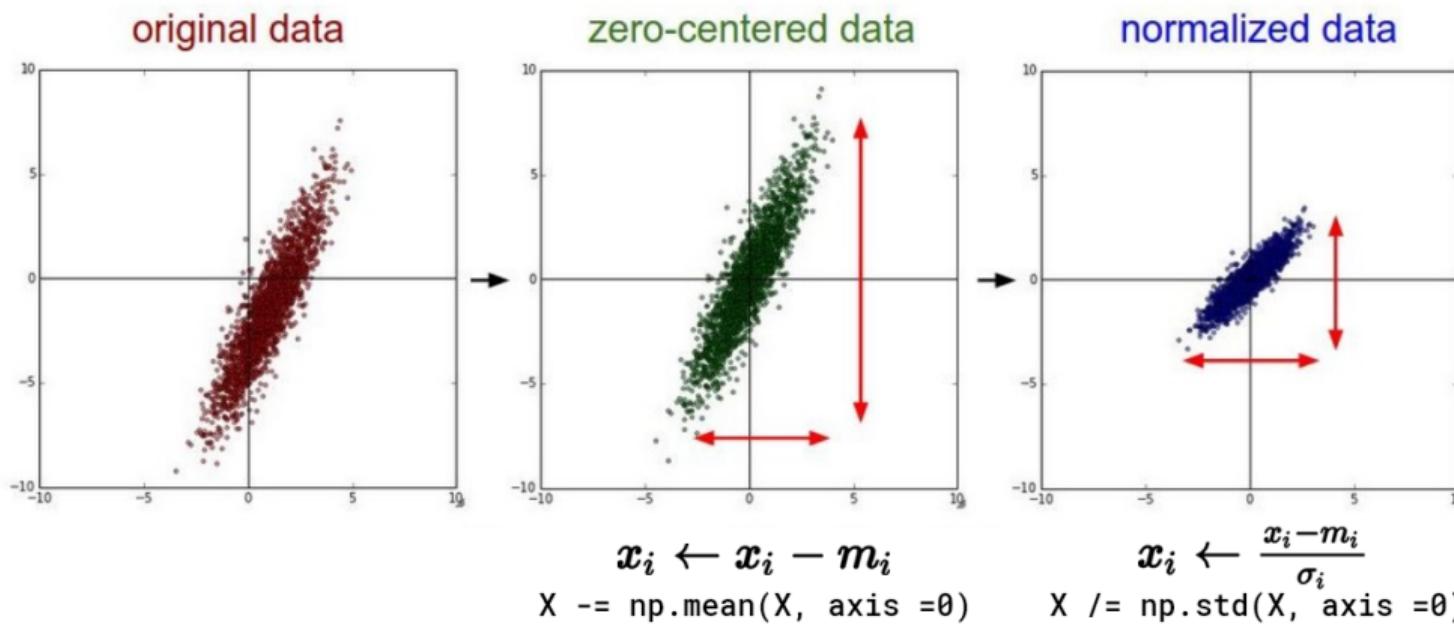
## 5 Data Augmentation

## 6 Transfer Learning

## 7 References

# Normalizing the data

- Assume  $X_{n \times d}$  is data matrix, each sample in a row
- $i$  is the index of dimension ( $i = 1, \dots, d$ )

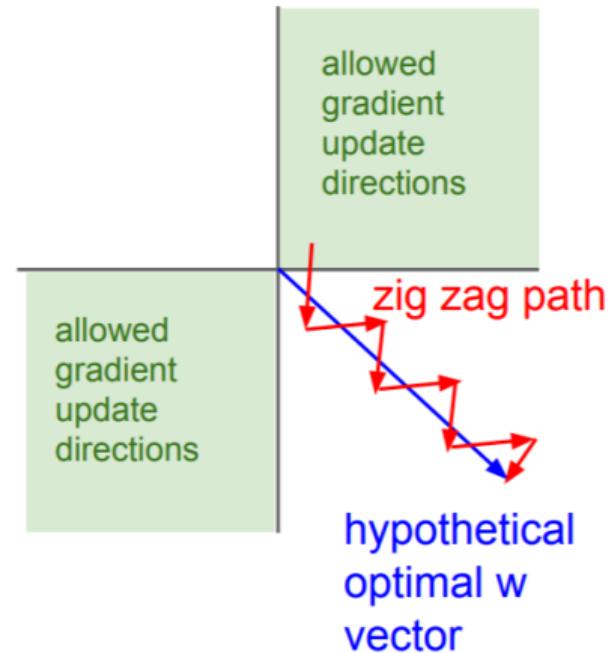


# TLDR: In practice for Images: center only

- Example: consider CIFAR-10 example with  $[32, 32, 3]$  images
- 3 different approaches:
  - ① Subtract the mean image (e.g. AlexNet)
    - mean image =  $[32, 32, 3]$  array
  - ② Subtract per-channel mean (e.g. VGGNet)
    - mean along each channel = 3 numbers
  - ③ Subtract per-channel mean and divide by per-channel std (e.g. ResNet)

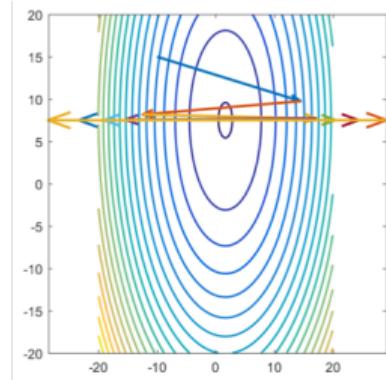
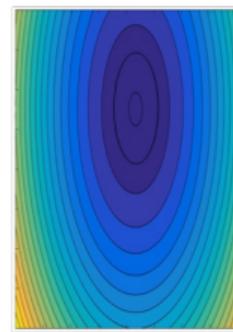
# Why zero-mean the input?

- Reminder: sigmoid
- Consider what happens when the input to a neuron is always positive...
- What can we say about the gradients on  $w$ ?  
**Always all positive or all negative**
  - this is also why you want zero-mean data!

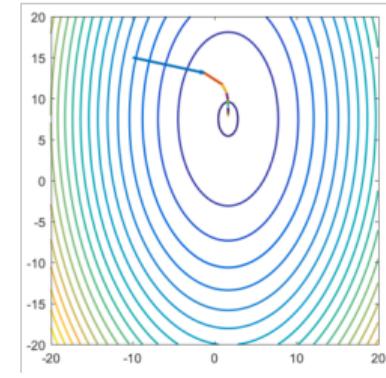
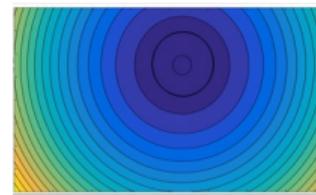


## Why normalize the input?

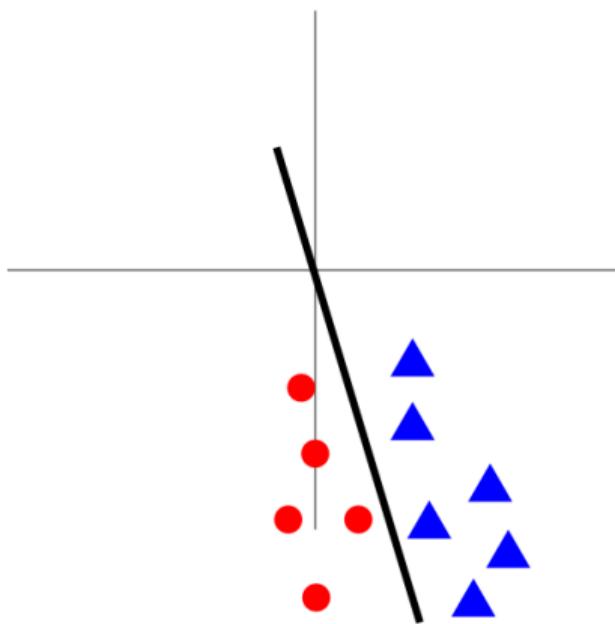
- Very different range of input features
  - i.e. poor conditioning
  - causes noisy movement of gradient



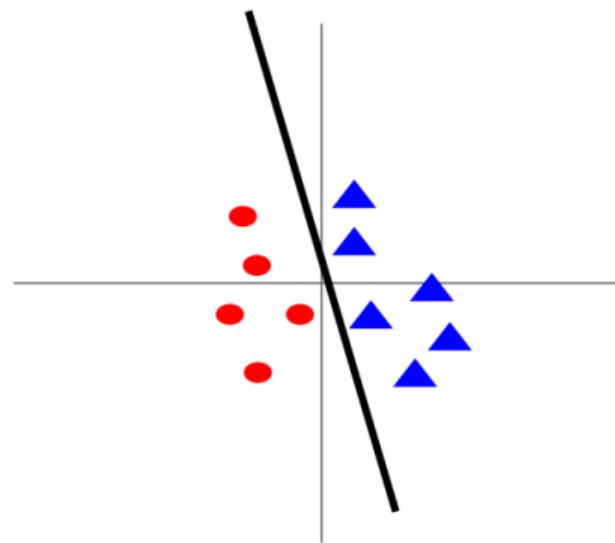
- Normalized data:
    - improves loss landscape  
(alleviate poor conditioning)
    - faster optimization (training)



**Before Normalization:** classification loss very sensitive to changes in weight matrix; hard to optimize

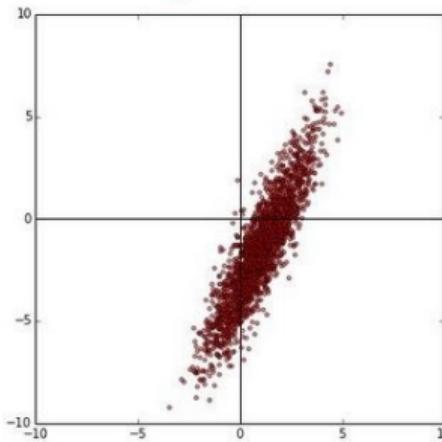


**After Normalization:** loss is less sensitive to small changes in weight matrix; easier to optimize

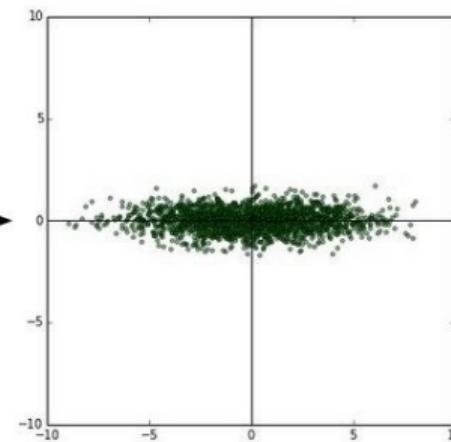


Not common to normalize variance, to do PCA or whitening

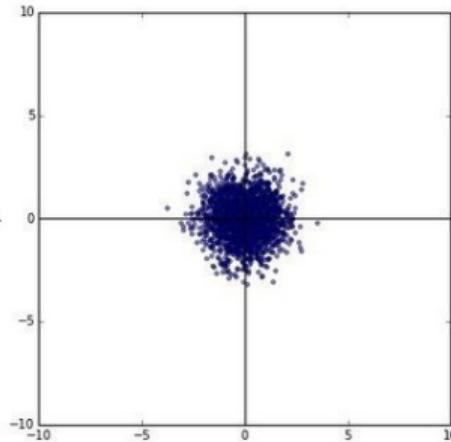
### original data



decorrelated data



## whitened data



Data has diagonal covarianve matrix

Covarianve matrix  
is identity matrix

## 1 CNNs in Vision

## 2 AlexNet

## 3 ResNet

## 4 Data Preprocessing

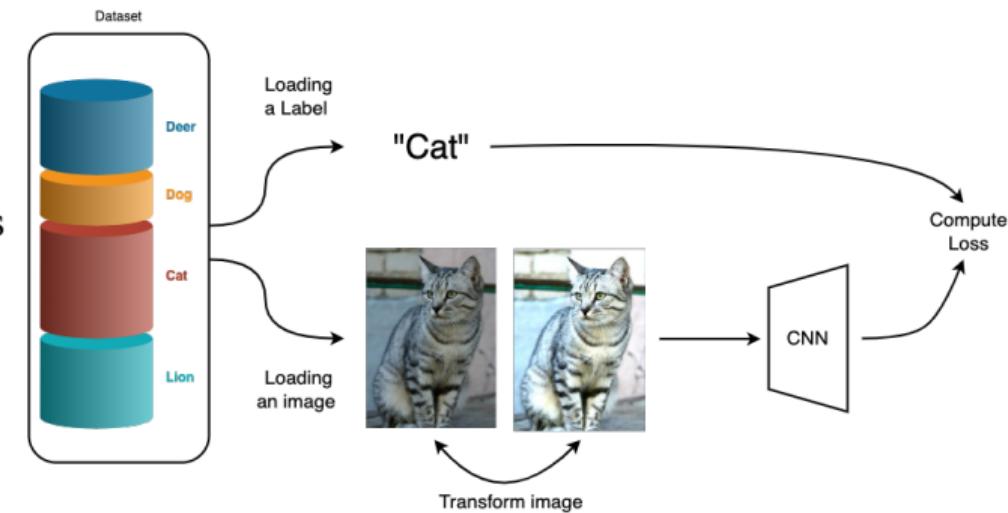
## 5 Data Augmentation

## 6 Transfer Learning

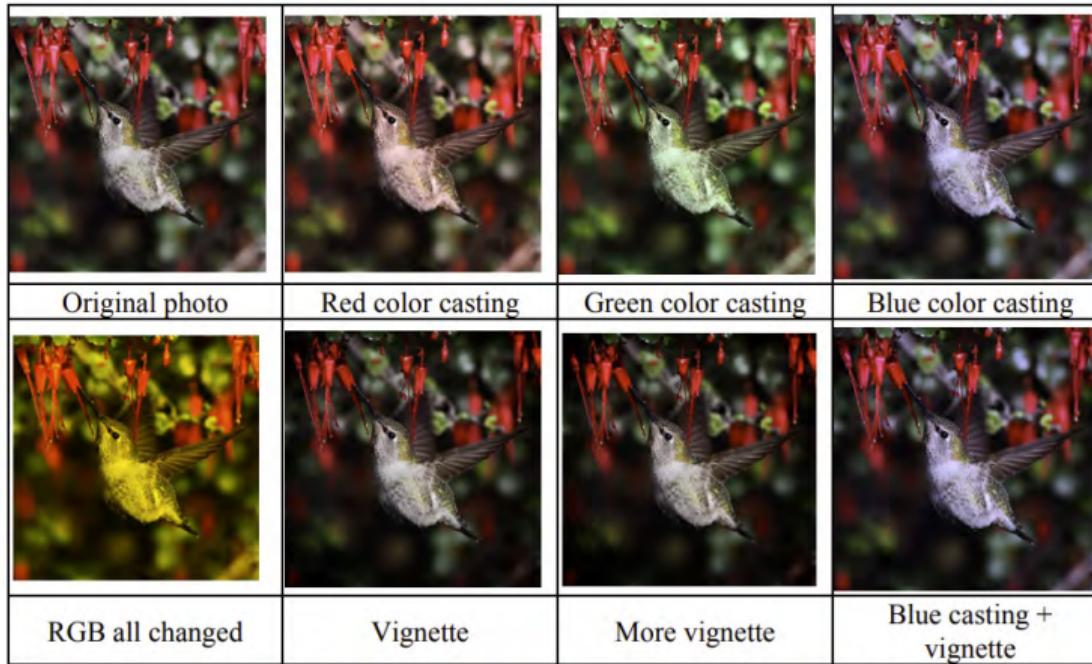
## 7 References

## Motivation

- Getting more training data can be expensive
  - But, sometimes we can generate more training examples from the datasets
  - Transform each input data example in such a way that the label stays the same
  - Benefits:
    - Reduces Overfitting
    - Improves generalization
    - Acts as a regularization
  - Examples of data augmentations
    - Translation
    - Rotation
    - Stretching
    - Shearing
    - Lens distortions, (go crazy)



# Examples of data augmentations

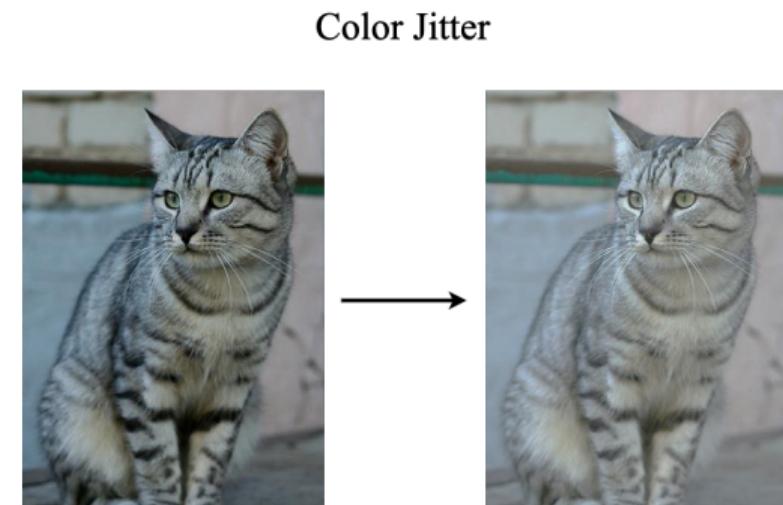


Wu, R., et al. "Deep image: Scaling up image recognition" (2015)

# Augmentations in action

- Simple
  - Randomize contrast and brightness
- Complex
  - ① Apply PCA to all [R, G, B] pixels in training set
  - ② Sample a color offset along principal component directions
  - ③ Add offset to all pixels of a training image

(As seen in AlexNet, ResNet, etc)



This image by [Nikita](#) is licensed under [CC-BY 2.0](#)

# Augmentations in action

- **Training:** sample random crops / scales

ResNet:

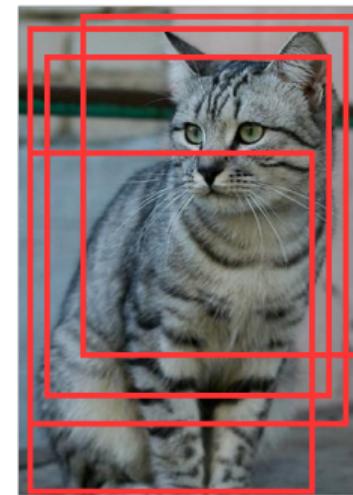
- ① Pick random  $L$  in range [256, 480]
- ② Resize training image, short side =  $L$
- ③ Sample random  $224 \times 224$  patch

- **Testing:** average a fixed set of crops

ResNet:

- ① Resize image at 5 scales: 224, 256, 384, 480, 640
- ② For each size, use 10  $224 \times 224$  crops:  
4 corners + center, + flips

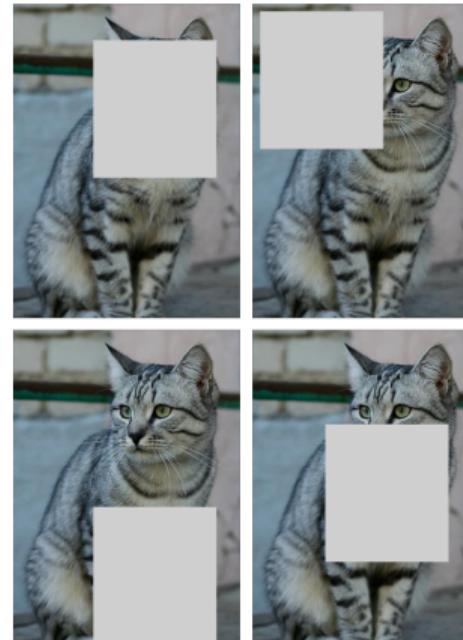
## Random Crops and Scales



# Augmentations in action

- **Training:** Set random image regions to zero
- **Testing:** Use full image
- Works very well for small datasets like CIFAR, less common for large datasets like ImageNet

Cutout



DeVries and Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout", arXiv 2017

## 1 CNNs in Vision

## 2 AlexNet

## 3 ResNet

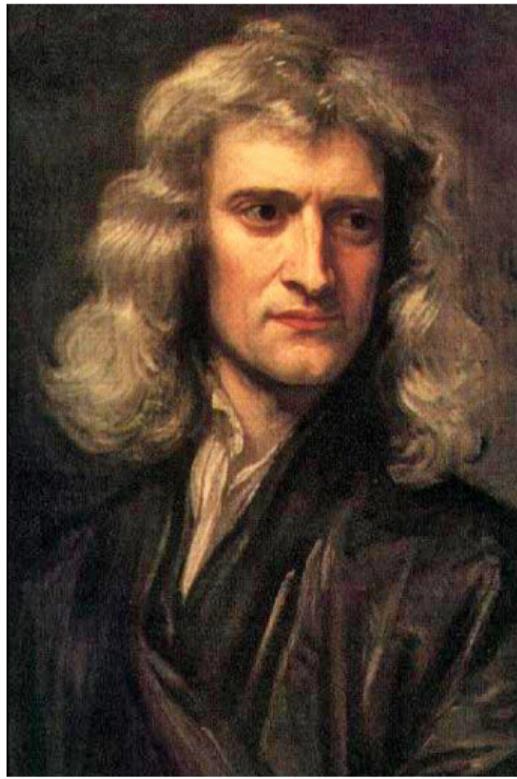
## 4 Data Preprocessing

## 5 Data Augmentation

## 6 Transfer Learning

## 7 References

## Motivation



“If I have seen further it is by standing on the shoulders of giants.”

**Sir Isaac Newton**  
scientist, born January 4, 1643  
(O.S. December 25, 1642)  
portrait by Sir Godfrey Kneller

Dobson's Improbable Quote of the Day

# Motivation

- You need a lot of data if you want to train/use CNNs
- Suppose you want to build an app to recognize flowers



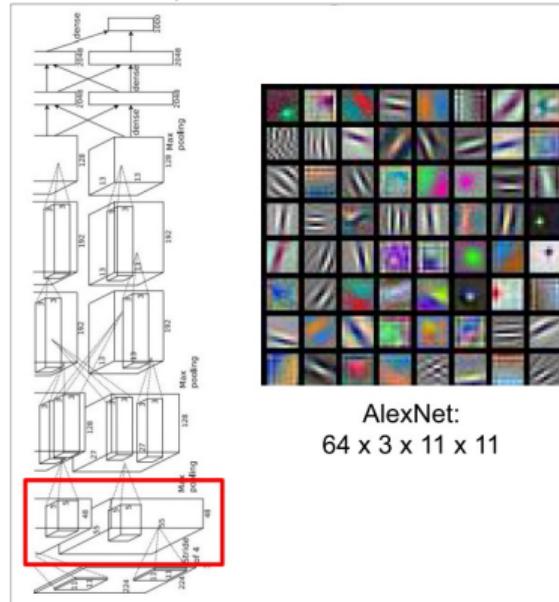
<https://www.robots.ox.ac.uk/~vgg/data/flowers/102>

# Transfer learning

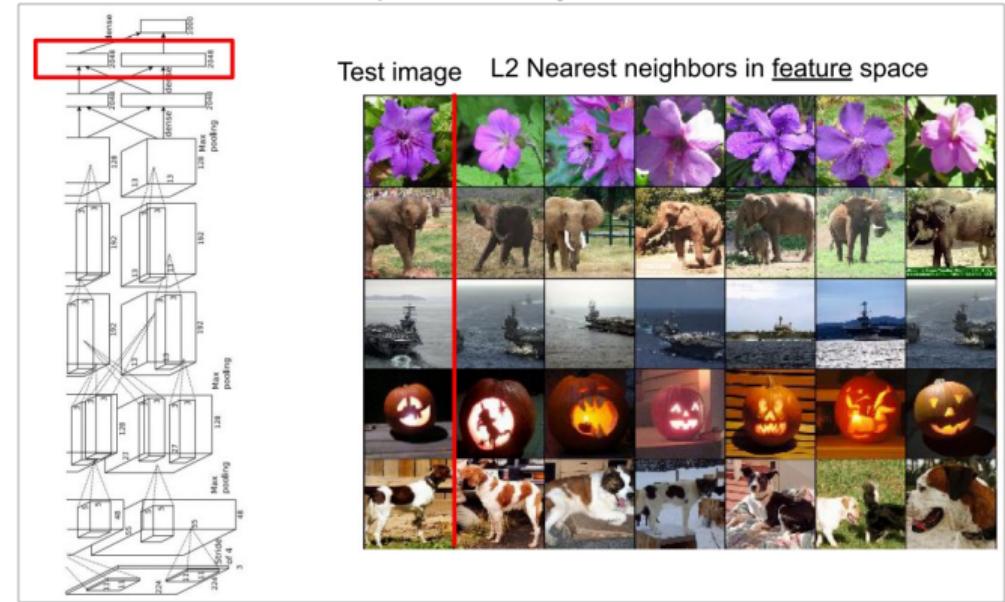
- However, by transfer learning you can use the learned features for a task (using a large amount of data) in another related task (for which you don't have enough data)
- **what should we train?**
- Option 1: Train only classification layer, freeze backbone
  - sometimes referred to as the linear evaluation protocol
  - Fast & simple
- Option 2: Train classification layer, fine-tune backbone at the same time
  - Slower, but can adapt feature extraction to dataset statistics

# Transfer learning: The idea

Earlier Layers → more local features



Final Layers → more global features

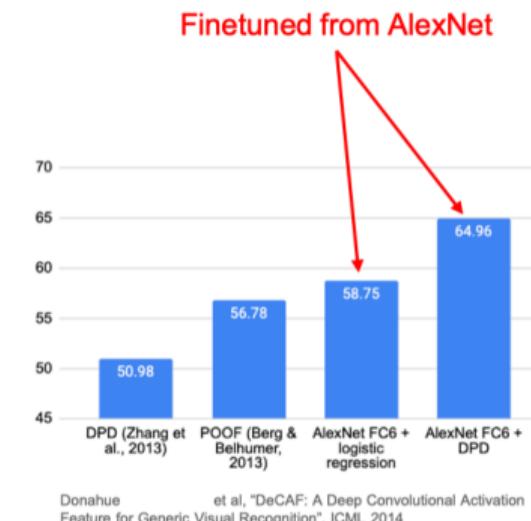
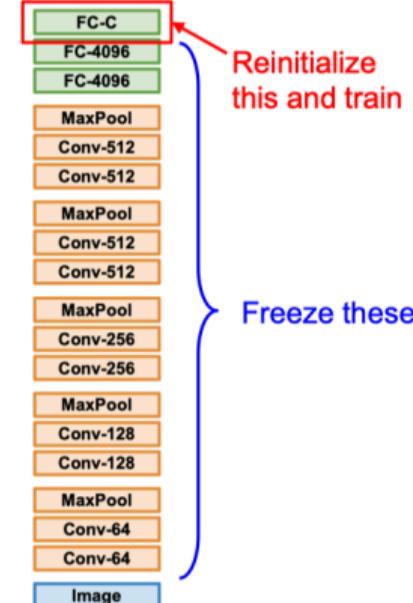


## Transfer Learning with CNNs

## 1. Train on Imagenet



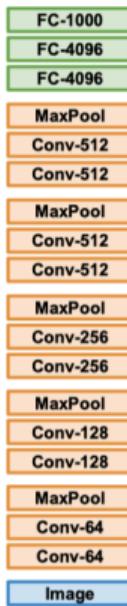
## 2. Small Dataset (C classes)



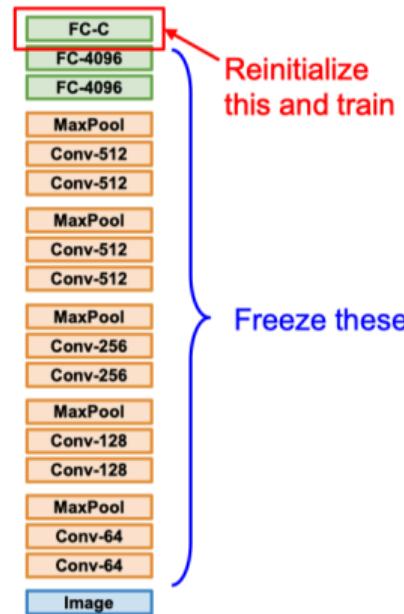
Donahue et al, DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, ICML 2014  
Razavian et al, CNN Features Off-the-Shelf: An Astounding Baseline for Recognition, CVPR Workshops 2014

# Transfer Learning with CNNs

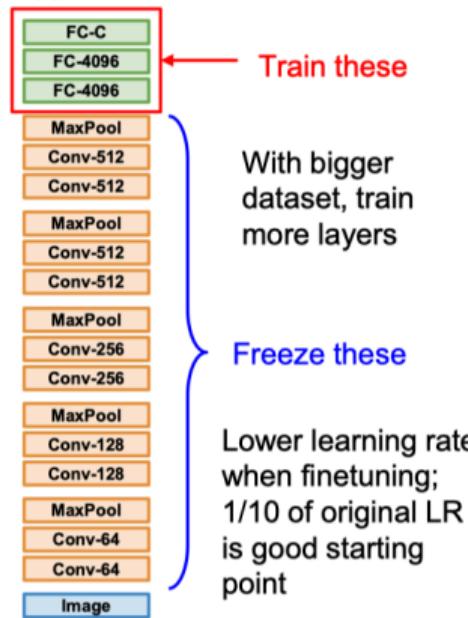
## 1. Train on Imagenet



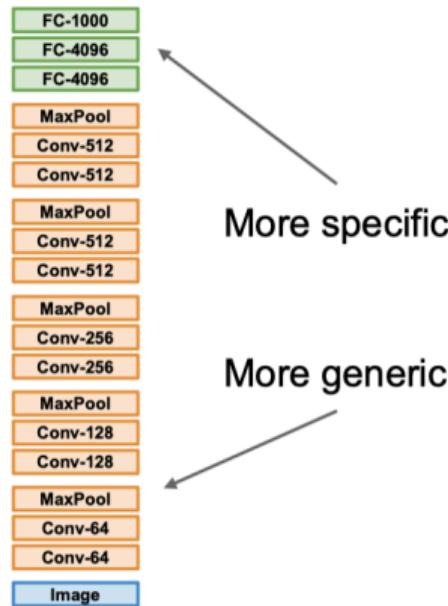
## 2. Small Dataset (C classes)



## 3. Bigger dataset



# Transfer Learning with CNNs

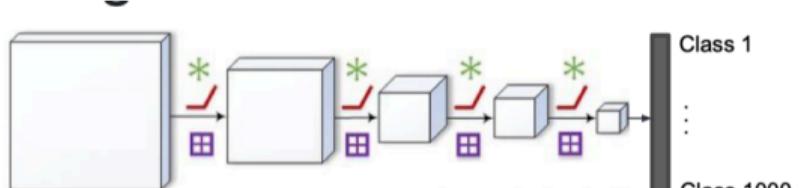


	<b>very similar dataset</b>	<b>very different dataset</b>
<b>very little data</b>	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
<b>quite a lot of data</b>	Finetune a few layers	Finetune a larger number of layers or start from scratch!

# Transfer learning with CNNs

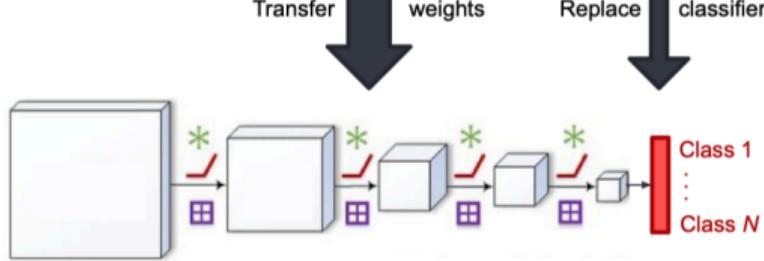
## 1 Pre-training (massive data)

IMAGENET  
1.2 million images  
1000 object classes



IMAGENET

## 2 Fine-tuning (much less data)



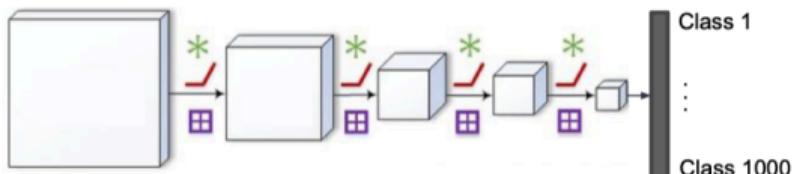
Recognize flowers



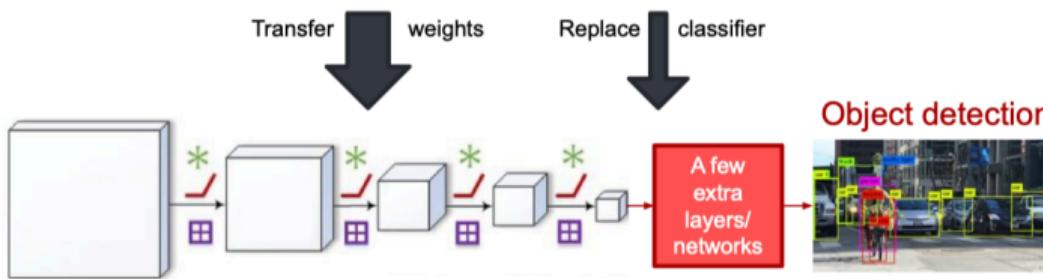
# Transfer learning: Domain Change

## 1 Pre-training (massive data)

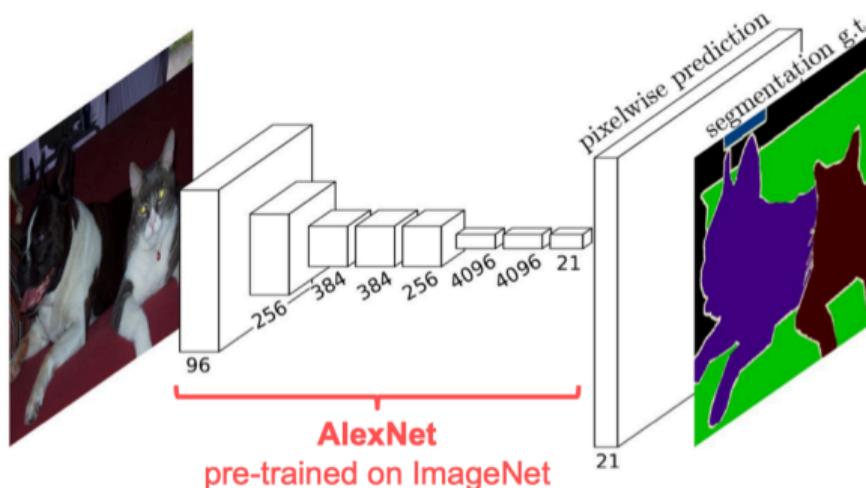
IMAGENET  
1.2 million images  
1000 object classes



## 2 Fine-tuning (much less data)

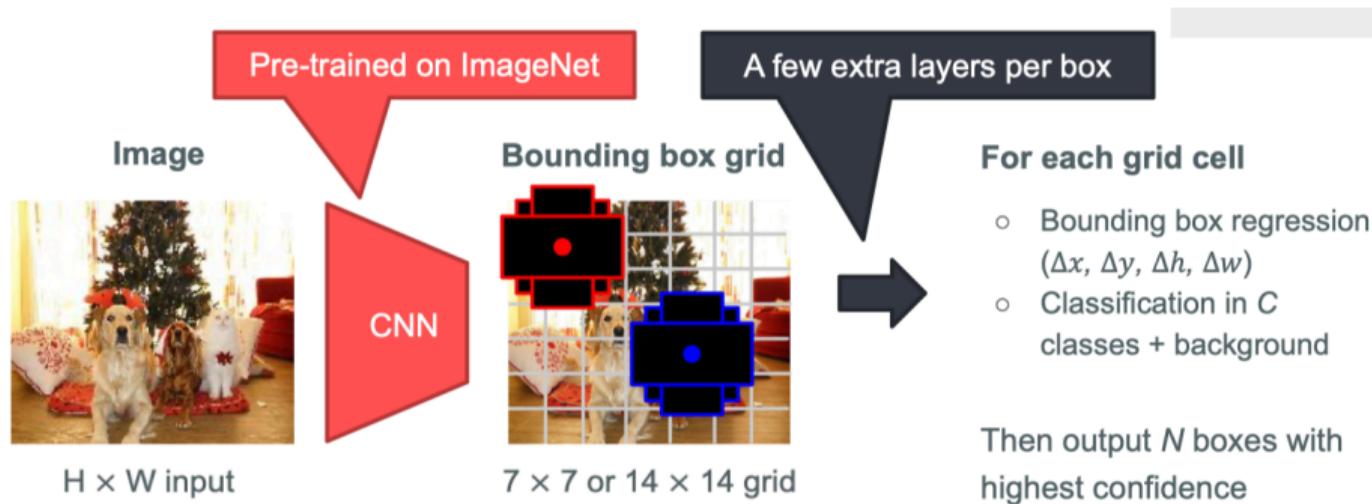


## Example: semantic segmentation



Long, Shelhamer, Darrell, CVPR 2015

# Example: Single-stage object detectors: SSD, YOLO,



Liu et al., ECCV 2016 | Redmon et al., CVPR 2016 | Lin et al., CVPR 2017

## Transfer learning with CNNs is pervasive

## Pose estimation

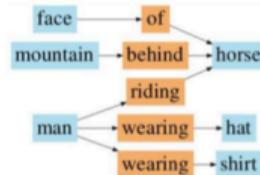
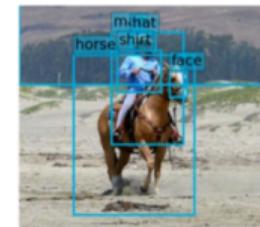


## Image captioning



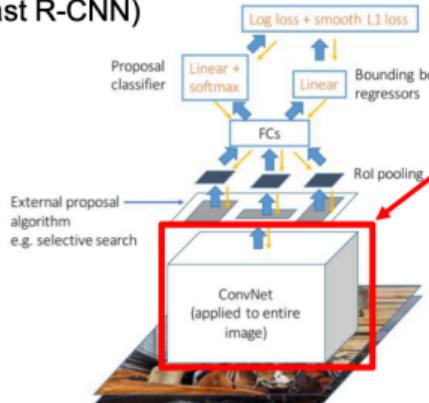
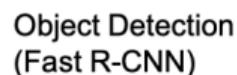
"two young girls are playing with  
lego toy."

## Scene graph prediction

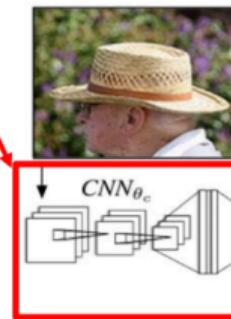


Many  
many  
more

## Transfer learning with CNNs is pervasive



## CNN pretrained on ImageNet



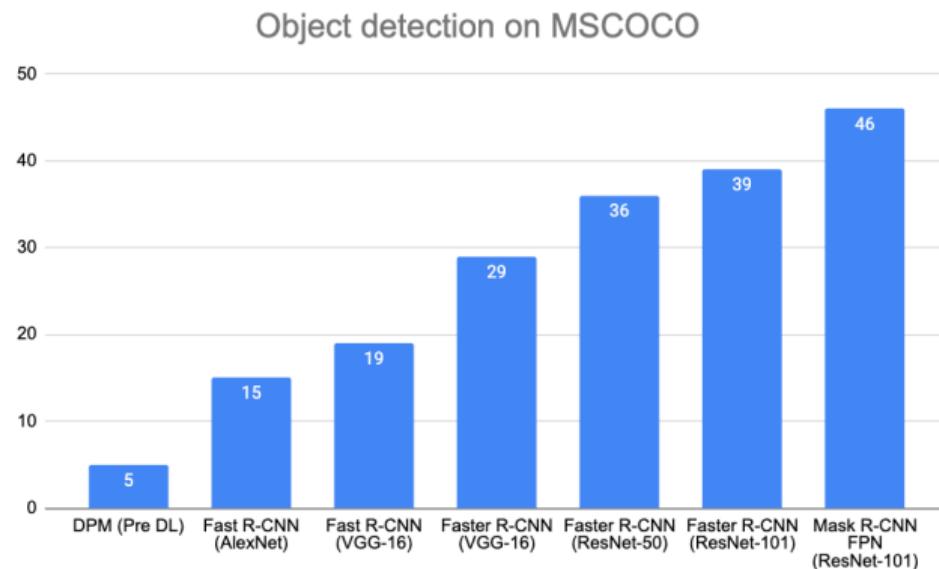
## Word vectors pretrained with word2vec

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

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

Girshick, Fast R-CNN, ICCV 2015 Figure copyright Ross Girshick, 2015. Reproduced with permission., Karpathy and Fei-Fei, Deep Visual-Semantic Alignments for Generating Image Descriptions, CVPR 2015 Figure copyright IEEE, 2015. Reproduced for educational purposes.

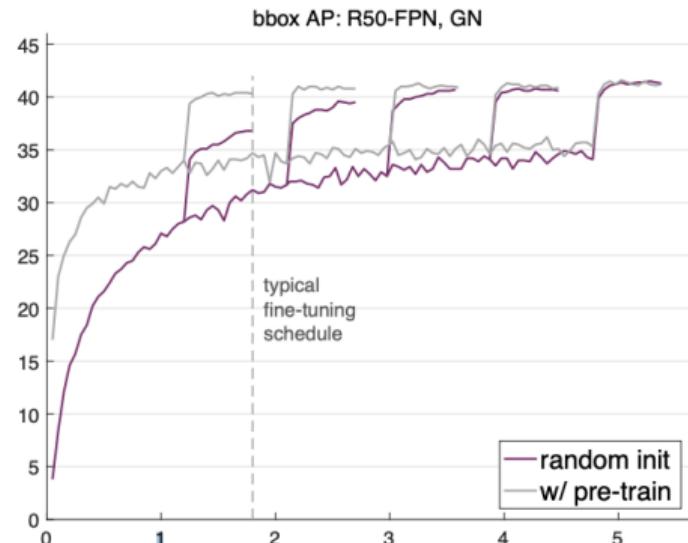
# Architecture matters



Girshick, The Generalized R-CNN Framework for Object Detection, ICCV 2017

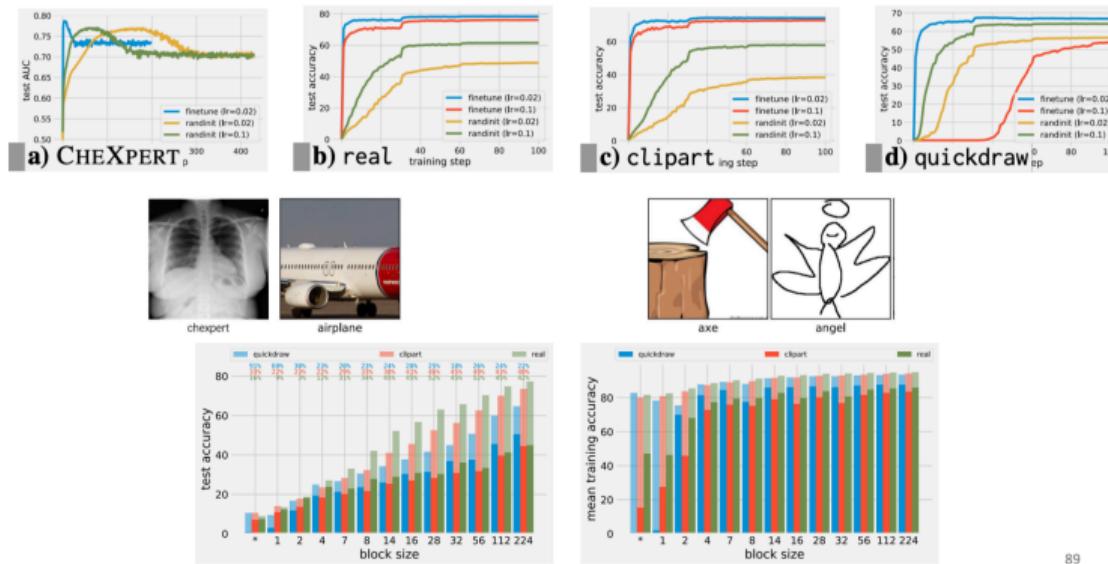
Transfer Learning might not always be necessary!

- Training from scratch can work just as well as training from a pretrained ImageNet model for object detection
  - But it takes 2-3x as long to train.
  - They also find that collecting more data is better than finetuning on a related task



He et al, Rethinking ImageNet Pre-training,  
ICCV 2019 Figure copyright Kaiming He,  
2019. Reproduced with permission.

# What is being transferred?

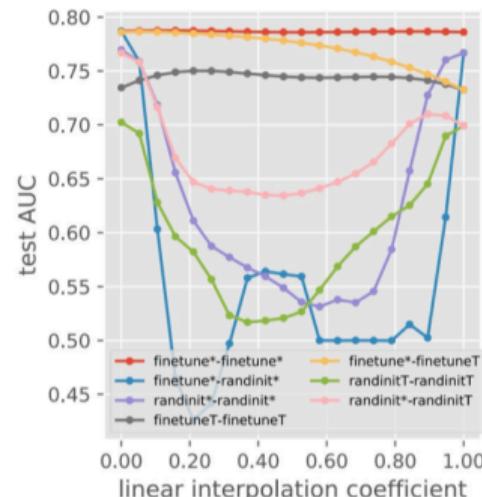
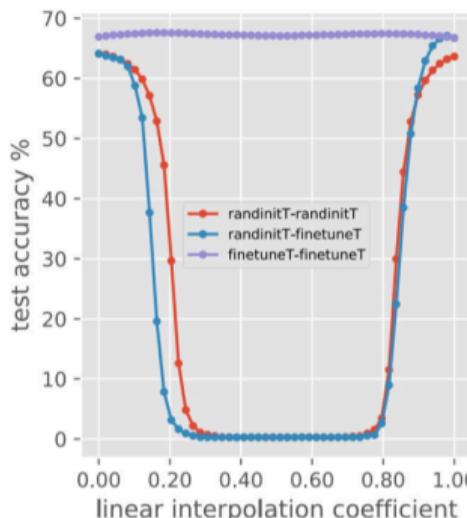


89

Neyshabour et al, What is being transferred in transfer learning?, NeurIPS 2020

## Performance barrier between different solutions

- There is no performance barriers between finetune models
  - Finetune models reside in the same basin
  - RandInits end up in a different basin, even for the same random seed



He et al, Rethinking ImageNet Pre-training, ICCV 2019 Figure: Kaiming He, 2019

## Takehome message

- Have some dataset of interest but it has  $< \sim 1M$  images?
  - ① Find a very large dataset that has similar data, train a big ConvNet there
  - ② Transfer learn to your dataset
- Deep learning frameworks provide a "Model Zoo" of pretrained models so you dont need to train your own
  - **Pytorch:** <https://github.com/pytorch/vision>
  - **TensorFlow:** <https://github.com/tensorflow/models>

## 1 CNNs in Vision

## 2 AlexNet

## 3 ResNet

## 4 Data Preprocessing

## 5 Data Augmentation

## 6 Transfer Learning

## 7 References

- [1] Fei Fei Li, “Stanford University cs231n.” Lecture slides.
- [2] M. Soleymani Baghshah, “Machine learning.” Lecture slides.
- [3] B. Raj, “Carnegie Mellon University 11-785.” Lecture slides.
- [4] Alexander Ecker, “Neuromatch Academy.” Lecture slides.

# Any Questions?