Artificial Neural Network
and Deep Learning
Lecture 6

Regularization
CNN Architectures

Agenda

- **Loss Function**
- **Hyperparameters**
- **Regularization** for good generalization
  - Dropout
  - Data augmentation
  - DropConnect
  - Reduce the number of parameters
  - Weight decay
- CNN Applications
  - Object Classification
- Different Dataset for Object Recognition.
- Different CNN Architectures for Object recognition
  - AlexNet, ZFNet, VGGNet, GoogLeNet, ResNet

Important Components of Neural Network apart from the neurons

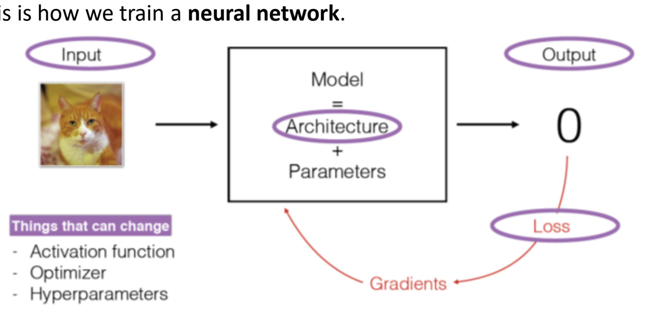
- **Activation functions**. Transforms the sum of weights and biases of each layer – adds non-linearity to the model.
- **Loss function (cost function, objective function, error function)**. Measures how well the NN reproduces the experimental training data.
- **Optimization algorithm**. Finds weights and bias values that minimize (locally) the Loss function.

Deep learning neural networks are trained using the stochastic gradient descent optimization algorithm.

- **Hyperparameters.** It are some setting that is difficult to optimize (LR, Momentum term, # of hidden layers, etc). Settled at first. No training for them.
- **Regulation techniques**. Prevents over-fitting of the NN to the training data.

Loss Function

- A **loss function** tells how good our current classifier is.
- The **loss** is **calculated** using **loss** function by matching the target(actual) value and predicted value by a **neural network**.
- Then we use the gradient descent method to update the weights of the **neural network** such that the **loss** is minimized. This is how we train a **neural network**.



- The loss function used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation.

The choice of Loss Function

- Regression Loss Functions
  - Mean Squared Error Loss
  - Mean Squared Logarithmic Error Loss
  - Mean Absolute Error Loss
- Binary Classification Loss Functions
  - Binary Cross-Entropy
  - Hinge Loss
  - Squared Hinge Loss
- Multi-Class Classification Loss Functions
  - Multi-Class Cross-Entropy Loss
  - Sparse Multiclass Cross-Entropy Loss
  - Kullback Leibler Divergence Loss

Cross-entropy and mean squared error are the two main types of loss functions to use when training neural network models.

Reference: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

Hyperparameters

- It are some setting that is difficult to optimize.

i.e. It is not appropriate to learn that on the training set.

- Examples:
  - Network architecture
  - Learning rate
  - Filter size for convolution layer

Regularization for Good Generalization

Regularization

- Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

i.e. any method that prevent over-fitting or help the optimization.

Under- and Over-fitting

- are factors determining how well an ML algorithm will perform. i.e. its ability to:

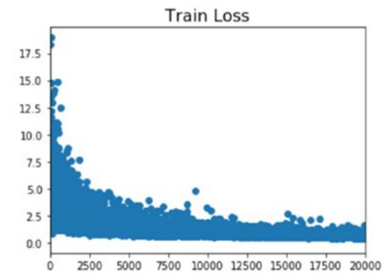
1. Make the training error small
2. Make gap between training and test errors small

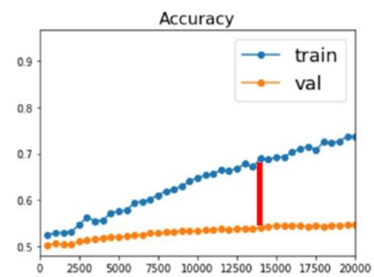
- Underfitting

Inability to obtain low enough error rate on the training set.

- Overfitting

Gap between training error and testing error is too large





Source: Fei-Fei Li & Justin Johnson & Serena Yeung 2019

Regularization: Add term to loss

It is any method that <u>prevent over-fitting</u> or <u>help the optimization</u>. This done by using additional terms in the training optimization objective.

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

λ = regularization strength (hyperparameter)

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Simple examples

<u>L2 regularization:</u> $R(W) = \sum_k \sum_l W_{k,l}^2$ (Weight decay)

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

More complex:

Dropout

Batch normalization

Stochastic depth, fractional pooling, etc

Regularization Strategies

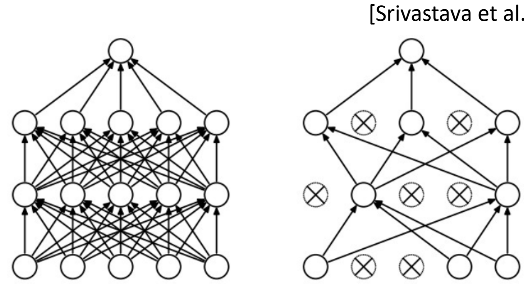
1. Parameter Norm Penalties
– (L2- and L1- regularization)
2. Norm Penalties as Constrained Optimization
3. Regularization and Under-constrained Problems
4. Data Set Augmentation
5. Noise Robustness
6. Semi-supervised learning
7. Multi-task learning
8. Early Stopping
9. Parameter tying and parameter sharing
10. Sparse representations
11. Bagging and other ensemble methods
12. Dropout
13. Adversarial training
14. Tangent methods

The best-performing models on most benchmarks use some or all of these tricks.

Regularization: Dropout

“randomly set some neurons to zero”

[Srivastava et al.]



(a) Standard Neural Net (b) After applying dropout.

- Dropout is a technique used to improve over-fit on neural networks.
- Randomly drop units (along with their connections) during training.
- Probability of dropping is a hyperparameter; 0.5 is common.
- Technique proposed by:

Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of machine learning research (2014).

Regularization: Dropout, cont.

See : http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

Dropout was used for training of fully connected layers.

Training:

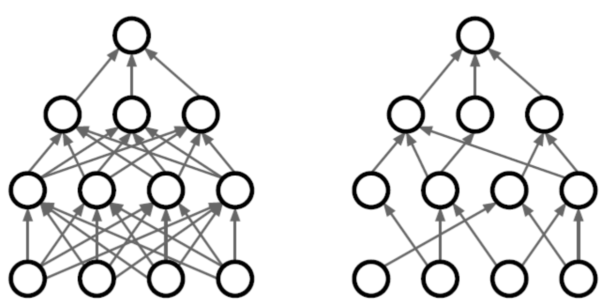
- Setting to 0 the output of each hidden neuron with probability 0.5 (50%).
- The neurons which are "dropped out" in this way
  - do not contribute to the forward pass
  - and do not participate in back-propagation.
- So, every time an input is presented, the neural network samples a **different architecture**, but all these architectures share weights.

Test:

At test time, we use all the neurons.

Regularization: DropConnect

- Training: Drop connections between neurons (set weights to 0)
- Testing: Use all the connections.

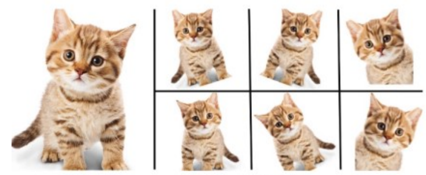


- Technique proposed by: Wan et al., "Regularization of Neural Networks using DropConnect", ICML 2013.

Regularization: Data Augmentation

(How to use Deep Learning when you have Limited Data for training?)

- The best way to improve generalization is to collect more data for training.
- We can augment the training data by transforming the examples. This is called data augmentation.
- Examples (for visual recognition)
  - translation
  - horizontal or vertical flip
  - rotation
  - smooth warping
  - noise (e.g. flip random pixels)
  - Padding
  - cropping

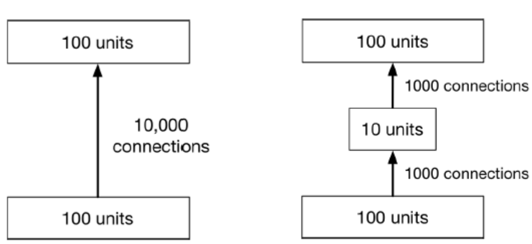


Enlarge your Dataset

- Only warp the training, not the test, examples.
- The choice of transformations depends on the task. (E.g. horizontal flip for object recognition, but not handwritten digit recognition.)

Reducing the Number of Parameters

- Can reduce the number of layers or the number of parameters per layer.
- Adding a linear bottleneck layer is a way to reduce the number of parameters:



Weight Decay

- Encouraging the weights to be small in magnitude.
- The **weight decay** is an additional term in the **weight** update rule that causes the **weights** to exponentially **decay** to zero.
- When training neural networks, it is common to use "**weight decay**," where after each update, the **weights are** multiplied by a factor slightly less than 1. This prevents the **weights** from growing too large.
- We regularize the cost function by change it to (adds a penalty equal to the sum of the squared value of the coefficients, this called **L2 regularization**)

$$\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^2$$

The regularization parameter λ determines how you trade off the original cost E with the large weights penalization.

Weight Decay, cont.

- The gradient descent update can be interpreted as weight decay:

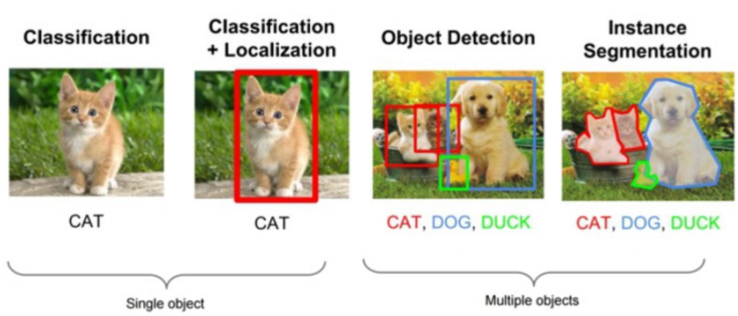
$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} - \eta\left(\frac{\partial E}{\partial \mathbf{w}} + \frac{\lambda}{2}\frac{\partial w^2}{\partial \mathbf{w}}\right) \\ &= \mathbf{w} - \eta\left(\frac{\partial E}{\partial \mathbf{w}} + \lambda \mathbf{w}\right) \\ &= (1 - \eta\lambda)\mathbf{w} - \eta\frac{\partial E}{\partial \mathbf{w}}\end{aligned}$$

The new term $-\eta\lambda w$ causes the weight to decay in proportion to its size.

when the regularization hyperparameter lambda increases, Weights are pushed toward becoming smaller (closer to 0).

CNN Applications

CNN Applications



Object Recognition

Object Recognition

Classification

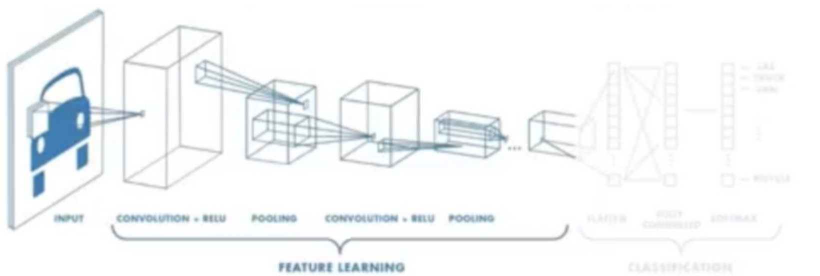
CAT

- Object recognition is the task of identifying which object category is present in an image.
- It's challenging because objects can differ widely in

  position, size, shape, appearance, etc.,

  and we have to deal with occlusions, lighting changes, etc.
- Object recognition can be either in either:
  - Direct applications to image search.
  - Closely related to object detection, the task of locating all instances of an object in an image
    - E.g., a self-driving car detecting pedestrians or stop signs.

CNNs for Recognition or Classification: Feature Learning



1. Learn features in input image through **convolution**.
2. Introduce **non-linearity** through activation function (real-world data is non-linear).
3. Reduce dimensionality and preserve spatial invariance with **pooling**.

MIT 6.S191, Introduction to Deep Learning, 2020.

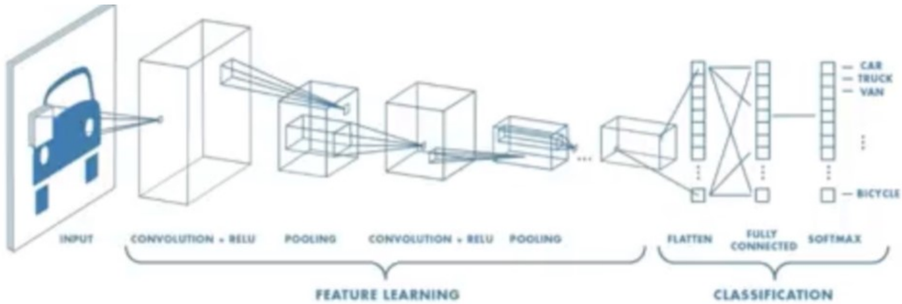
CNNs for Recognition or Classification: Class Probabilities

- CONV and POOL layers output high-level features of input.
- Fully connected layer uses these features for classifying input image.
- Express output as **probability** of image belonging to a particular class.

$$softmax(y_i) = \frac{e^{y_i}}{\sum_i e^{y_i}}$$

MIT 6.S191, Introduction to Deep Learning, 2020.

CNNs: Training with Backpropagation



Learn weights for convolutional filters and fully connected layers

Backpropagation: cross-entropy loss

$$\boldsymbol{L} = \sum_i \boldsymbol{y}^{(i)} \mathbf{log}(\widehat{\boldsymbol{y}}^{(i)})$$

Loss over the dataset is a sum of loss over examples

MIT 6.S191, Introduction to Deep Learning, 2020.

Recognition Datasets

Recognition Datasets

- In order to train and evaluate a machine learning system, we need to collect a dataset. The design of the dataset can have major implications.
- Some questions to consider:
  - Which categories to include?
  - Where should the images come from?
  - How many images to collect?
  - How to normalize (preprocess) the images?
- During the last two decades:
  - Datasets have gotten much larger (because of digital cameras and the Internet)
  - Computers got much faster
    - Graphics processing units (GPUs) turned out to be really good at training big neural nets; they're generally about 30 times faster than CPUs.

Recognition Datasets, cont.

- MNIST: Dataset of handwritten digits with 10 classes. 70k low resolution images (50Mb)

http://yann.lecun.com/exdb/mnist/

- CIFAR 10/100: Dataset with 60k low resolution images (10 and 100 classes respectively)

https://www.cs.toronto.edu/~kriz/cifar.html

- ImageNet: 14M images and more than 20k classes.

http://www.image-net.org/

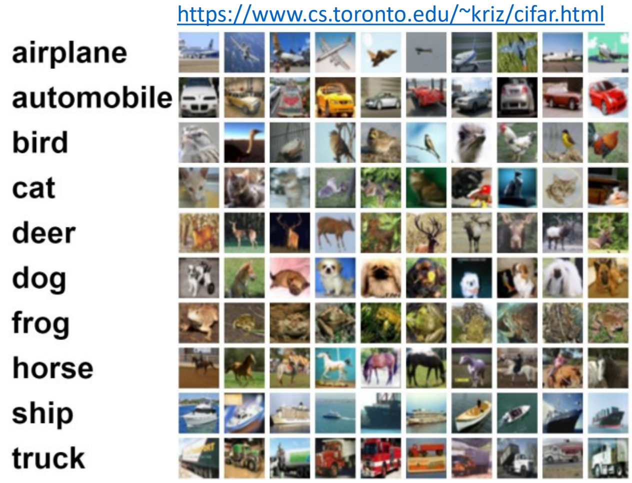
MNIST Dataset

- MNIST dataset of handwritten digits
  - **Categories**: 10 digit classes
  - **Source**: Scans of handwritten zip codes from envelopes
  - **Size**: 60,000 training images and 10,000 test images, grayscale, of size 28 x 28
  - **Normalization**: centered within in the image, scaled to a consistent size
    - The assumption is that the digit recognizer would be part of a larger pipeline that segments and normalizes images.
- In 1998, Yann LeCun and colleagues built a conv net called LeNet which was able to classify digits with 98.9% test accuracy.

CIFAR-10 Dataset

- It consists of 60,000 32x32 color images in 10 classes.
  - 50,000 training images
  - 10,000 testing images.

https://www.cs.toronto.edu/~kriz/cifar.html



ImageNet Dataset

- ImageNet is the modern object recognition benchmark dataset. It was introduced in 2009, and has led to amazing progress in object recognition since then.
- ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers

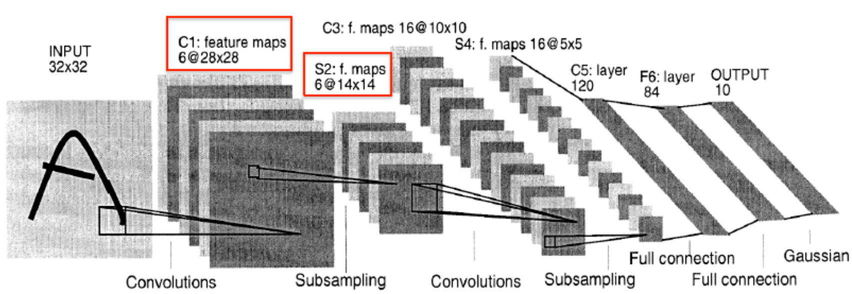
ImageNet, cont.

- Used for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2010 contest, an annual benchmark competition for object recognition.
- ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images.

Different CNN Architectures

LeNet

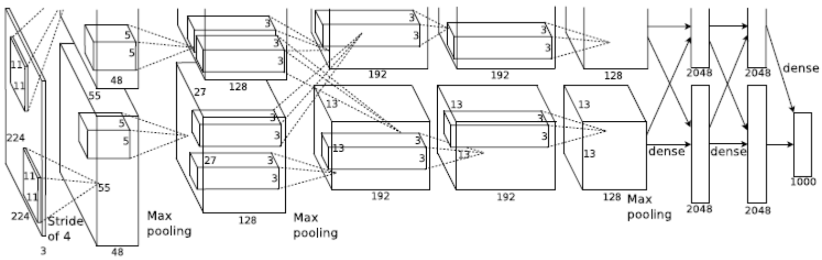
[LeCun et al., 1998]



- It was applied to handwritten digit recognition on MNIST in 1998.
- Conv filters were 5x5, applied at stride 1.
- Subsampling (Pooling) layers were 2x2 applied at stride 2
- i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

AlexNet

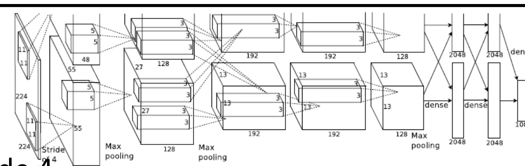
Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012.



(Krizhevsky et al., 2012)

- It contains 8 weight learned layers (5 convolutional and 3 fully-connected).
- **Architecture:** [CONV1, MAX POOL1, NORM1, CONV2, MAX POOL2, NORM2, CONV3, CONV4, CONV5, Max POOL3, FC6, FC7, FC8]
- They used lots of tricks (ReLU units, weight decay, data augmentation, stochastic gradient descent (SGD) on training with momentum, dropout).
- AlexNet achieved 16.4% top-5 error (i.e. the network gets 5 tries to guess the right category).

AlexNet



(Krizhevsky et al., 2012)

Input: 227x227x3 images
First layer (CONV1): 96 11x11 filters applied at stride 4
=>

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

Output volume **[55x55x96]**

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

Parameters: (11*11*3)*96 = **35K**

Input: 227x227x3 images
After CONV1: 55x55x96
Second layer (POOL1): 3x3 filters applied at stride 2

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

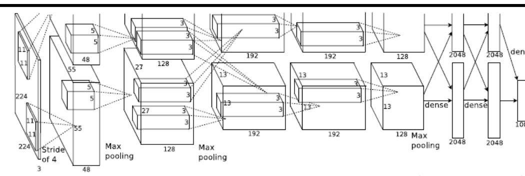
Output volume: 27x27x96

- Q: what is the number of parameters in this layer?

Parameters: 0!

Input: 227x227x3 images
After CONV1: 55x55x96
After POOL1: 27x27x96
...

AlexNet

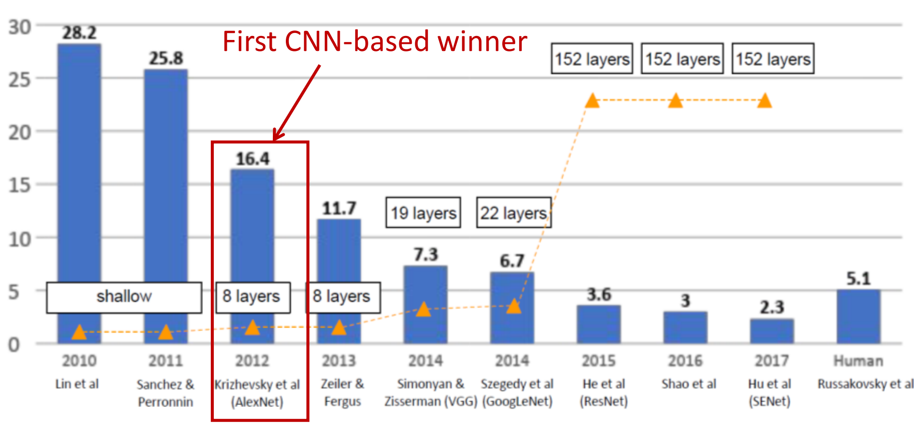


(Krizhevsky et al., 2012)

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

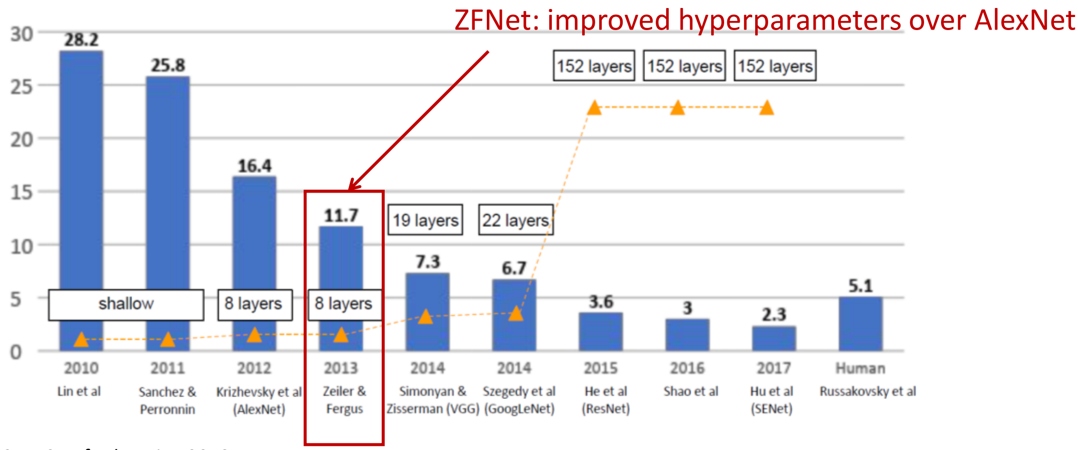
Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Reference: cs321n, Stanford, spring 2019

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

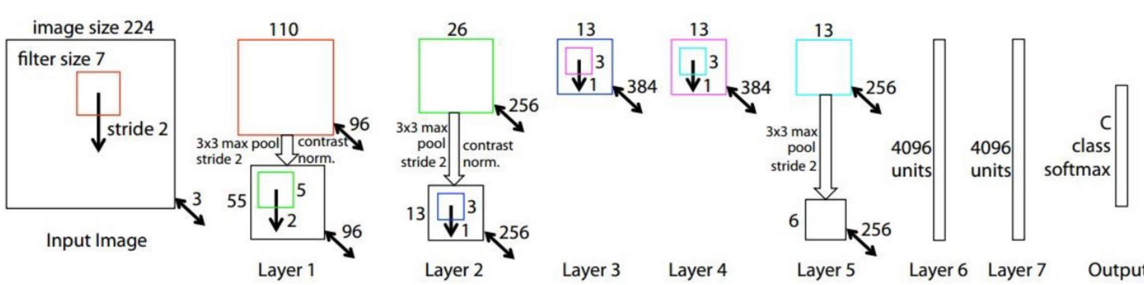


Reference: cs321n, Stanford, spring 2019

Reference: cs321n, Stanford, spring 2019

ZFNet

[Zeiler and Fergus, 2013]



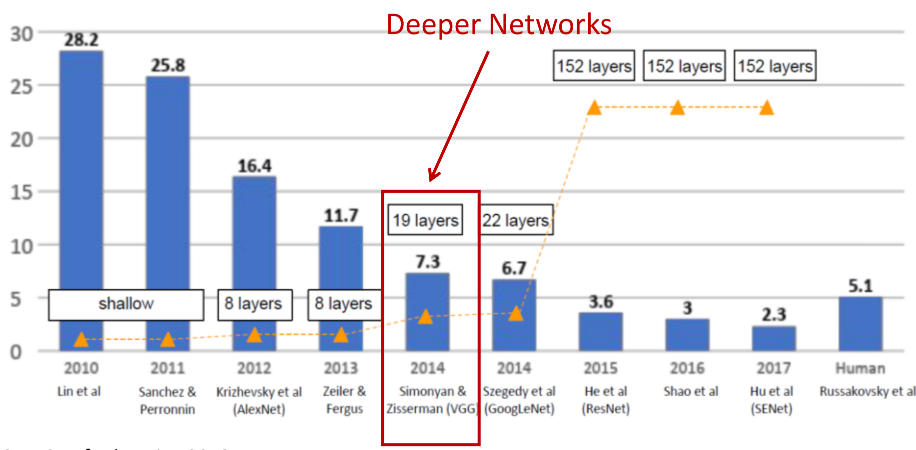
- It is AlexNet but:

CONV1: change from (11x11 stride 4) to (7x7 stride 2)

CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

- Top-5 error in ILSVRC'13: 11.7%

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Reference: cs321n, Stanford, spring 2019

VGGNet

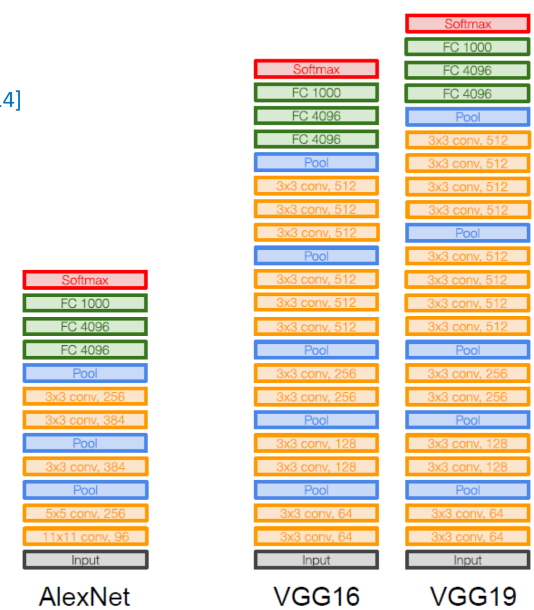
[K. Simonyan and A. Zisserman, University of Oxford, 2014]

- It is a Convolution Neural Network model.
- 16 – 19 layers.
- Only 3x3 CONV stride 1, pad 1

and 2x2 MAX POOL stride 2

- Top-5 error in ILSVRC'14:

7.3%



VGGNet

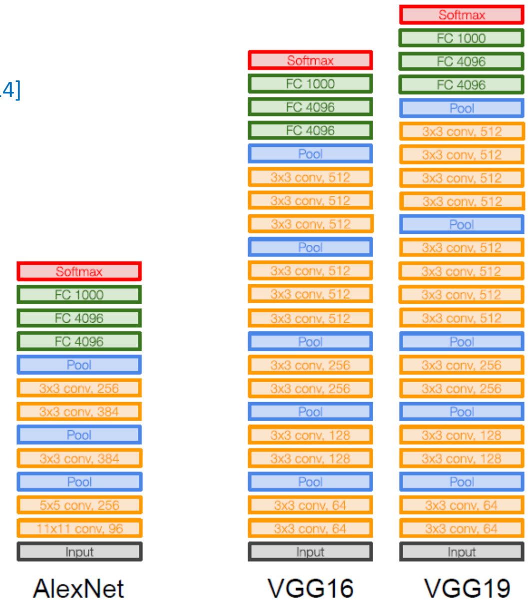
[K. Simonyan and A. Zisserman, University of Oxford, 2014]

Q: Why use smaller filters?

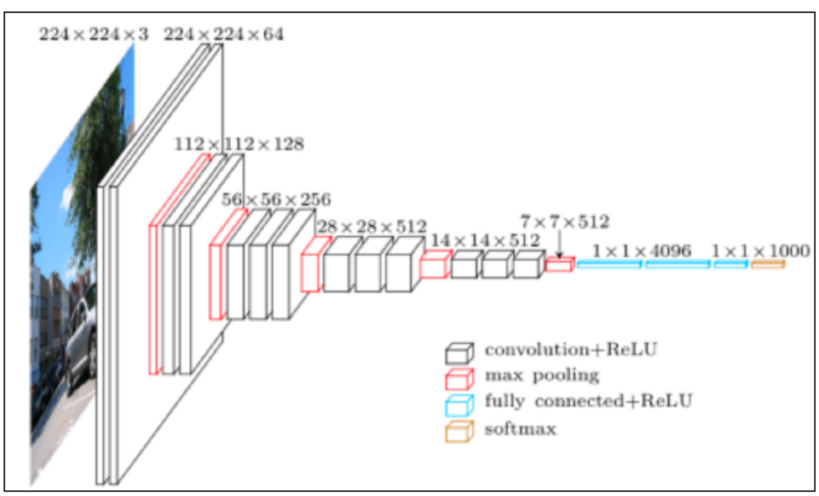
Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer.

But deeper, more non-linearities

And fewer parameters: 3 * (3^2C^2) vs. 7^2C^2 for C channels per layer



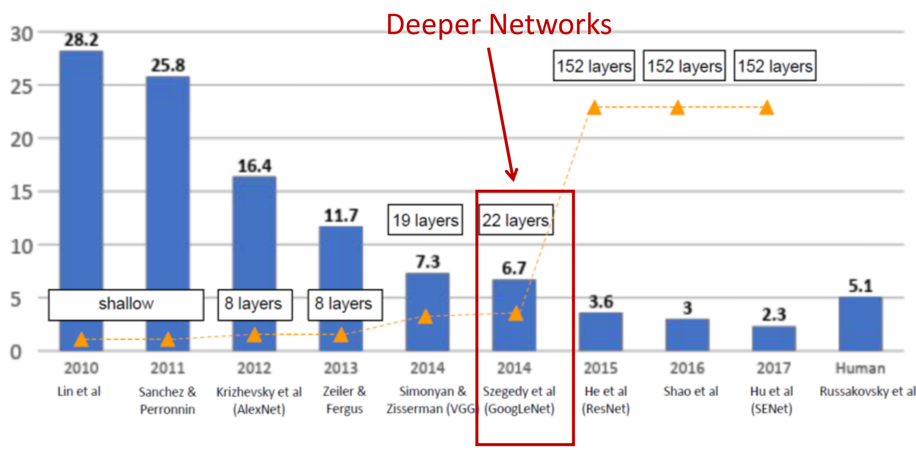
VGG 16



Source: https://www.cs.toronto.edu/~frossard/post/vgg16/

19

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

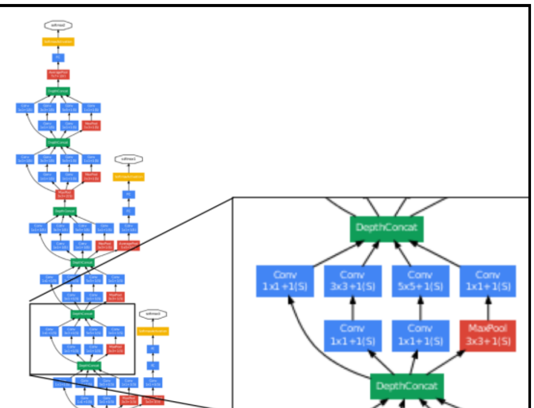


Reference: cs321n, Stanford, spring 2019

GoogLeNet

[Szegedy et al., 2014]

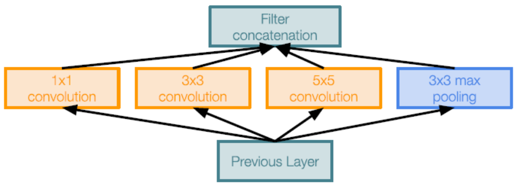
- 22 layers.
- No fully connected FC layers.
- Convolutions are broken down into a bunch of smaller convolutions (since this requires fewer parameters total)
- GoogLeNet has only 5 million parameters, compared with 60 million for AlexNet. 12x less than AlexNet
- Top-5 error in ILSVRC'14: 6.7% test error on ImageNet.



Inception module

"Inception module": design a good local network topology (network within a network) and then stack these modules on top of each other

GoogLeNet



Naive Inception module

- **Apply parallel filter operations on the input from previous layer:**

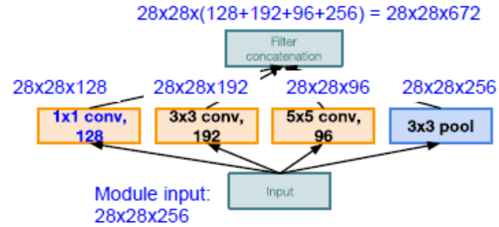
- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)

- **Pooling operation (3x3)**

Concatenate all filter outputs together depth-wise

Q: What is the problem with this?

Computational complexity



Naive Inception module

Q1: What is the output size of the 1x1 conv, with 128 filters?

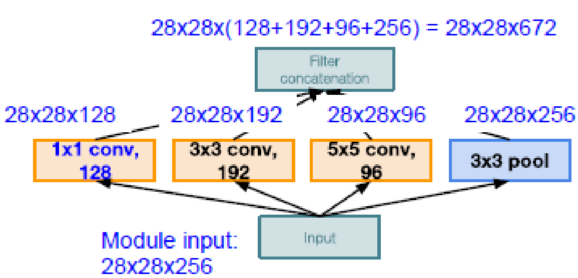
Q2: What are the output sizes of all different filter operations?

Q3:What is output size after filter concatenation?

GoogLeNet

Q: What is the problem with this?

Computational complexity



Naive Inception module

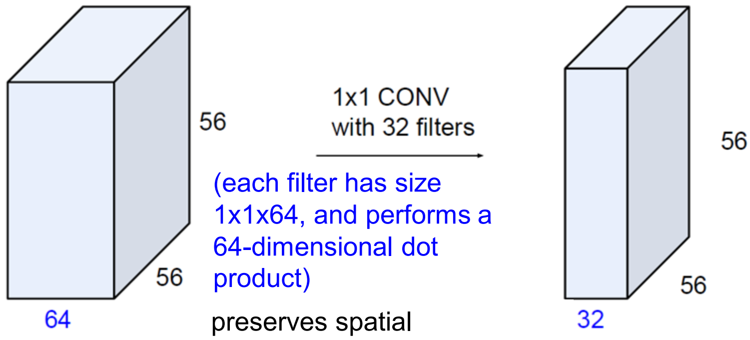
Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

Very expensive compute

Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!

Solution: “bottleneck” layers that use 1x1 convolutions to reduce feature depth

Reminder: 1x1 convolutions



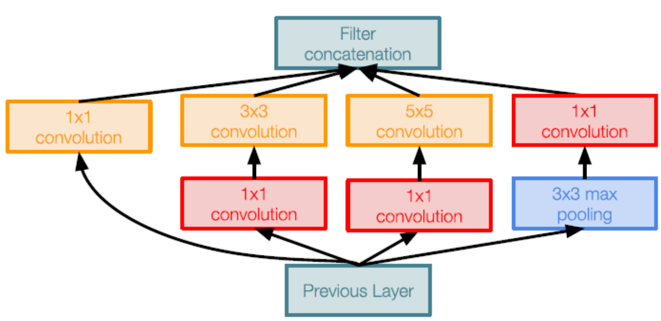
preserves spatial dimensions, reduces depth!

Projects depth to lower dimension (combination of feature maps)

GoogLeNet

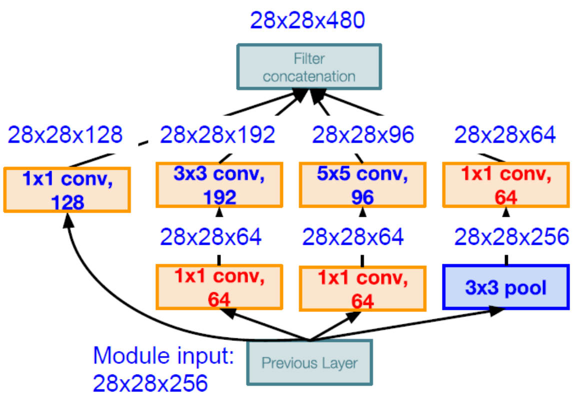
Solution: "bottleneck" layers that use 1x1 convolutions to reduce feature depth.

1x1 conv "bottleneck" layers



Inception module with dimension reduction

GoogLeNet



Inception module with dimension reduction

Using same parallel layers as naive example, and adding "1x1 conv, 64 filter" bottlenecks:

Conv Ops:

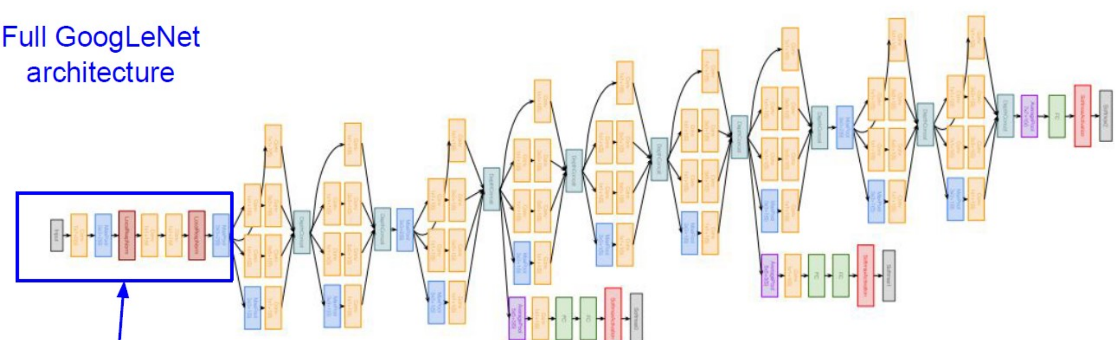
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256

Total: 358M ops

- Bottleneck can also reduce depth after pooling layer
- Compared to **854M ops** for naive version

GoogLeNet

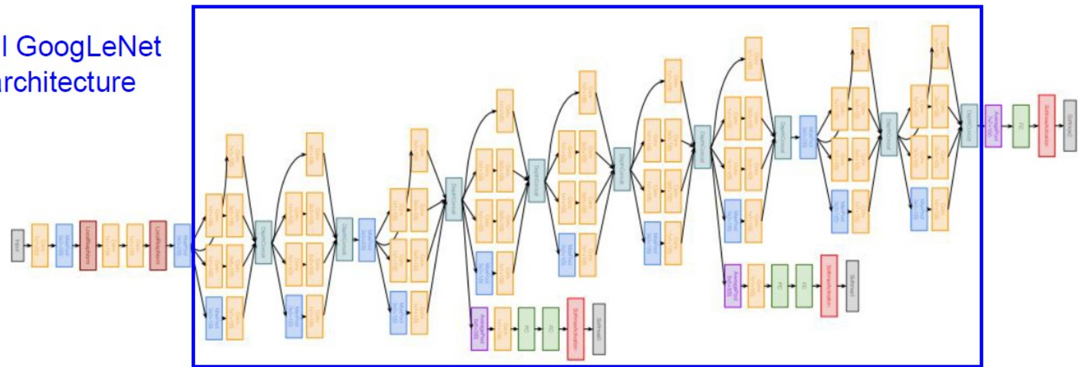
Full GoogLeNet architecture



Stem Network: Conv-Pool- 2x Conv-Pool

GoogLeNet

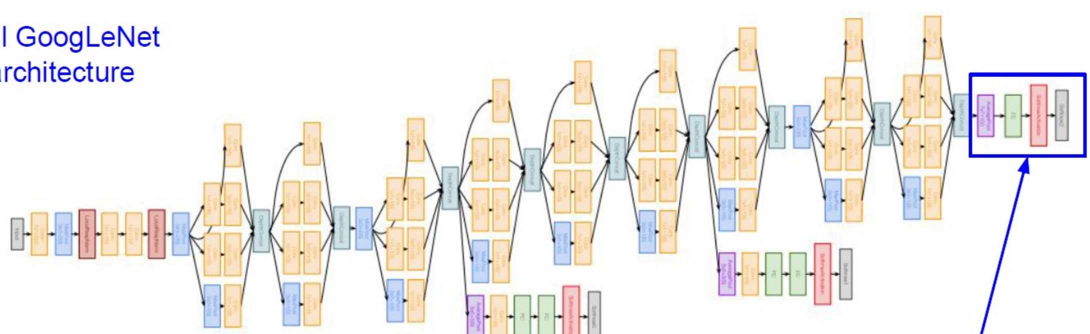
Full GoogLeNet architecture



Stacked Inception Modules

GoogLeNet

Full GoogLeNet architecture

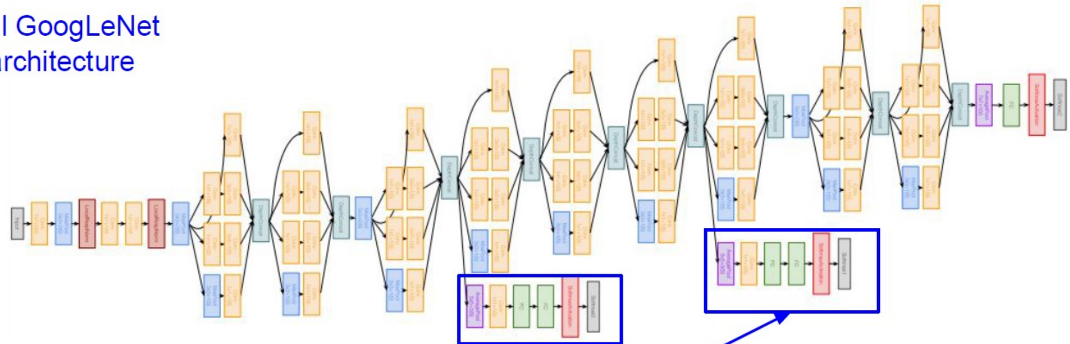


Note: after the last convolutional layer, a <u>global average pooling layer</u> is used that spatially averages across each feature map, before final FC layer. No longer multiple expensive FC layers!

Classifier output

GoogLeNet

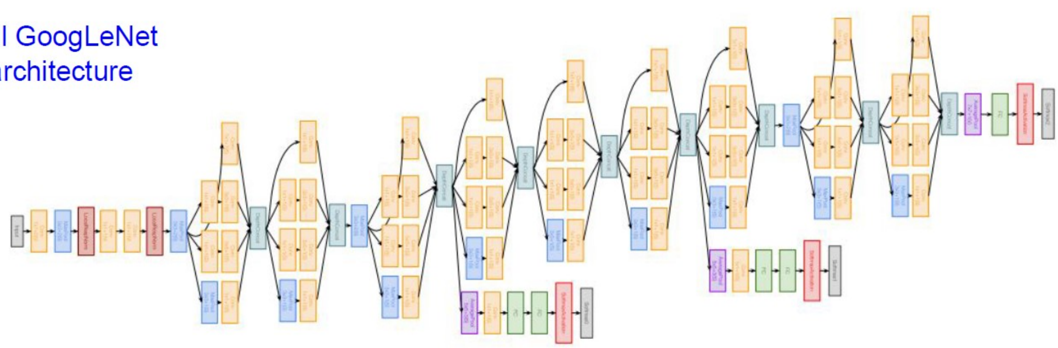
Full GoogLeNet architecture



Auxiliary classification outputs to inject additional gradient at lower layers (AvgPool-1x1Conv-FC-FC-Softmax)

GoogLeNet

Full GoogLeNet architecture



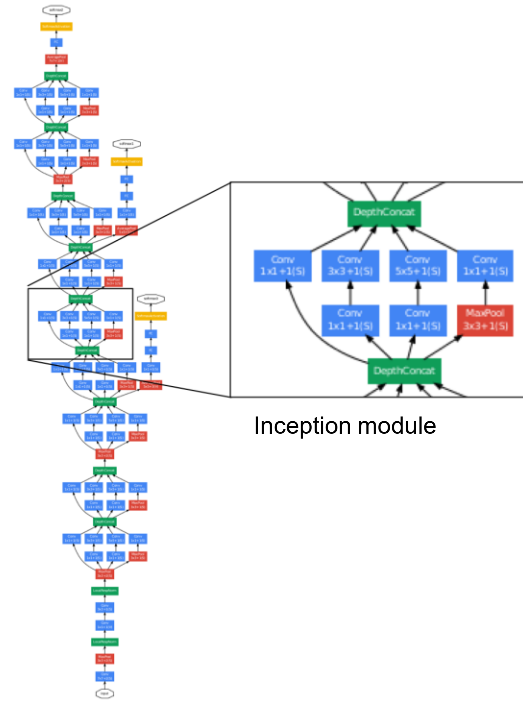
22 total layers with weights
(parallel layers count as 1 layer => 2 layers per Inception module. Don't count auxiliary output layers)

GoogLeNet

[Szegedy et al., 2014]

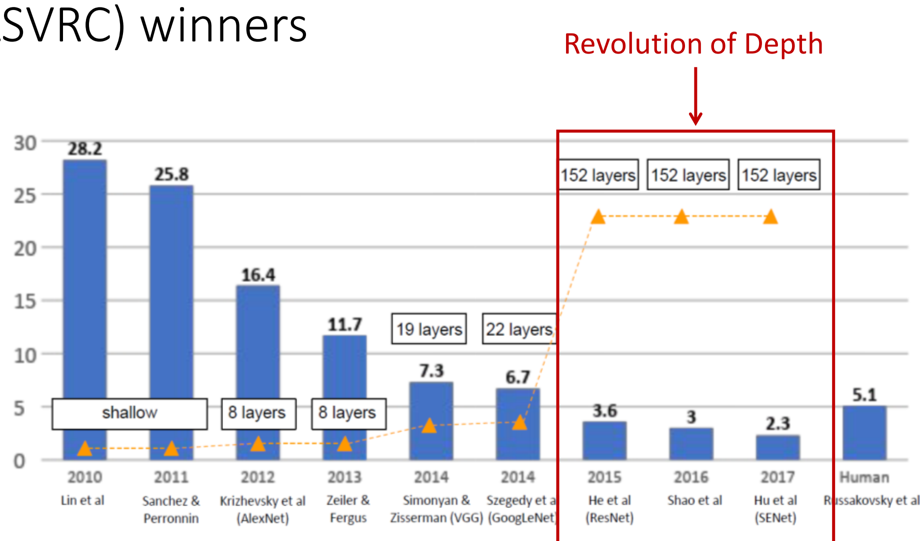
Deep networks, with computational efficiency.

- 22 layers.
- Efficient Inception module.
- Avoid expensive FC layers.
- 12x less parameters than AlexNet
- Top-5 error in ILSVRC'14: 6.7% test error on ImageNet.



Inception module

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



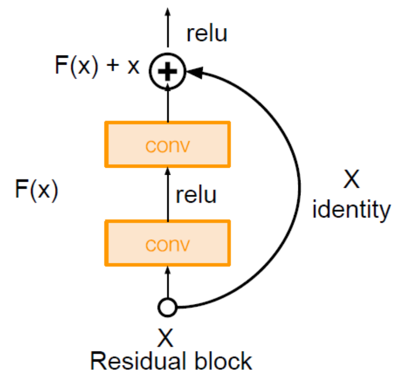
Reference: cs321n, Stanford, spring 2019

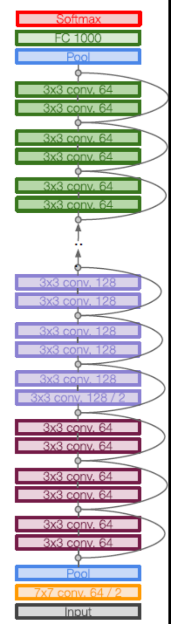
ResNet

[He et al., 2015]

Very deep networks using residual Connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!

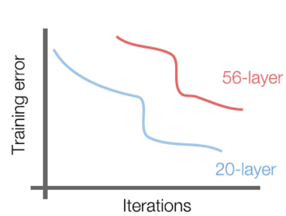


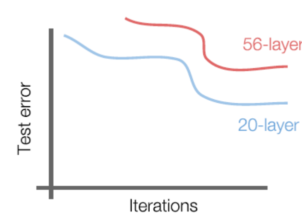


ResNet

[He et al., 2015]

- What happens when we continue stacking deeper layers on a "plain" convolutional neural network?





Q: What's strange about these training and test curves?

look at the order of the curves:

56-layer model performs worse on both training and test error

-> The deeper model performs worse, but it's not caused by overfitting!

ResNet

[He et al., 2015]

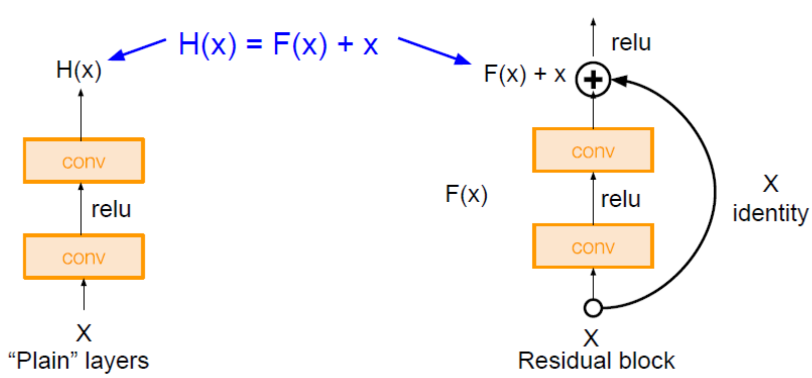
Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize.

- The deeper model should be able to perform at least as well 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

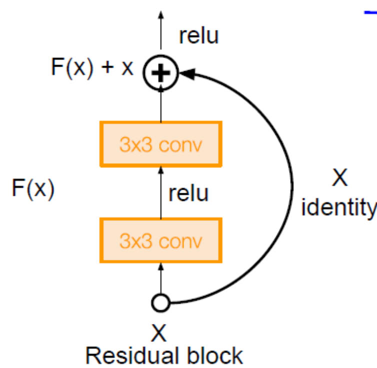


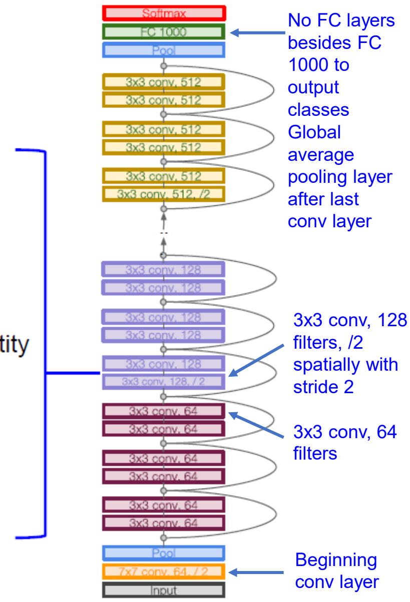
Use layers to fit residual F(x) = H(x) - x instead of H(x) directly

ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)

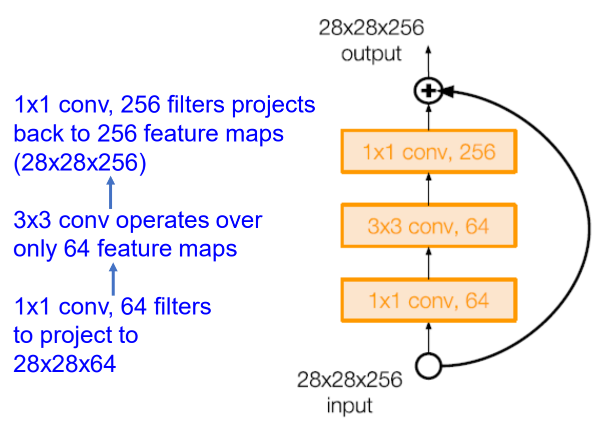




ResNet

[He et al., 2015]

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



ResNet

[He et al., 2015]

Training ResNet in practice:

- Batch Normalization after every CONV layer
- 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

ResNet

[He et al., 2015]

Experimental Results:

- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lowing 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)

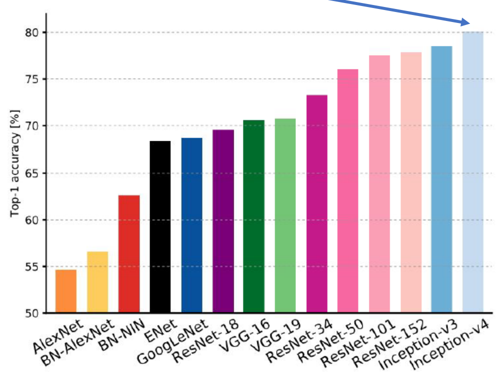
Comparing complexity...

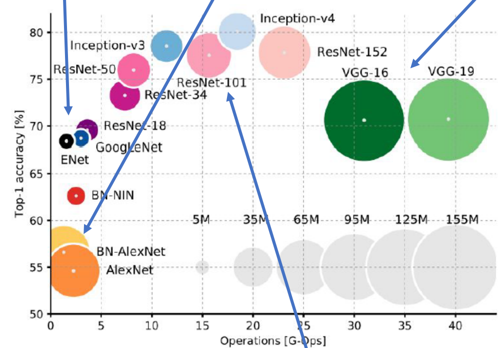
Inception-v4: Resnet + Inception!

GoogLeNet: Most efficient

AlexNet: Smaller compute, still memory heavy, lower accuracy

VGG: Highest memory, most operations

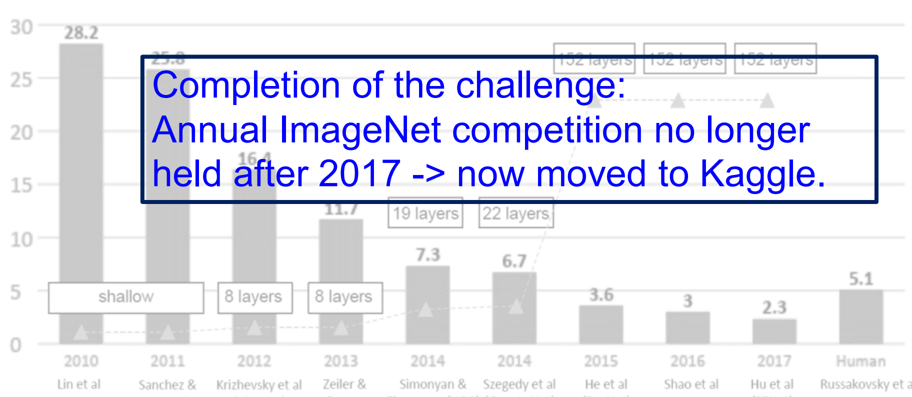




An Analysis of Deep Neural Network Models for Practical Applications, 2017.

ResNet: Moderate efficiency depending on model, highest accuracy

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Completion of the challenge:
Annual ImageNet competition no longer held after 2017 -> now moved to Kaggle.

Reference: cs321n, Stanford, spring 2019

Summary

- **Loss Function**
- **Hyperparameters**
- **Regularization** for good generalization
  - Dropout
  - Data augmentation
  - DropConnect
  - Reduce the number of parameters
  - Weight decay
- CNN Applications
  - Object Classification
- Different Dataset for Object Recognition.
- Different CNN Architectures for Object recognition
  - AlexNet, ZFNet, VGGNet, GoogLeNet, ResNet

Resources

1. Roger Grosse and Jimmy Ba, CSC421 /2516 winter 2019 Neural Network and Deep Learning, http://www.cs.toronto.edu.
2. Related Lecture from CS231n @ Stanford. http://cs231n.stanford.edu/
3. MIT 6.S191, Introduction to Deep Learning, 2020.

Thanks
for your attention…