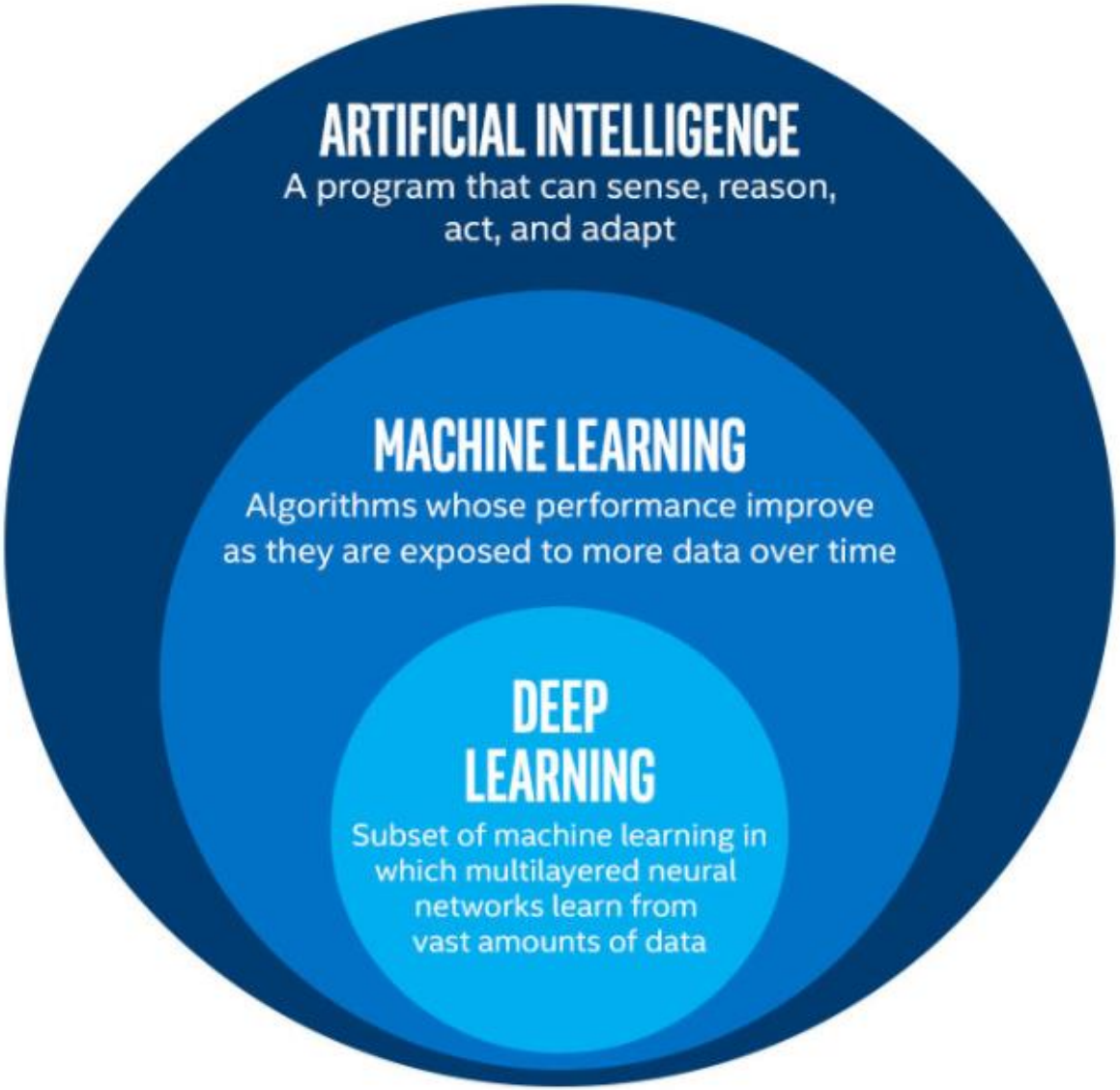
CAP 5516
Medical Image Computing
(Spring 2025)

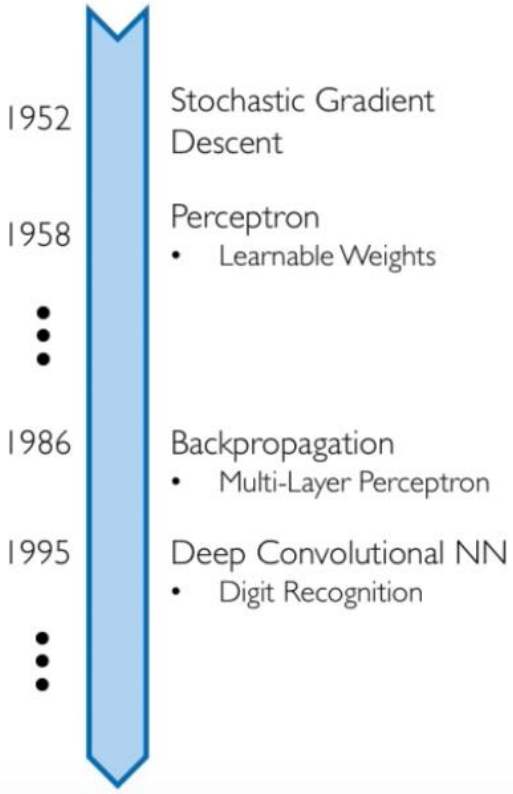
Dr. Chen Chen
Associate Professor
Center for Research in Computer Vision (CRCV)
University of Central Florida
Office: HEC 221
Email: chen.chen@crcv.ucf.edu
Web: https://www.crcv.ucf.edu/chenchen/

Lecture 6: Introduction to Deep Learning (1)



https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55

Why Now?



Neural Networks date back decades, so why the resurgence?

1. Big Data

- Larger Datasets
- Easier Collection & Storage

IMAGENET

WIKIPEDIA
The Free Encyclopedia

2. Hardware

- Graphics Processing Units (GPUs)
- Massively Parallelizable

3. Software

- Improved Techniques
- New Models
- Toolboxes

TensorFlow

PYTORCH

Caffe2

Chainer

mxnet

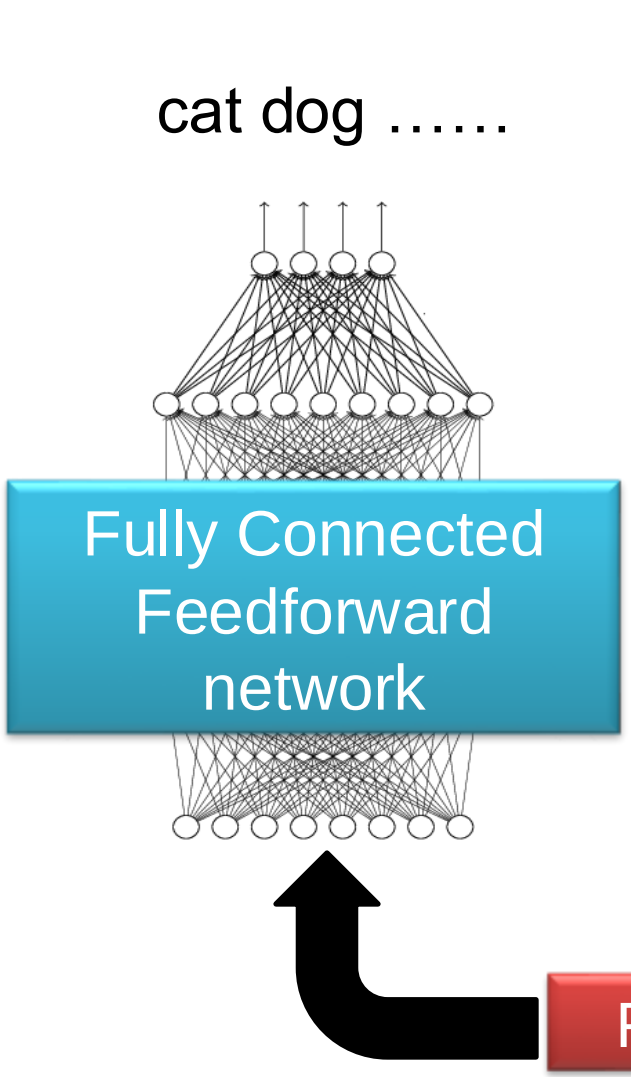
4. Open resource (GitHub)

Credit: Lex Fridman

Convolutional Neural Network (CNN)

Widely used in image processing and computer vision

The whole CNN



The whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

CNN – Convolution

Those are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter/kernel 1

Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

⋮ ⋮

Each filter detects a small pattern (3 x 3).

CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter/Kernel 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -1

CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

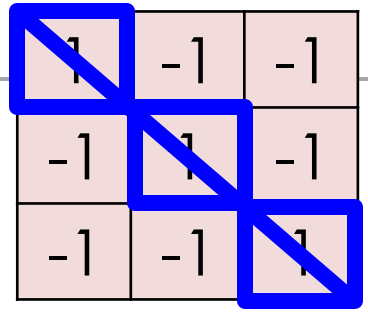
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -3

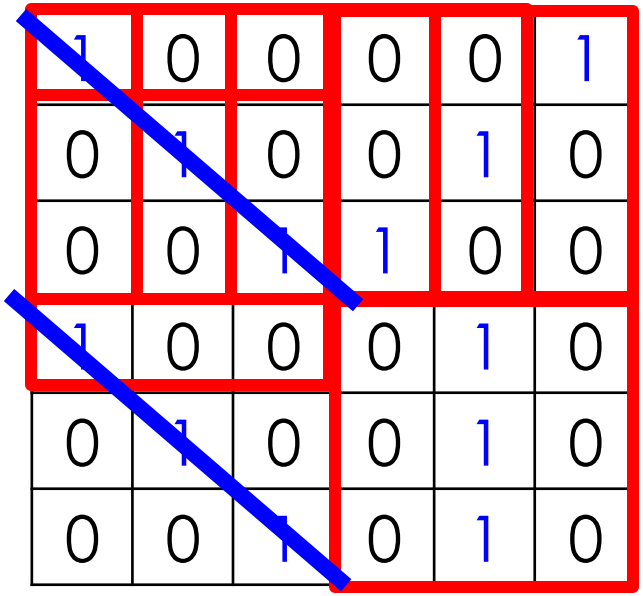
We set stride=1 below

CNN – Convolution

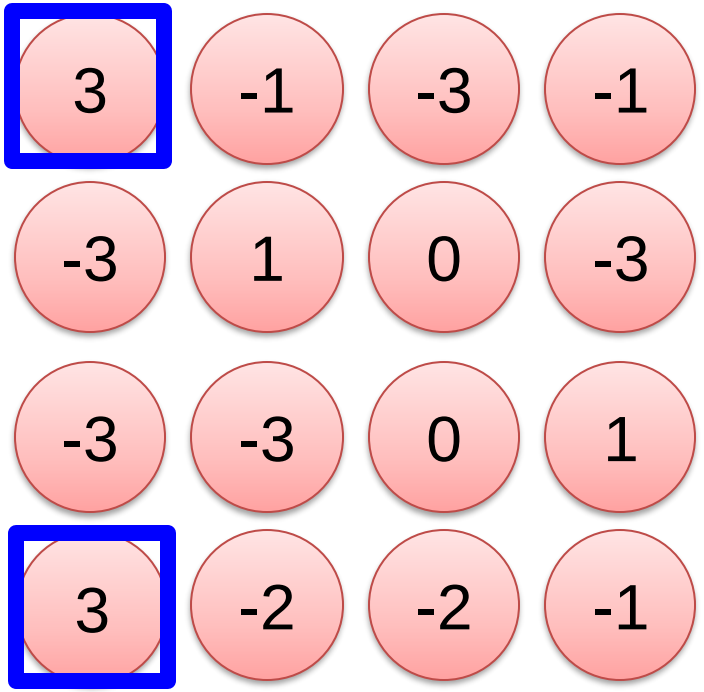


Filter 1

stride=1



6 x 6 image



Property 2

CNN – Convolution

-1	1	-1
-1	1	-1
-1	1	-1

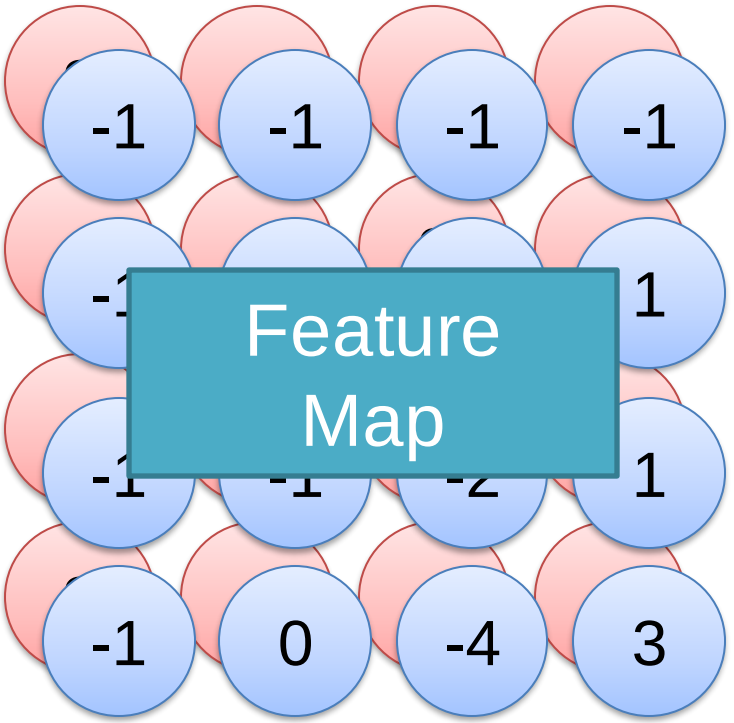
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Do the same process for every filter

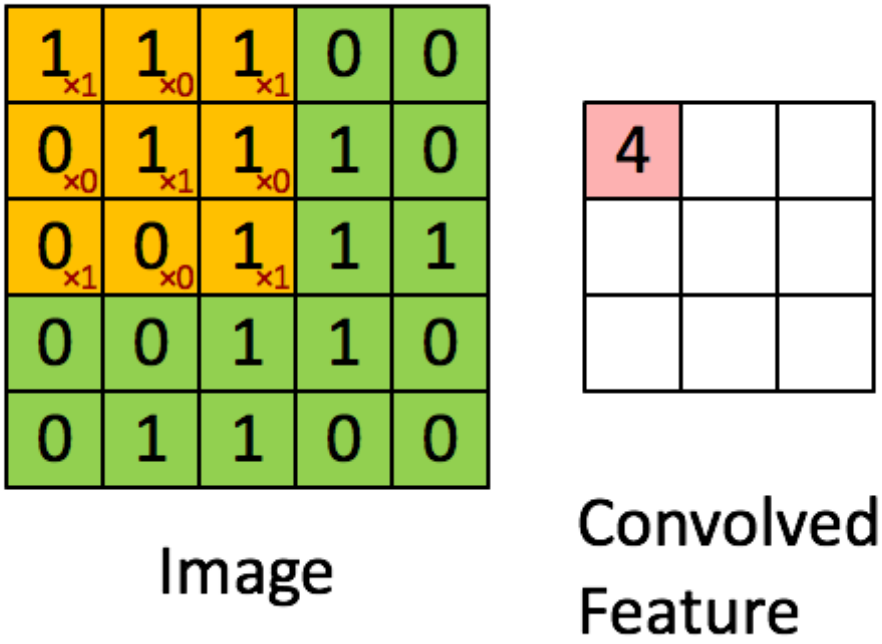


4 x 4 image (reduced size)

Demo

1	0	1
0	1	0
1	0	1

Filter/kernel

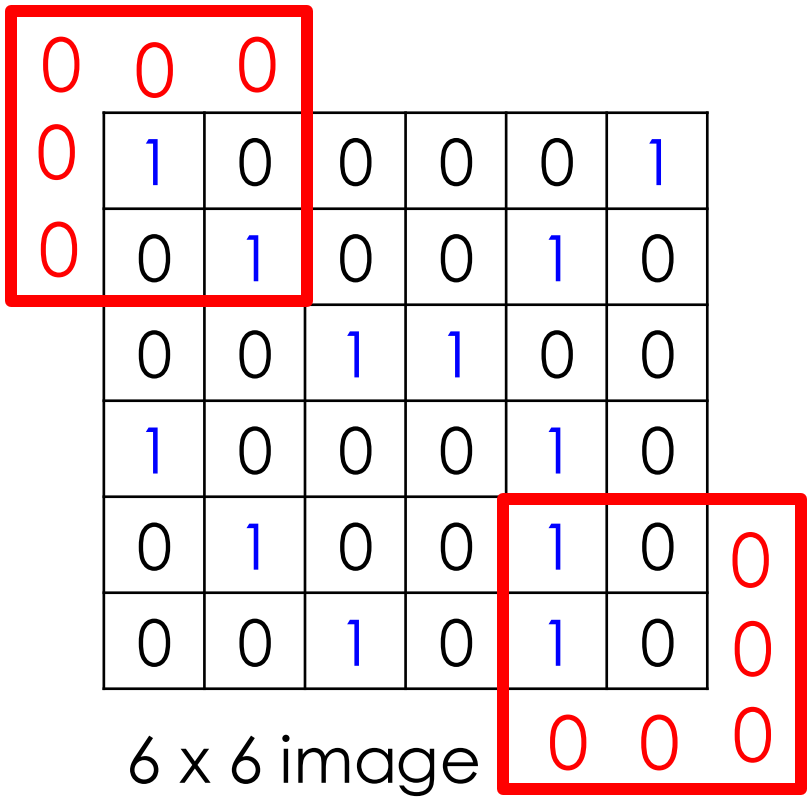


https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

CNN – Zero Padding

1	-1	-1
-1	1	-1
-1	-1	1

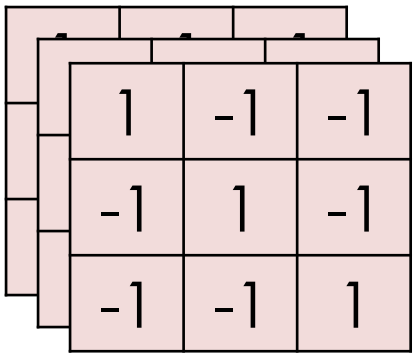
Filter 1



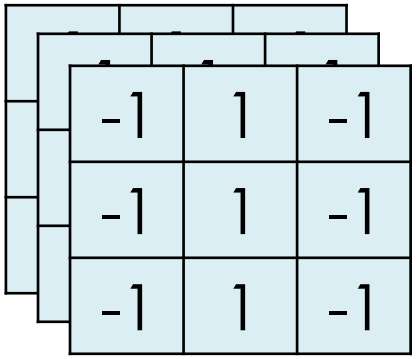
You will get another 6 x 6 image in this way

Zero padding

CNN – Color image

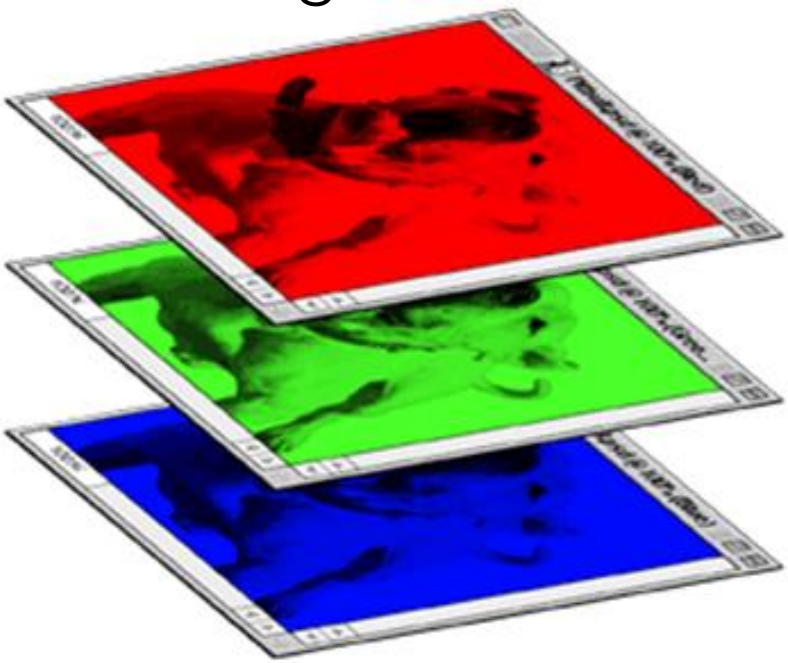


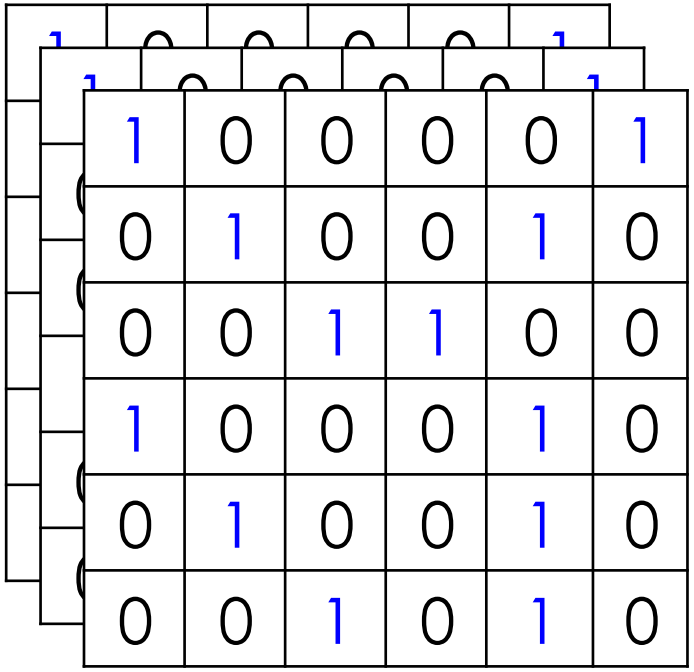
Filter 1

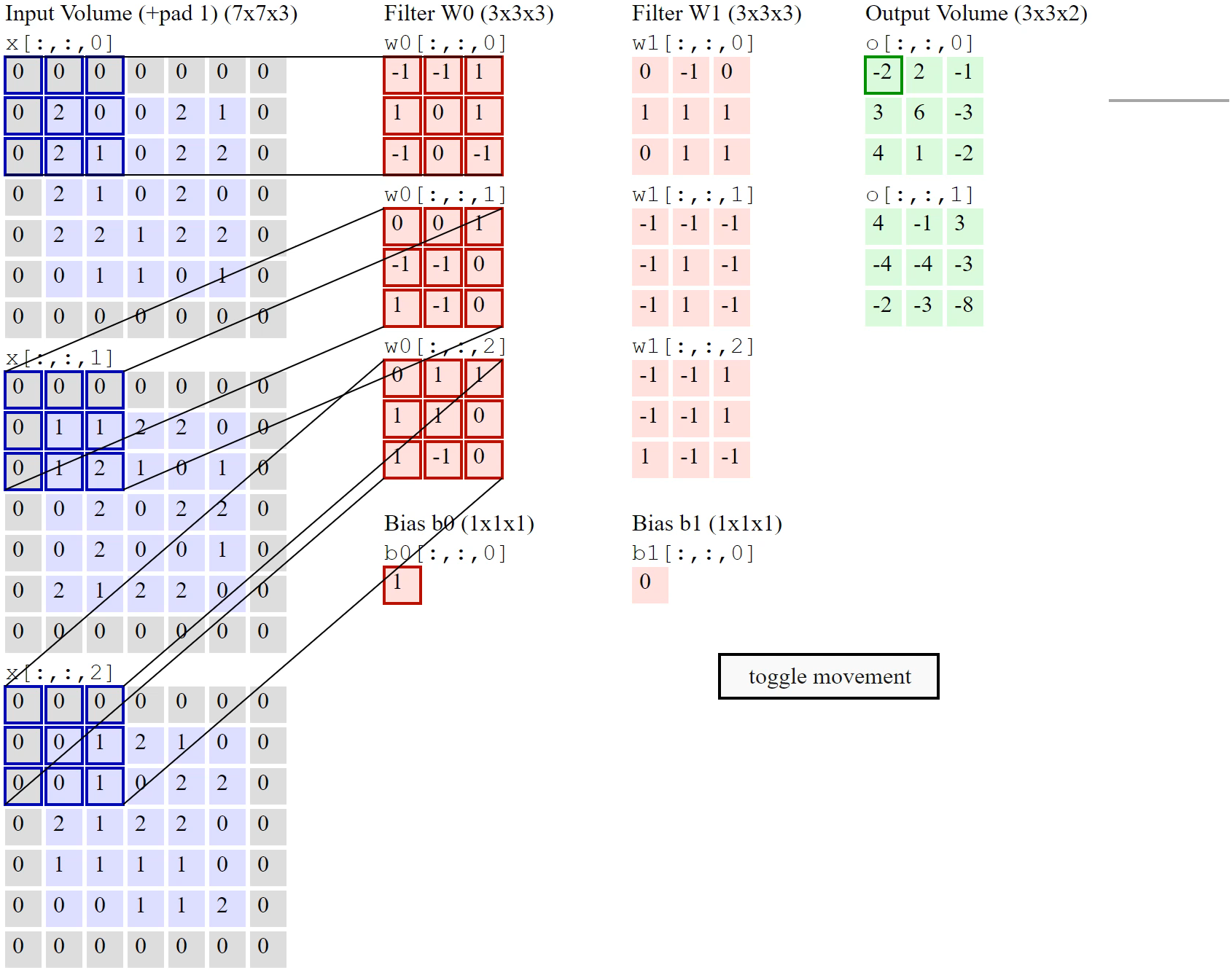


Filter 2

Color image



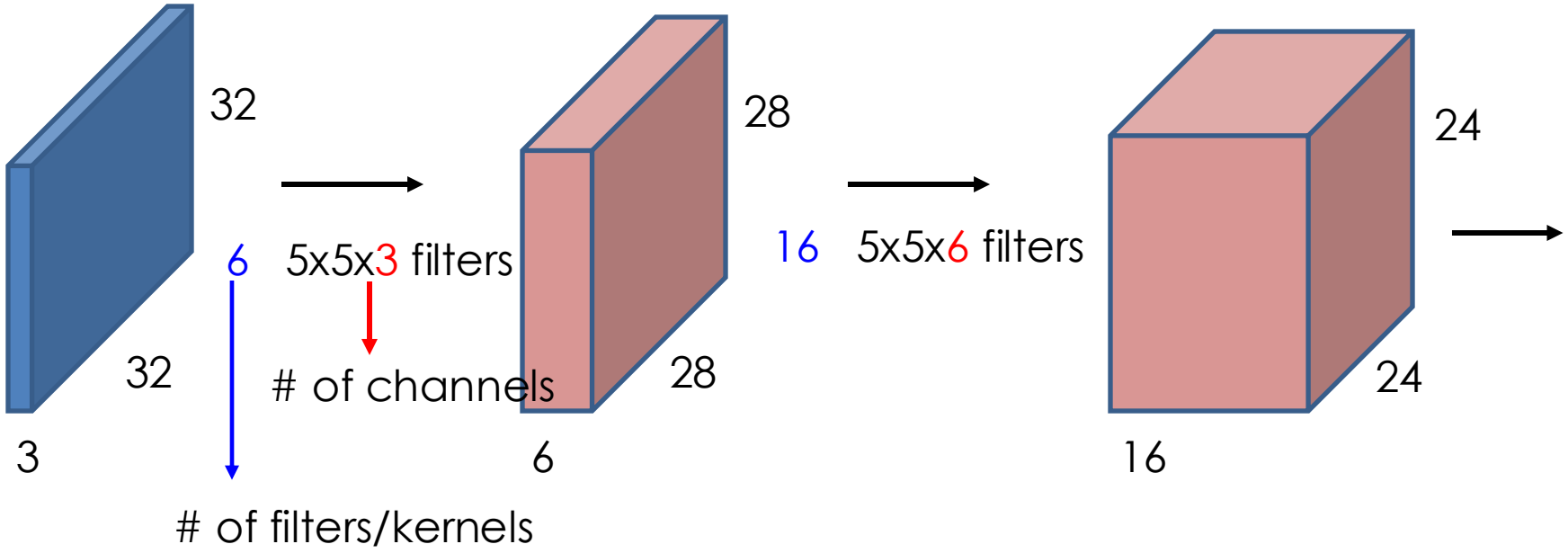




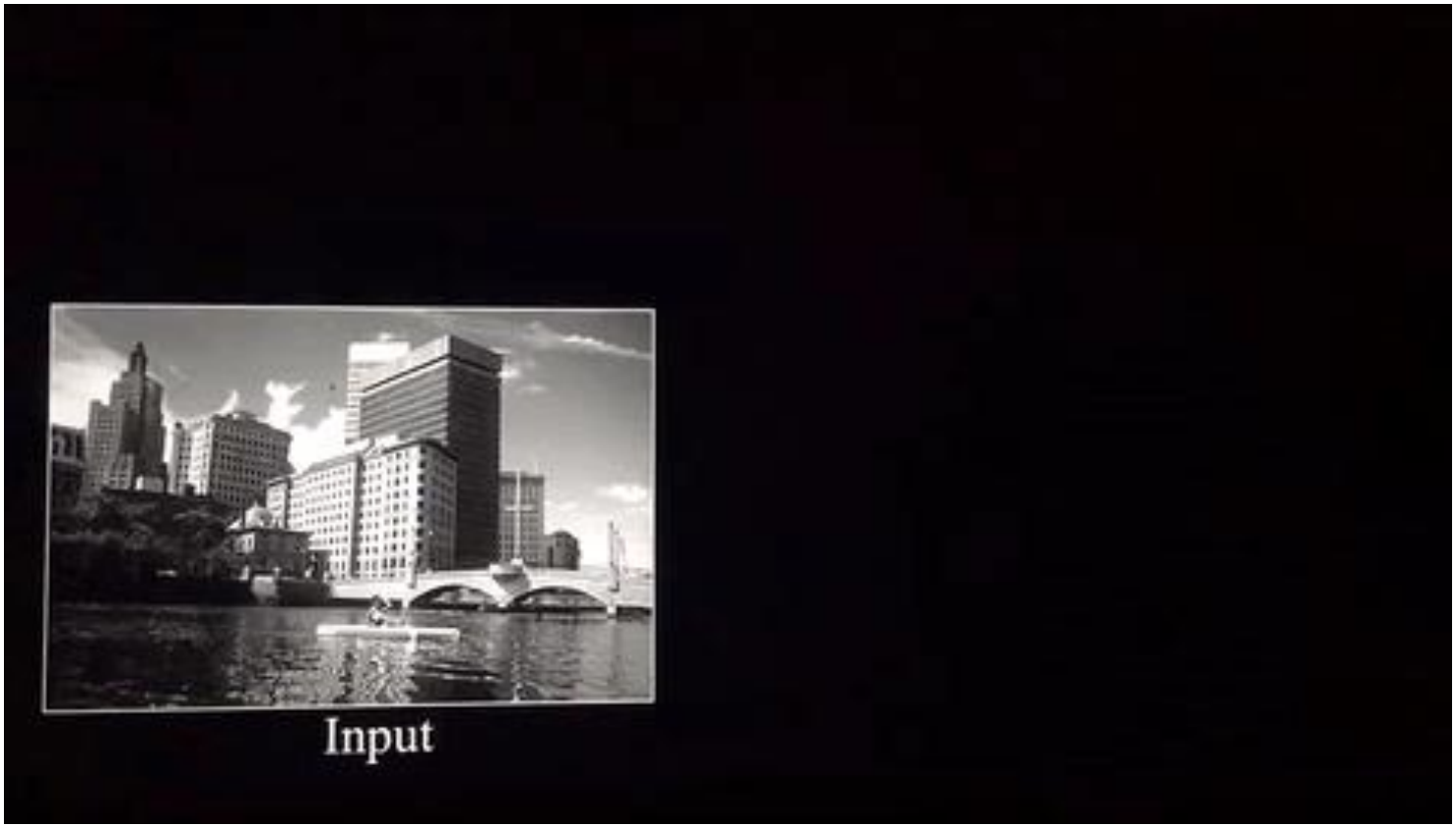
http://cs231n.github.io/convolutional-networks/

Convolutional Network

- Convolution network is a sequence of these layers

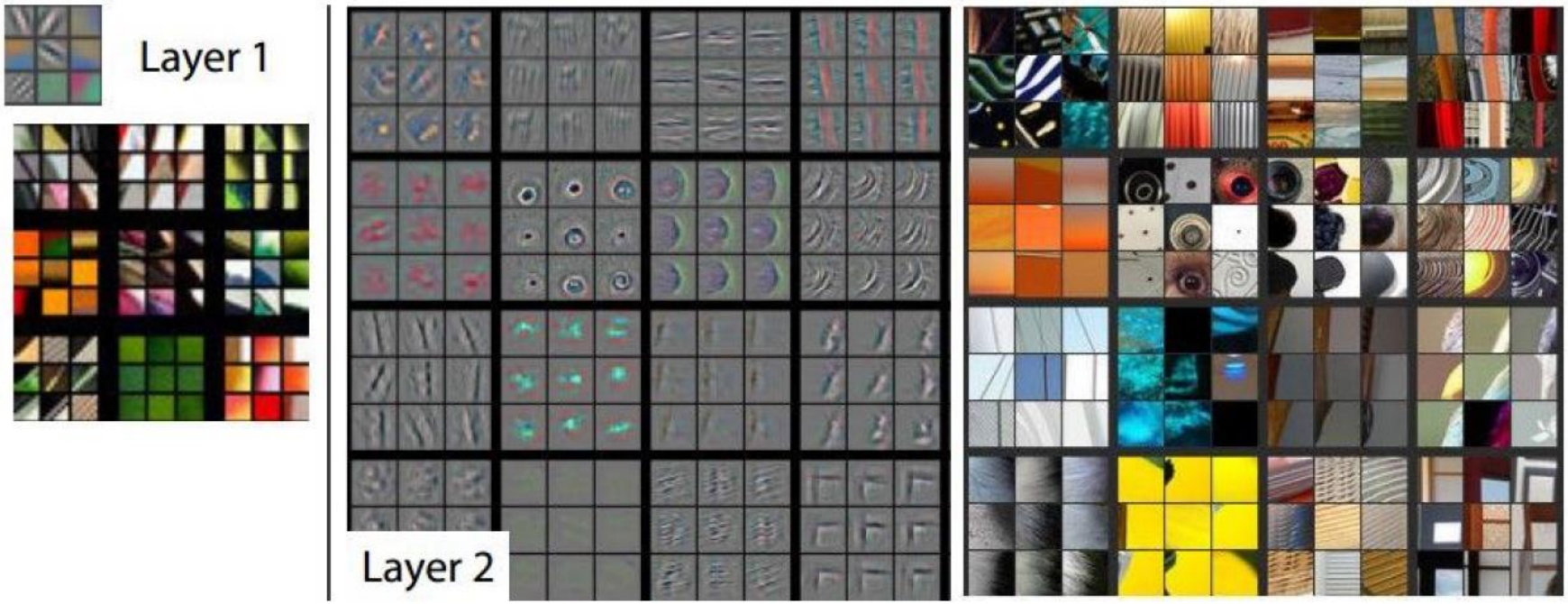


Convolution - Intuition

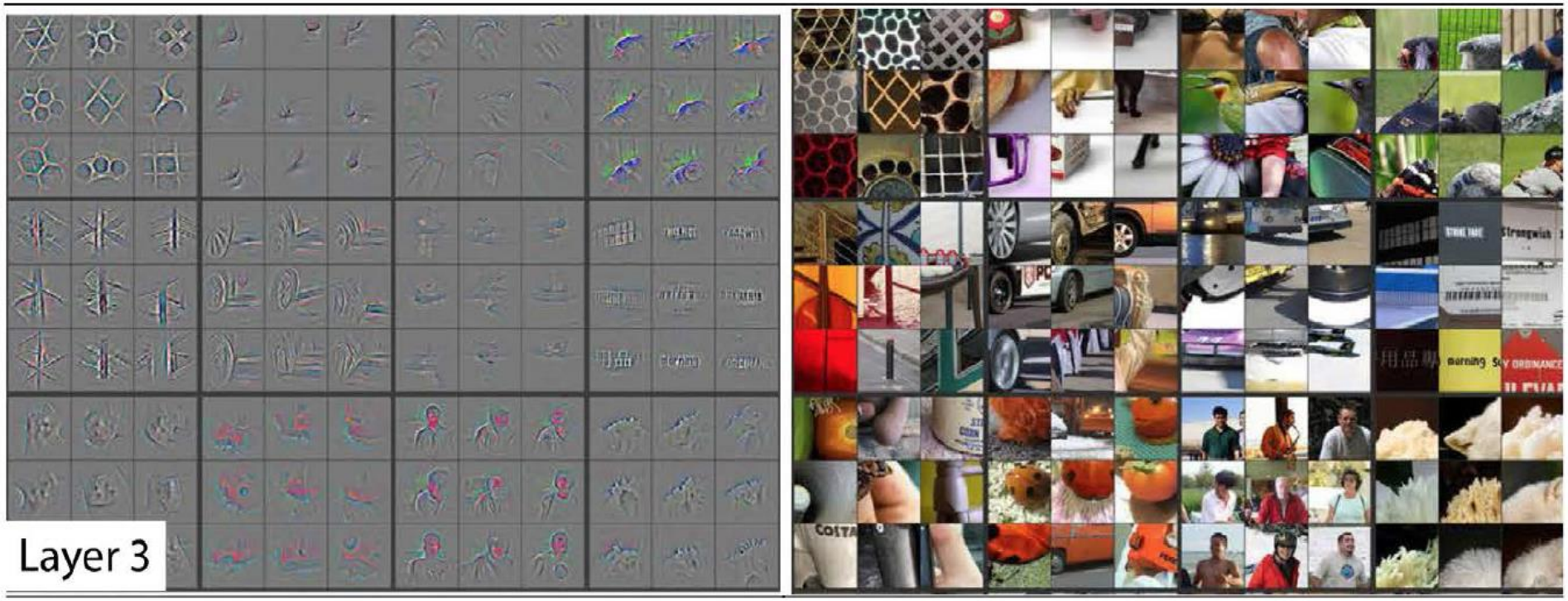


Source : https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

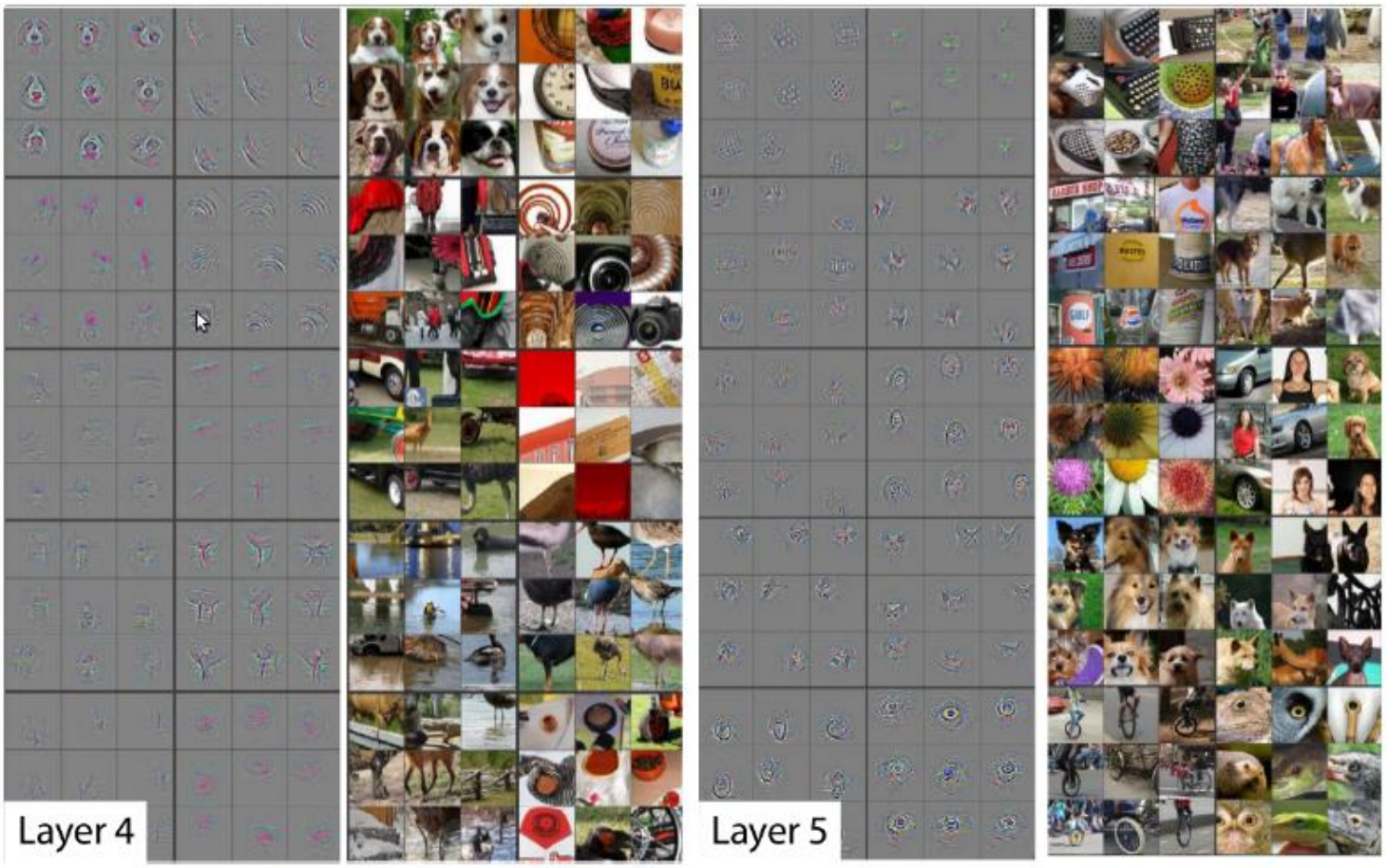
Visualizing CNN



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks."*European conference on computer vision*. Springer, Cham, 2014.



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.

Variants of Convolution Operation

- Dilated convolution [1]
- Depth-wise separable convolution [2]
- Grouped convolution [3]

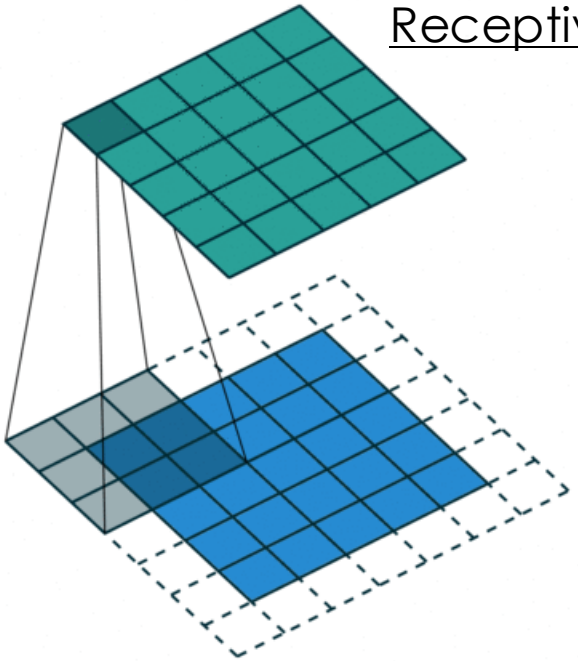
[1] Yu, Fisher, and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions." arXiv preprint arXiv:1511.07122 (2015).
[2] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).
[3] Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

Variants of Convolution Operation

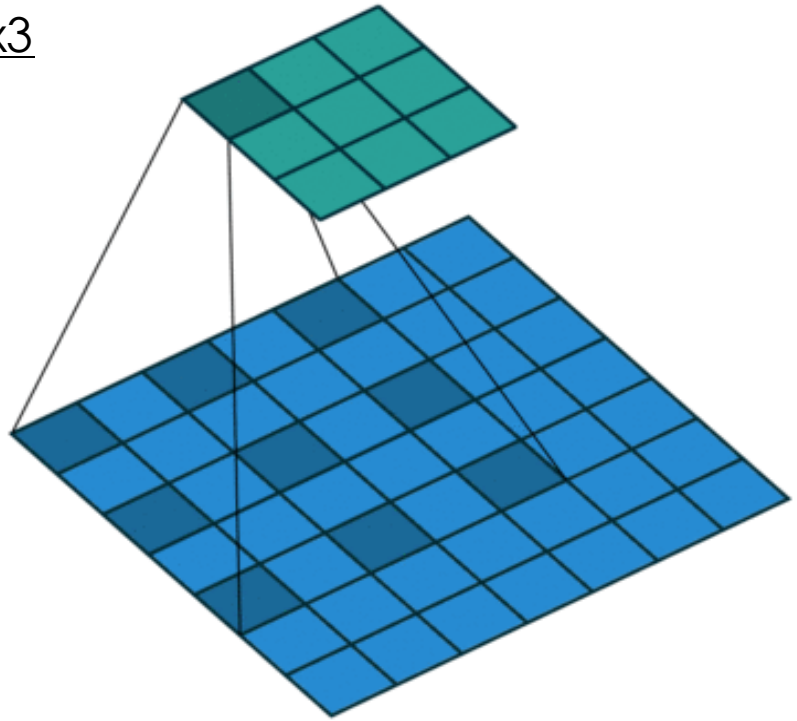
- Dilated/Atrous Convolution

Dilation rate = 2
Receptive filed = 5x5
With only 9 parameters

Receptive filed = 3x3



Normal convolution

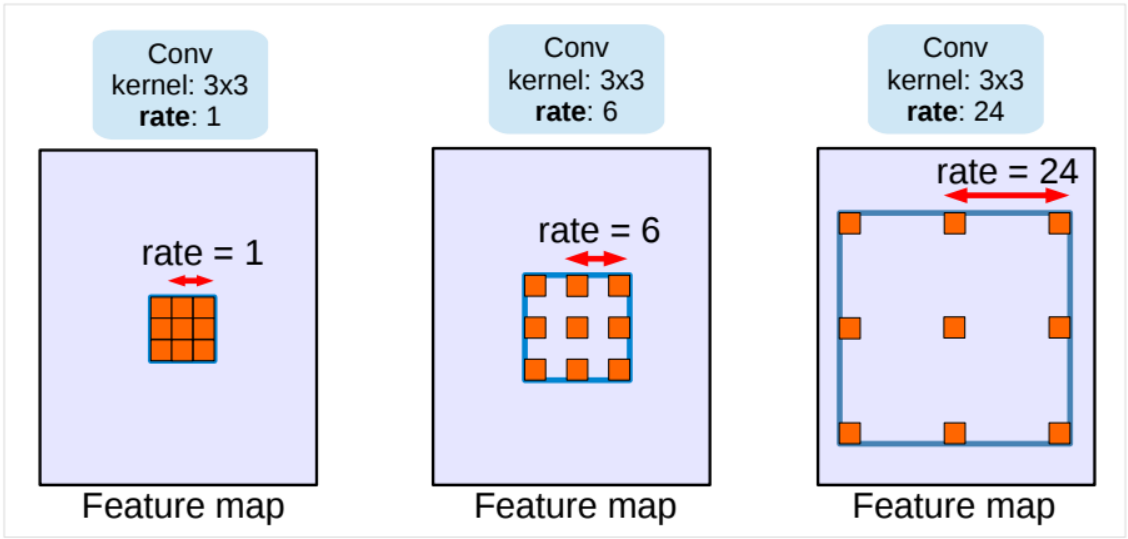


Effective in segmentation task

https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Variants of Convolution Operation

- Dilated Convolution

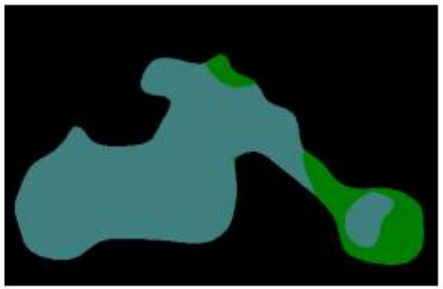


- Large value of atrous/dilation rate enlarges the model's field-of-view
- Enabling object encoding at multiple scales

Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." *arXiv preprint arXiv:1706.05587* (2017).

Variants of Convolution Operation

Input

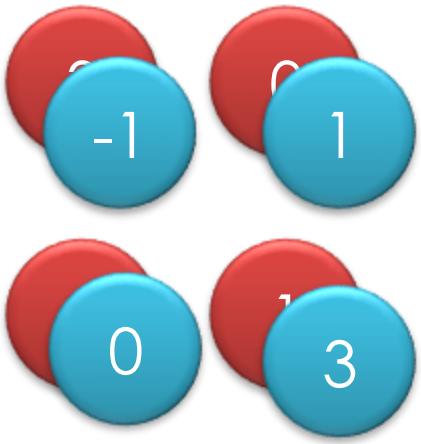


LargeFOV

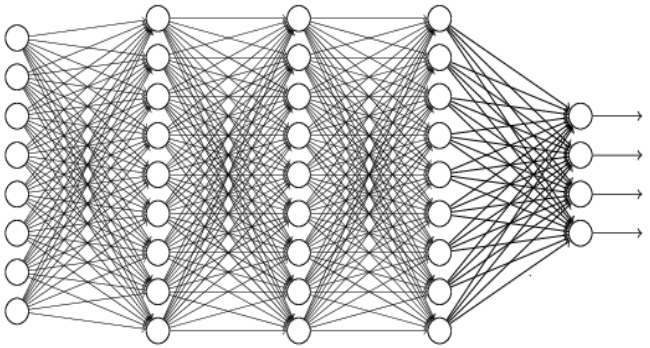
Atrous Spatial Pyramid Pooling (ASPP)

Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018): 834-848.

Flatten

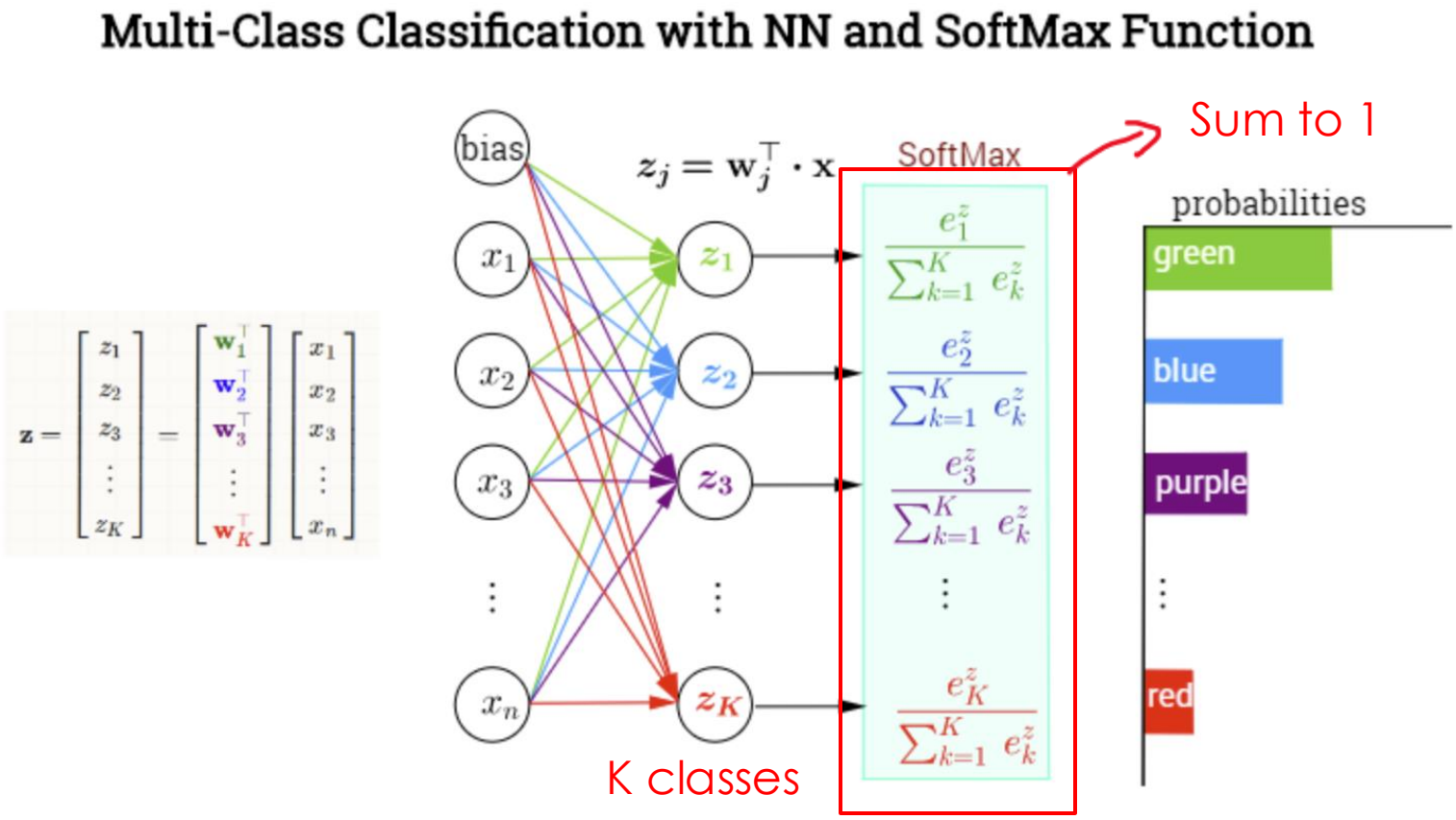


Flatten



Fully Connected Feedforward network

Softmax



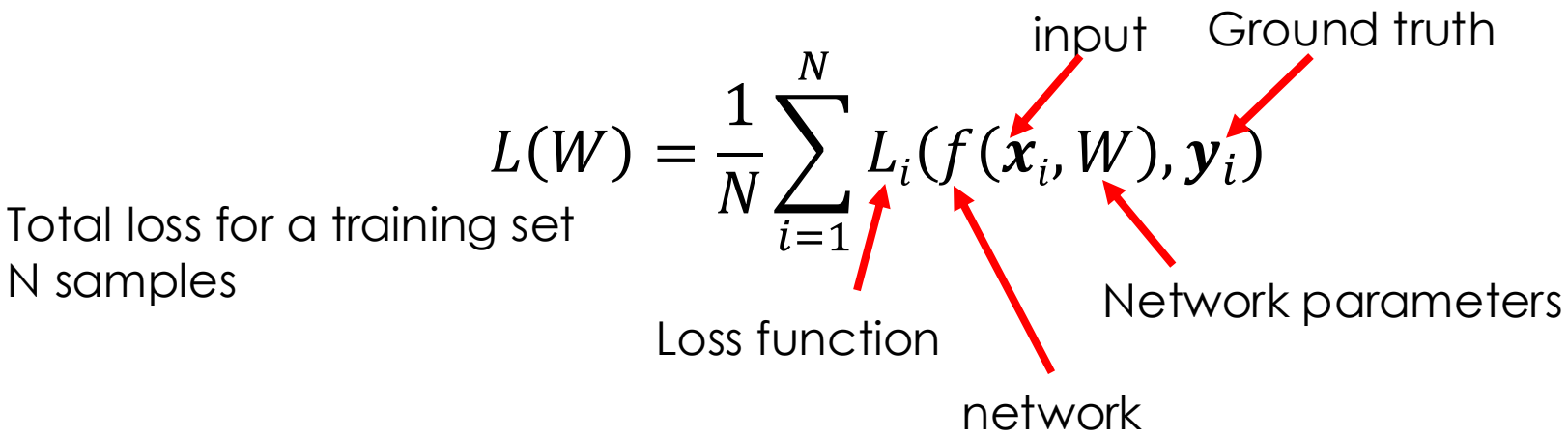
The softmax as

$$\sigma(j) = \frac{\exp(\mathbf{w}_j^\top \mathbf{x})}{\sum_{k=1}^K \exp(\mathbf{w}_k^\top \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^K \exp(z_k)}$$

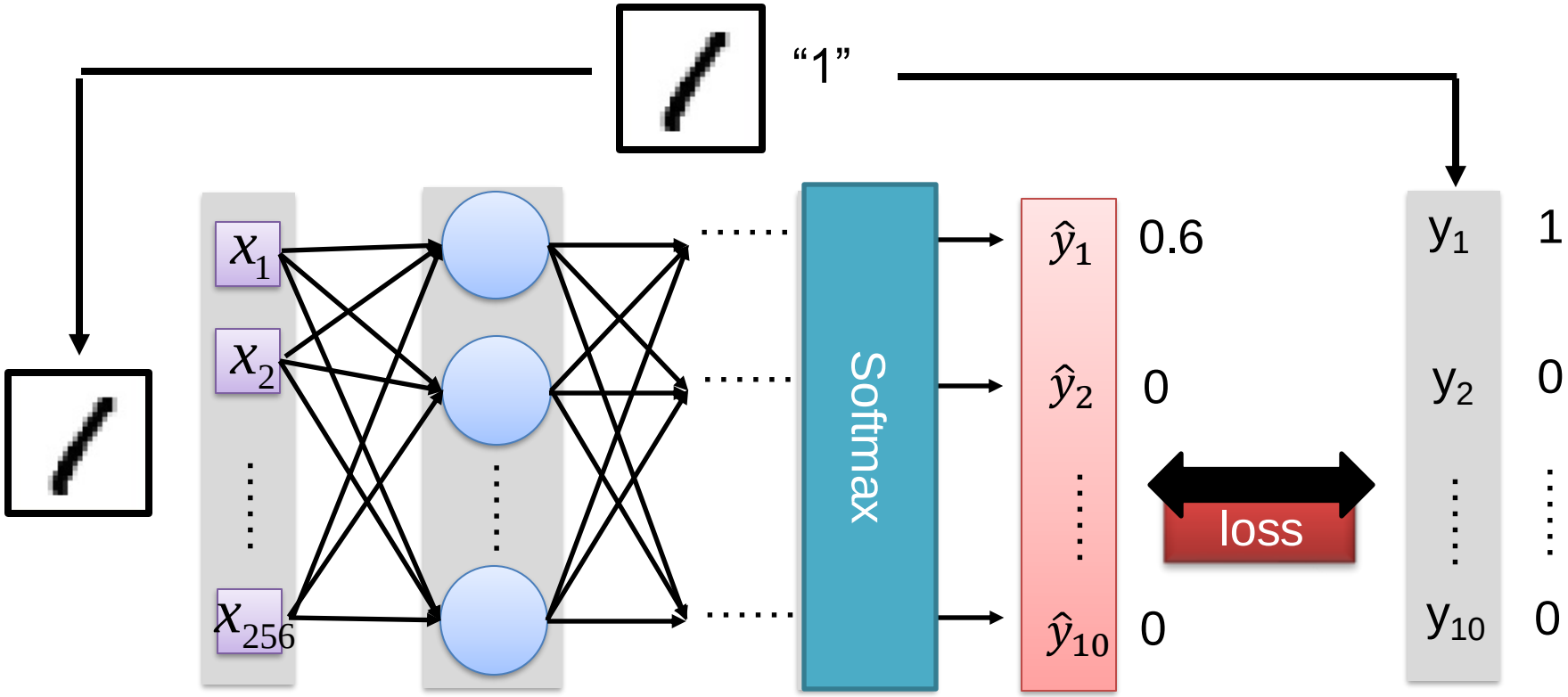
https://victorzhou.com/blog/softmax/

Loss Function

- Way to define how good the network is performing
  - In terms of prediction
- Network training (Optimization)
  - Find the best network parameters to minimize the loss



Choosing Proper Loss

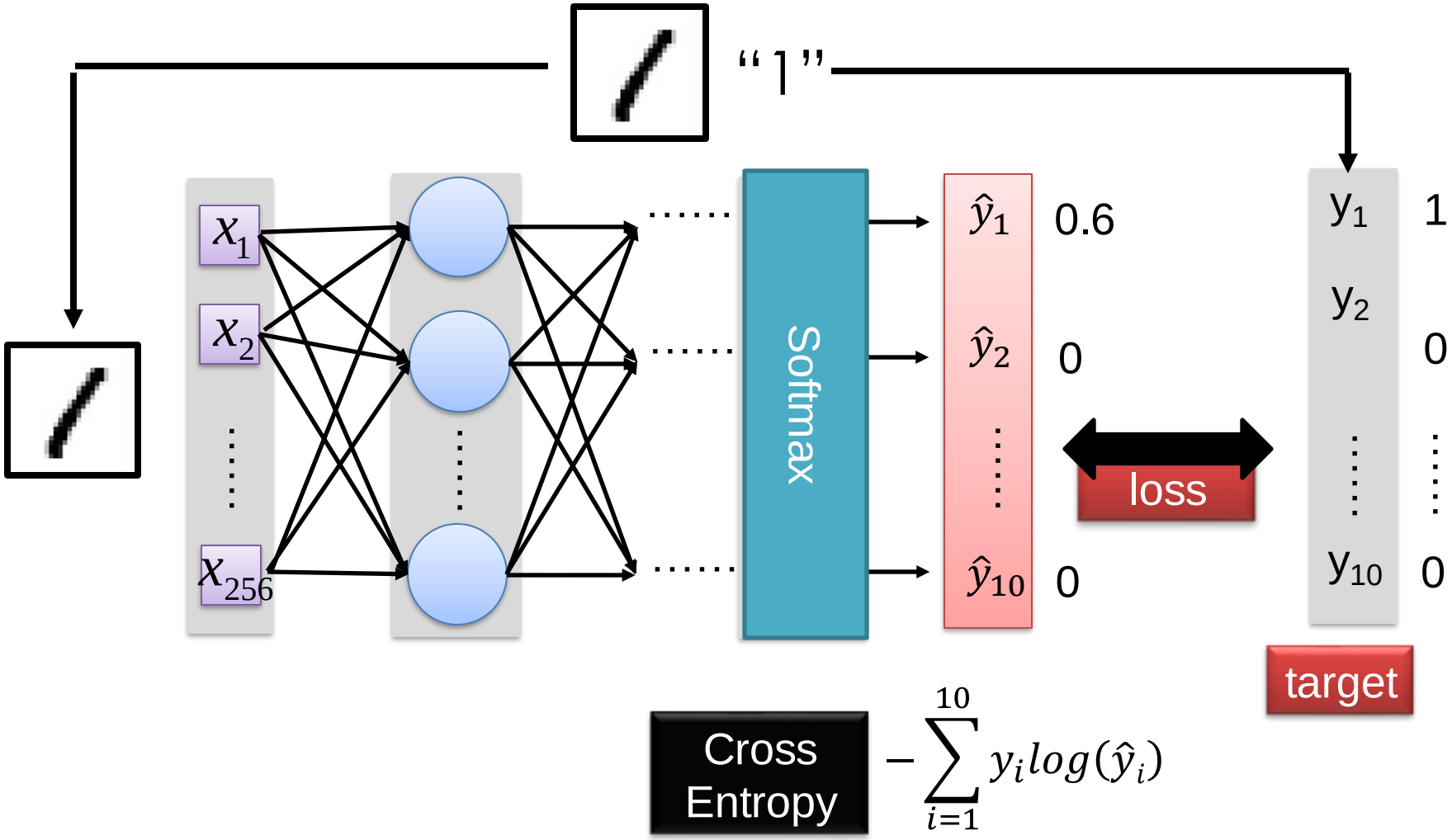


Square Error

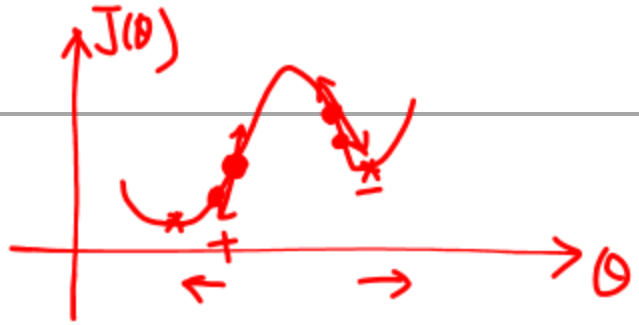
$$\sum_{i=1}^{10}(y_i - \hat{y}_i)^2$$

Target (one hot encoding)

Choosing Proper Loss



Gradient Descent

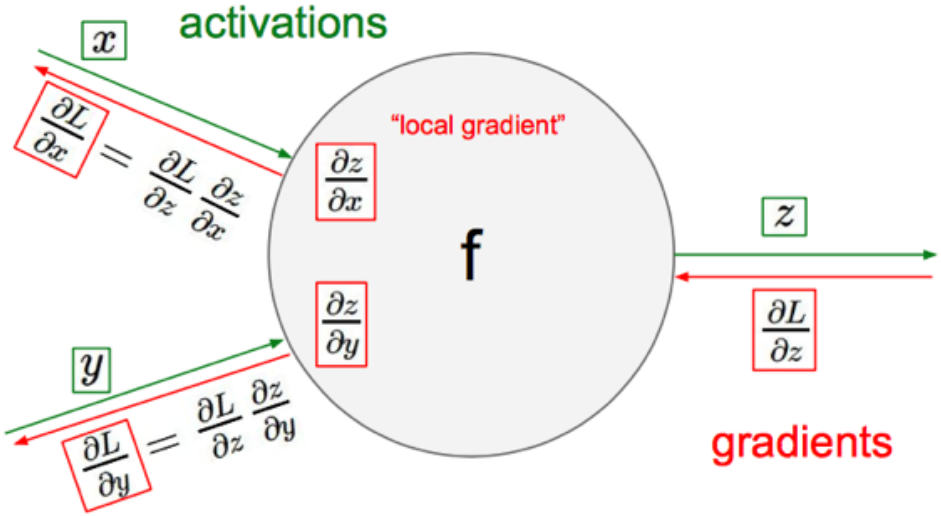


objective/cost function $J(\theta)$

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{d}{d\theta_j^{old}} J(\theta)$$ Update each element of θ

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$ Matrix notation for all parameters

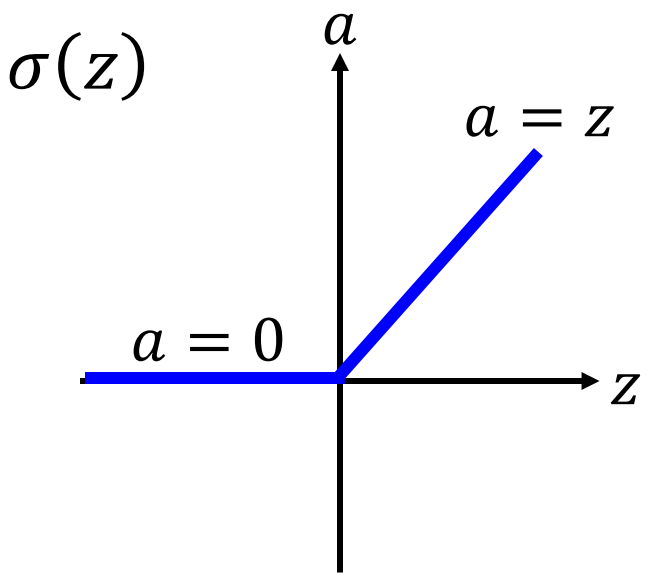
learning rate



Recursively apply **chain rule** though each node

ReLU

- Rectified Linear Unit (ReLU)

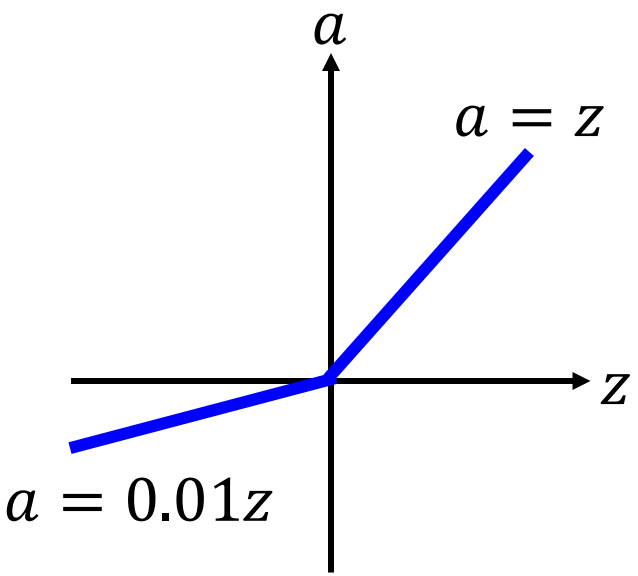


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

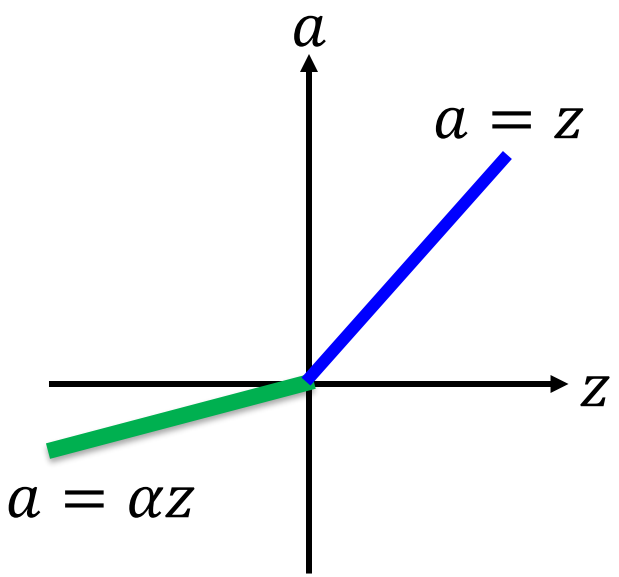
Vanishing gradient problem

ReLU - variant

Leaky ReLU



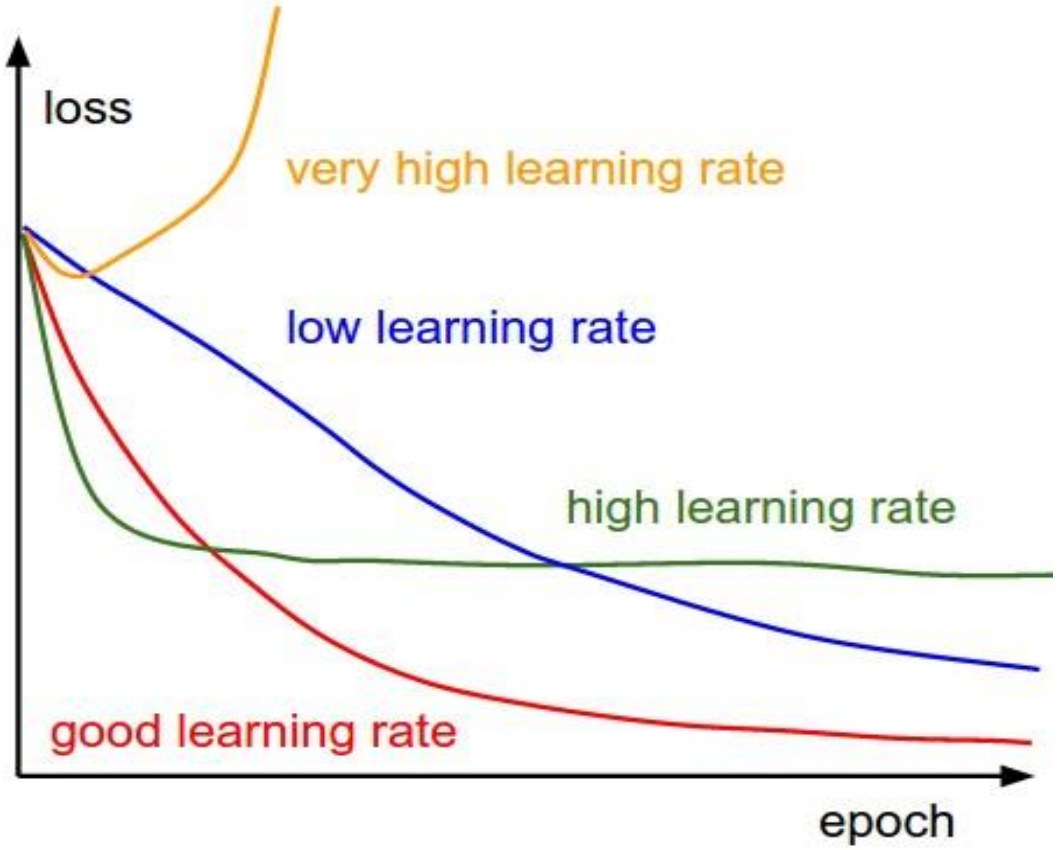
Parametric ReLU



α also learned by gradient descent

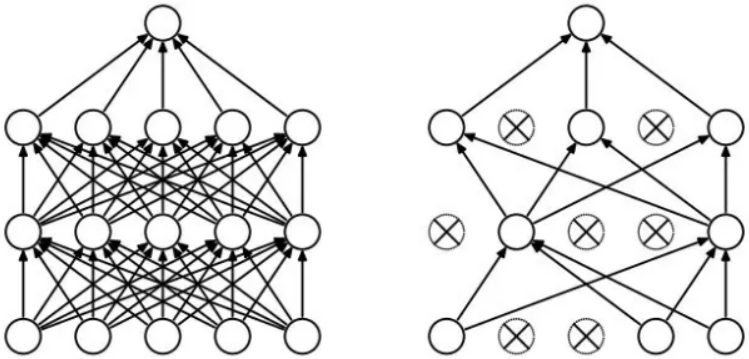
How to pick the learning rate?

- Too big = diverge, too small = slow convergence
- No “one learning rate to rule them all”
- Start from a high value and keep cutting by half if model diverges
- Learning rate schedule: decay learning rate over time



http://cs231n.github.io/assets/nn3/learningrates.jpeg

Regularization



Dropout

- Randomly drop units (along with their connections) during training
- Each unit retained with fixed probability p, independent of other units
- Hyper-parameter p to be chosen (tuned)

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

L2 = weight decay

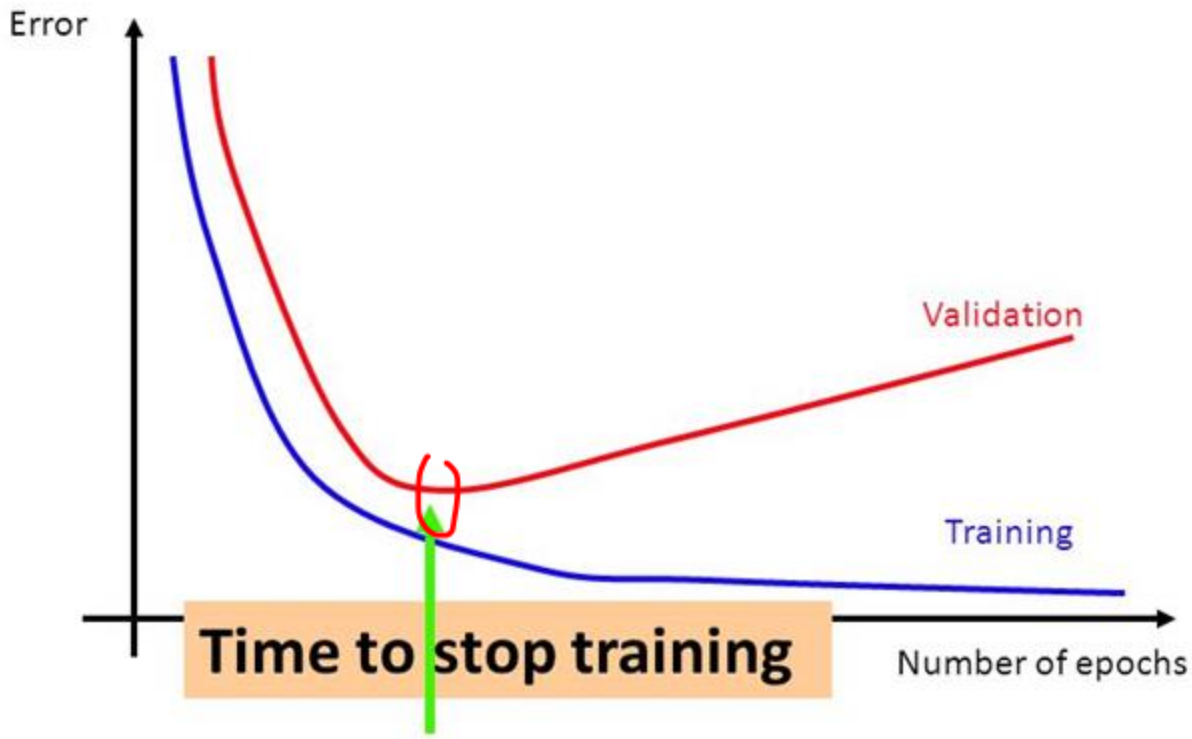
- Regularization term that penalizes big weights, added to the objective – reduce overfitting
- Weight decay value determines how dominant regularization is during gradient computation
- Big weight decay coefficient → big penalty for big weights

$$J_{reg}(\theta) = J(\theta) + \lambda \sum_k \theta_k^2$$

Early-stopping

- Use validation error to decide when to stop training
- Stop when monitored quantity has not improved after n subsequent epochs

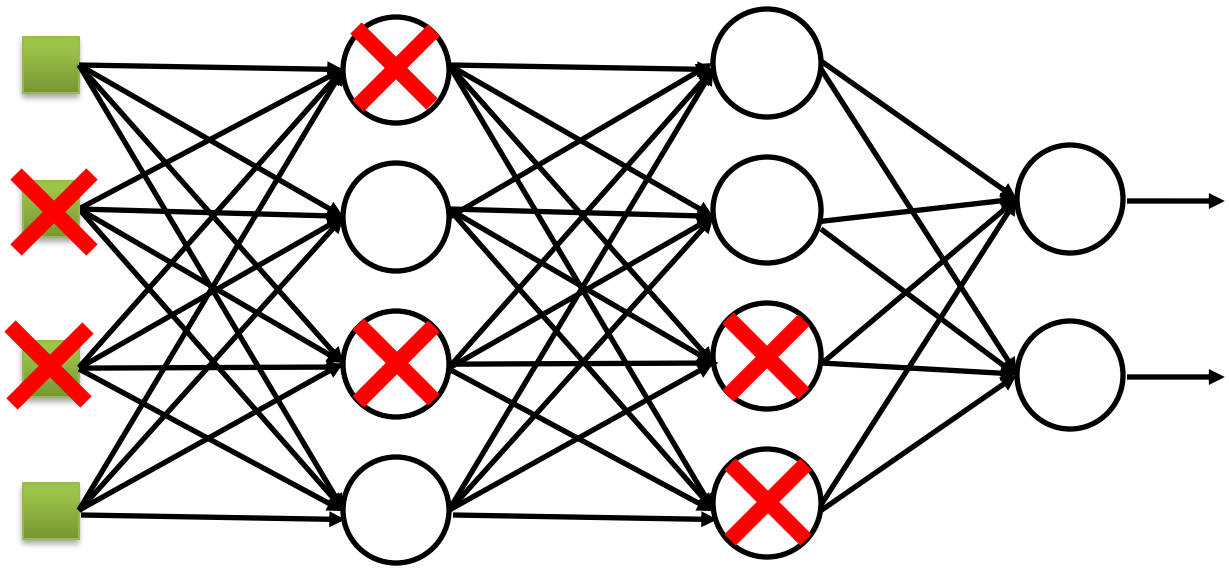
Early Stopping



Credit: Stephen Marsland

Dropout

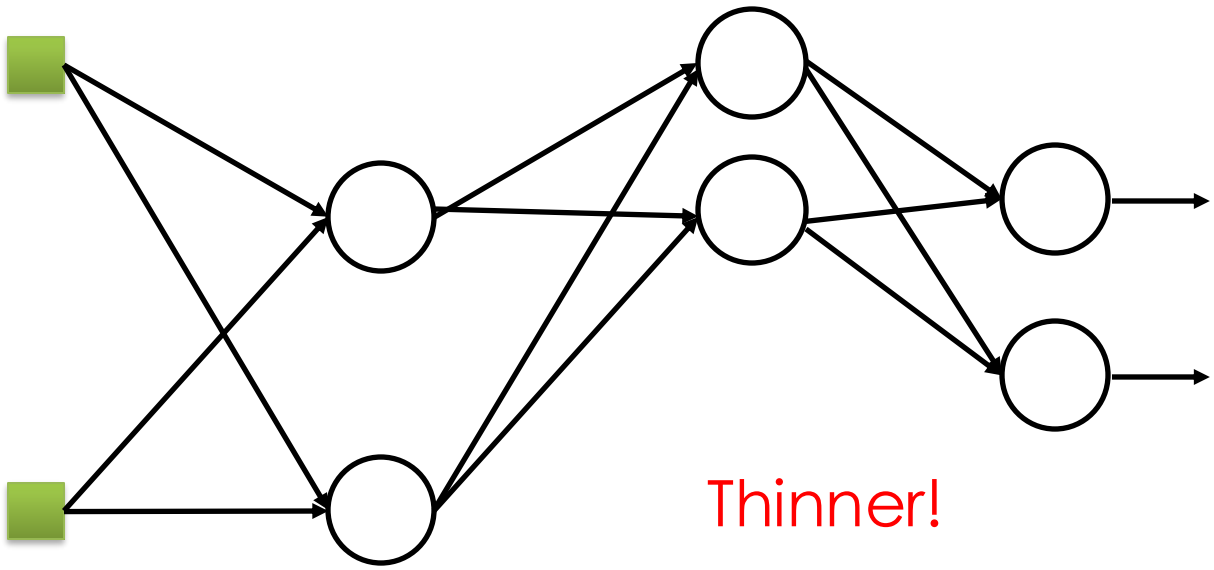
Training:



- **Each time before updating the parameters**
  - Each neuron has p% to dropout

Dropout

Training:

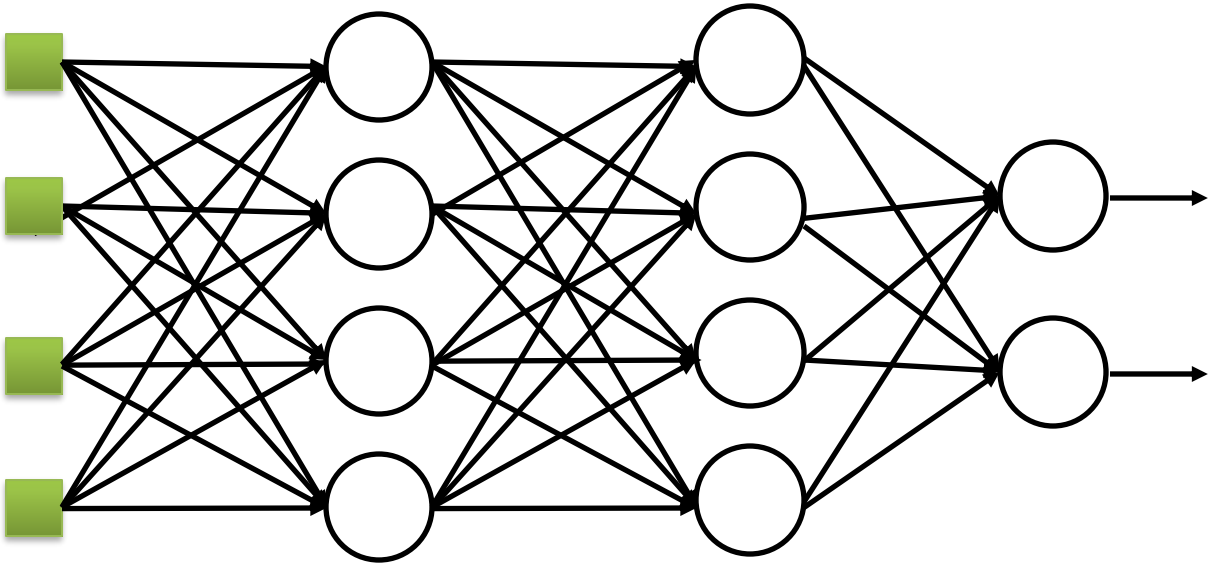


- **Each time before updating the parameters**
  - Each neuron has p% to dropout
  - Using the new network for training

The structure of the network is changed.

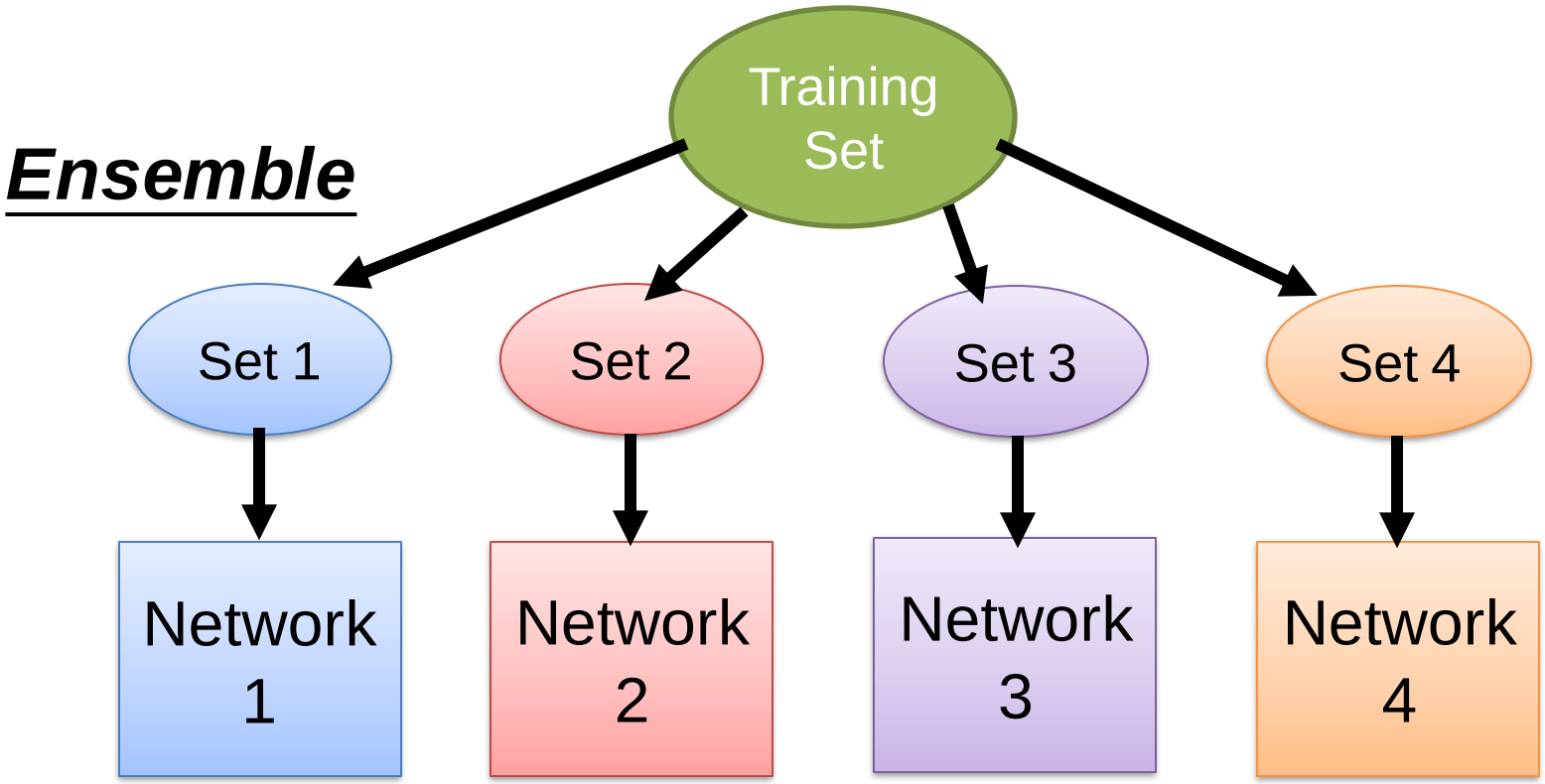
Dropout

Testing:



- No dropout

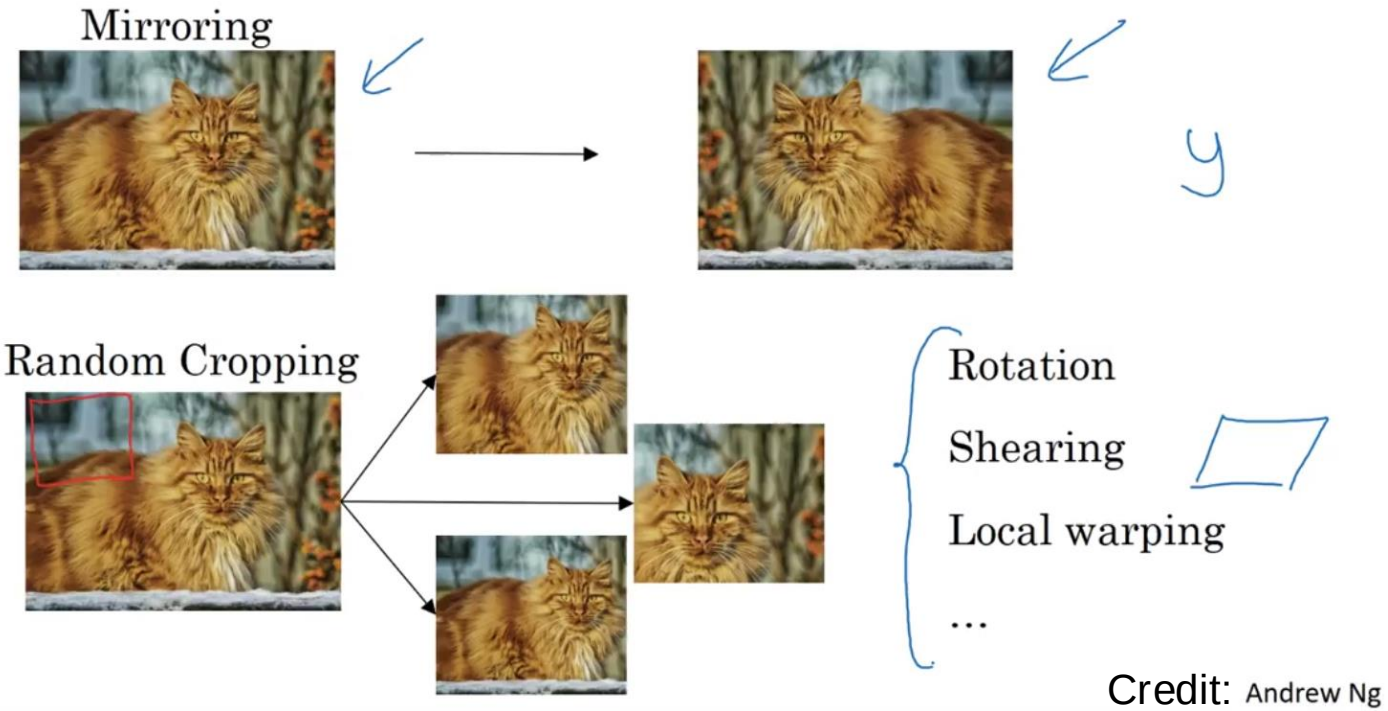
Dropout is a kind of ensemble



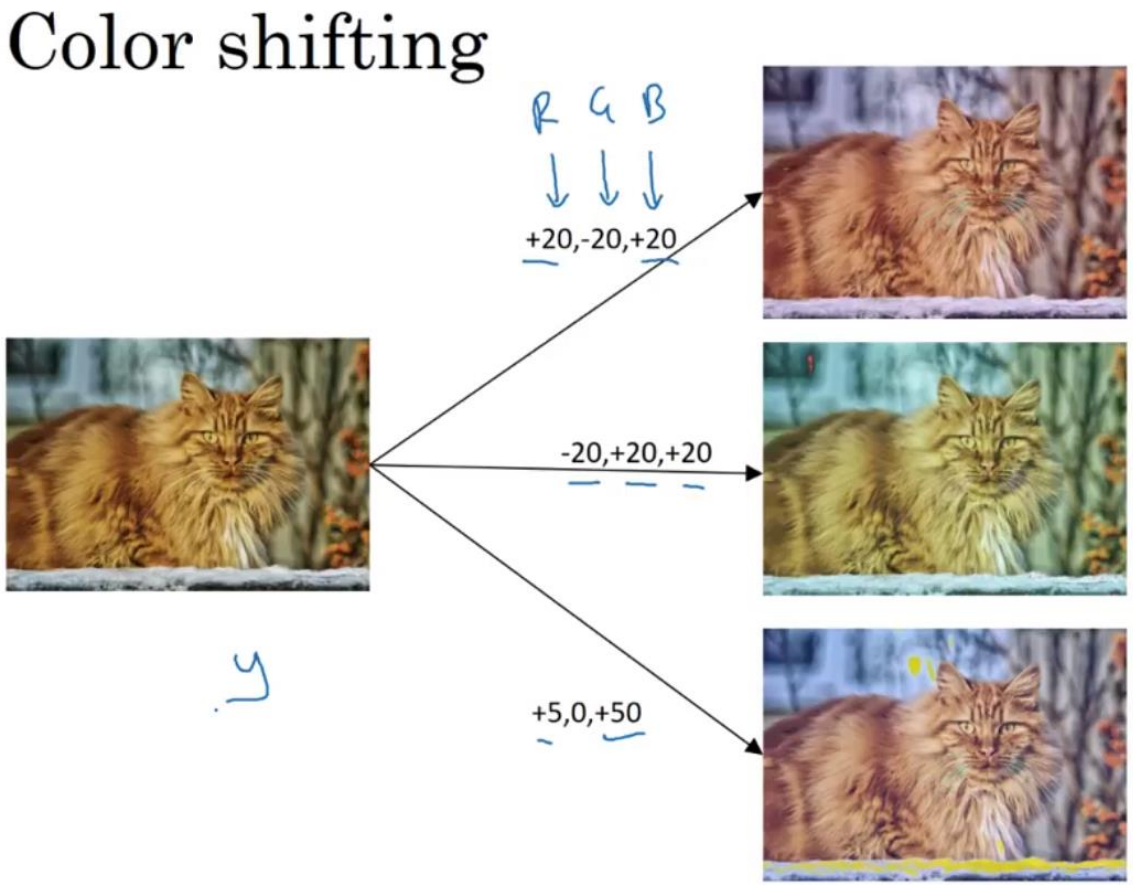
Train a bunch of networks with different structures

Data Augmentation

- Why data augmentation -> increase training data



Data Augmentation



Credit: Andrew Ng

Data Augmentation

- Color space conversion

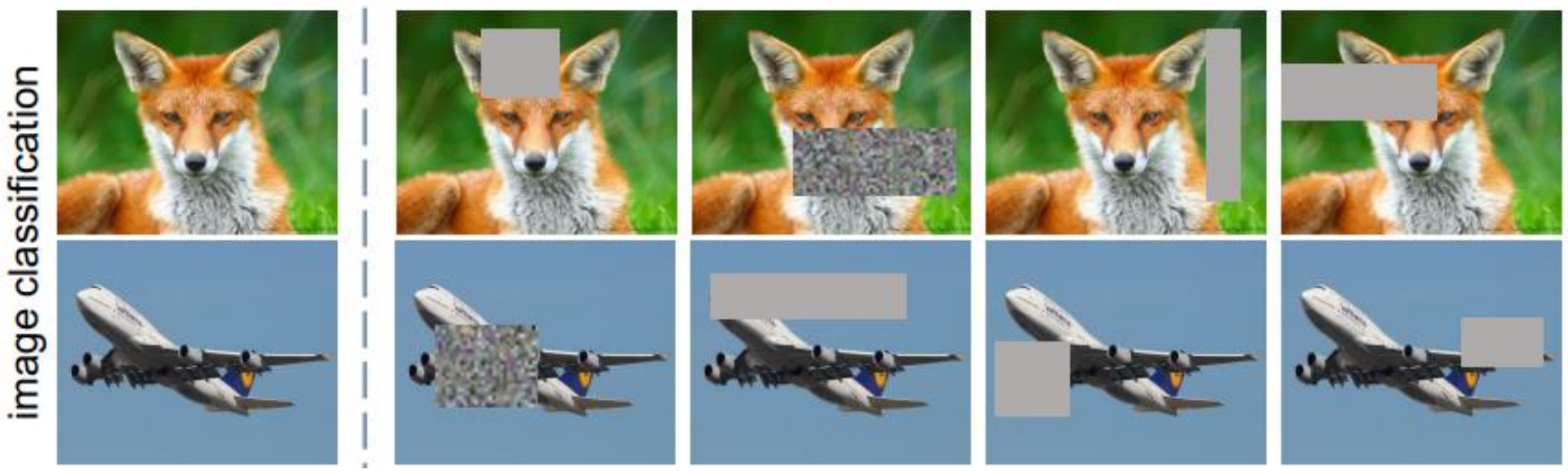
Comparison of results for different color spaces on CIFAR-10 with simple CNN

Color Space	Accuracy	Time
RGB	78.89	26 secs
HSV	78.57	26 secs
YUV	78.89	26 secs
LAB	**80.43**	26 secs
YIQ	78.79	26 secs
XYZ	78.72	26 secs
YPbPr	78.78	26 secs
YCbCr	78.81	26 secs
HED	78.98	26 secs
LCH	78.82	26 secs

Gowda, Shreyank N., and Chun Yuan. "ColorNet: Investigating the importance of color spaces for image classification." *arXiv preprint arXiv:1902.00267* (2019).

Data Augmentation

- Random erasing

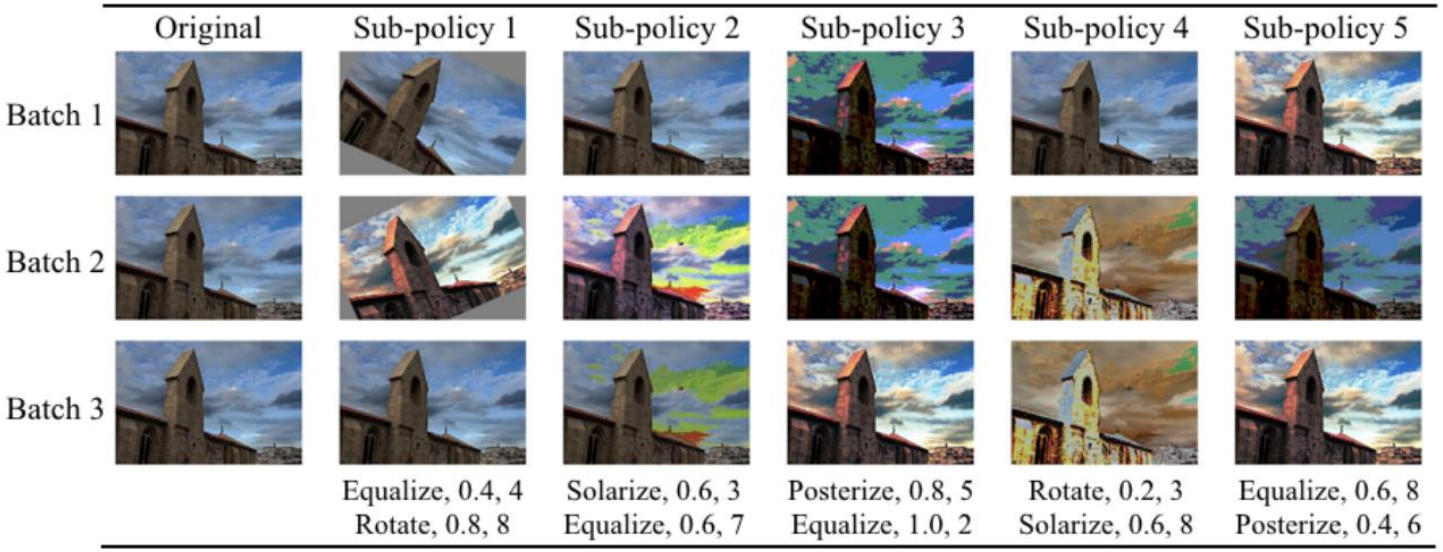


Reduces the risk of over-fitting and makes the model robust to occlusion
Improve the generalization ability of CNNs

Data Augmentation

- Learning Augmentation Policies from Data

Search space of operations: ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness,



Cubuk, Ekin D., et al. "Autoaugment: Learning augmentation policies from data." *arXiv preprint arXiv:1805.09501* (2018).

Data Augmentation

- Cutmix

	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.6** **(+2.3)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** **(+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7** **(+1.1)**

Yun, Sangdoo, et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features." *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

- More on data augmentation
  - (Pytorch) torchvision.transforms: https://pytorch.org/vision/master/transforms.html
  - https://github.com/AgaMiko/data-augmentation-review

Reading Material

- Training a Classifier — PyTorch Tutorials: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- CNNs, Part 1: An Introduction to Convolutional Neural Networks (Keras implementation)
  - https://victorzhou.com/blog/intro-to-cnns-part-1/
- CNNs, Part 2: Training a Convolutional Neural Network (Keras implementation)
  - https://victorzhou.com/blog/intro-to-cnns-part-2/
- **MNIST digit classification (Keras implementation)**
  - https://victorzhou.com/blog/keras-cnn-tutorial/#the-full-code
- Mnist classification (CNN Keras)
  - https://www.kaggle.com/elcaiseri/mnist-simple-cnn-keras-accuracy-0-99-top-1
- Bag of Tricks for Image Classification with Convolutional Neural Networks
  - https://openaccess.thecvf.com/content_CVPR_2019/papers/He_Bag_of_Tricks_for_Image_Classification_with_Convolutional_Neural_Networks_CVPR_2019_paper.pdf

Reading Material

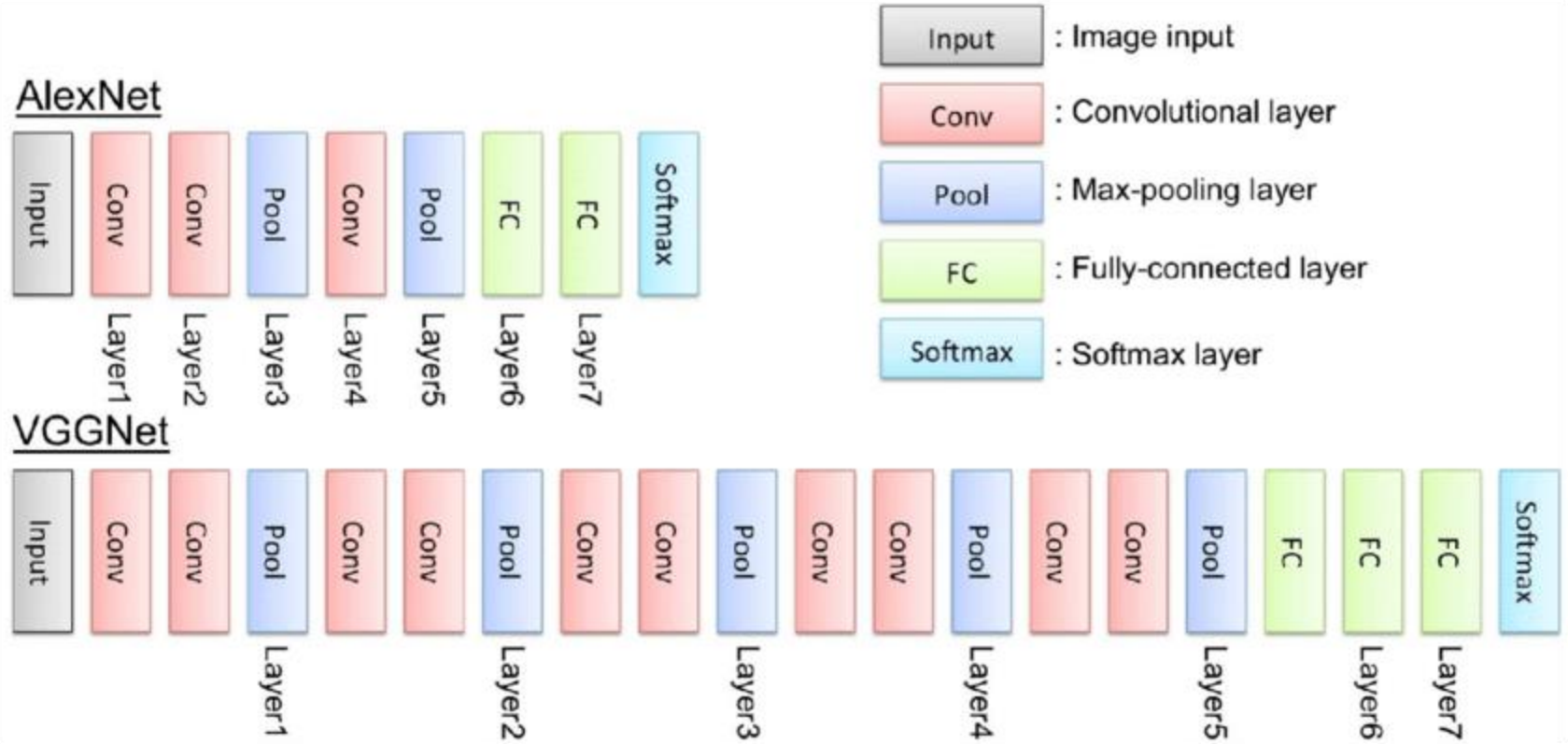
- Deep Learning Tuning Playbook
- https://github.com/google-research/tuning_playbook

Who is this document for?

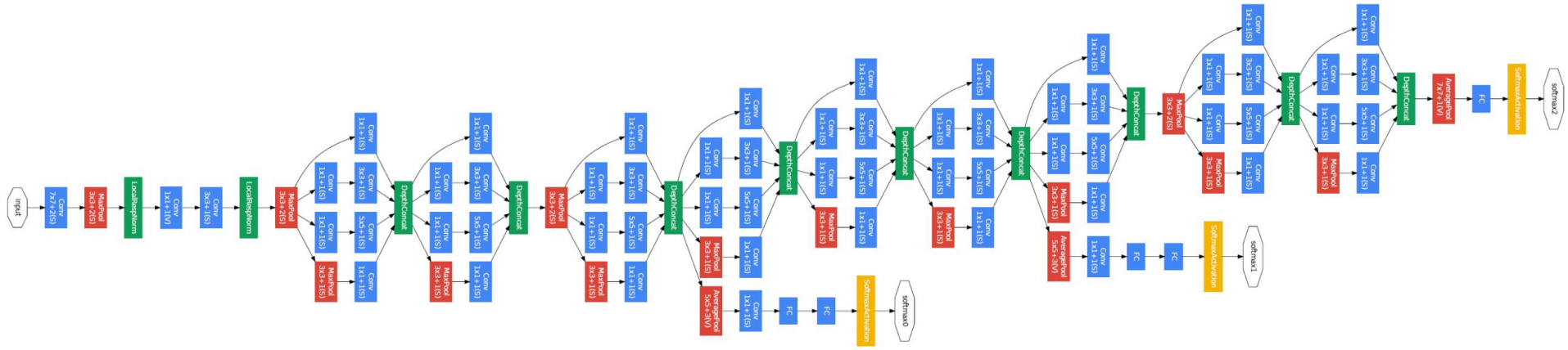
This document is for engineers and researchers (both individuals and teams) interested in **maximizing the performance of deep learning models**. We assume basic knowledge of machine learning and deep learning concepts.

Our emphasis is on the **process of hyperparameter tuning**. We touch on other aspects of deep learning training, such as pipeline implementation and optimization, but our treatment of those aspects is not intended to be complete.

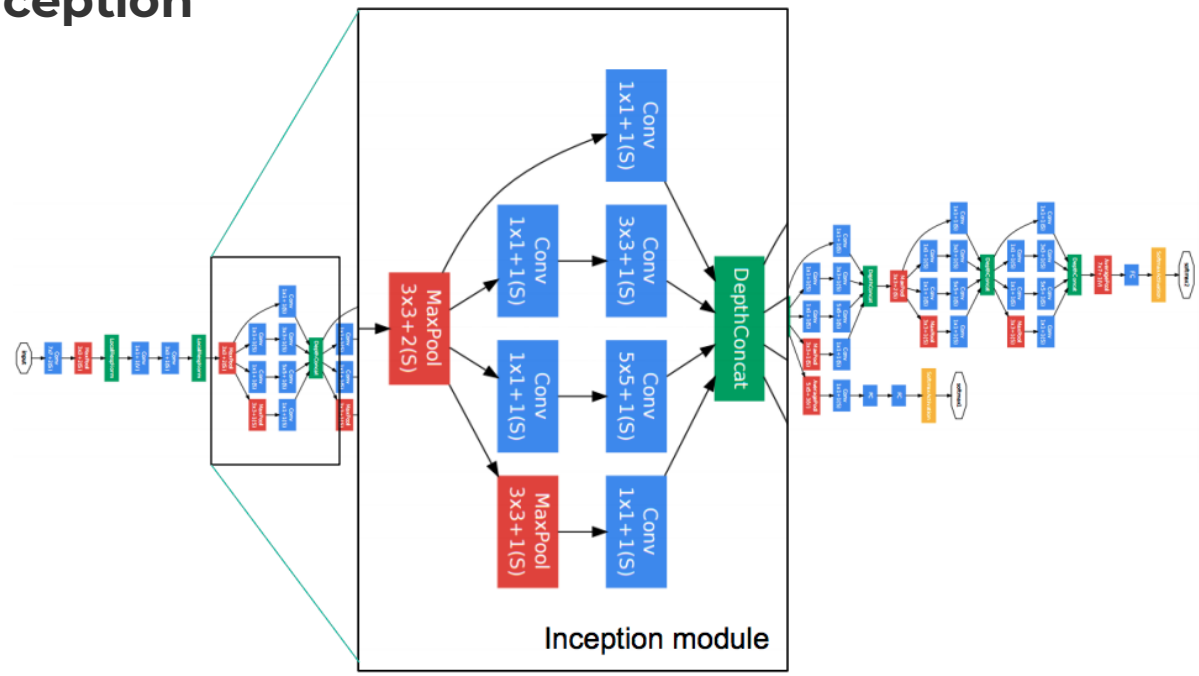
Case study


AlexNet
Input
Conv
Conv
Pool
Conv
Pool
FC
FC
Softmax
Layer1
Layer2
Layer3
Layer4
Layer5
Layer6
Layer7
VGGNet
Input
Conv
Conv
Pool
Conv
Conv
Pool
Conv
Conv
Pool
Conv
Conv
Pool
Conv
Conv
Pool
FC
FC
FC
Softmax
Layer1
Layer2
Layer3
Layer4
Layer5
Layer6
Layer7
Input : Image input
Conv : Convolutional layer
Pool : Max-pooling layer
FC : Fully-connected layer
Softmax : Softmax layer

Case study



GoogLeNet/Inception

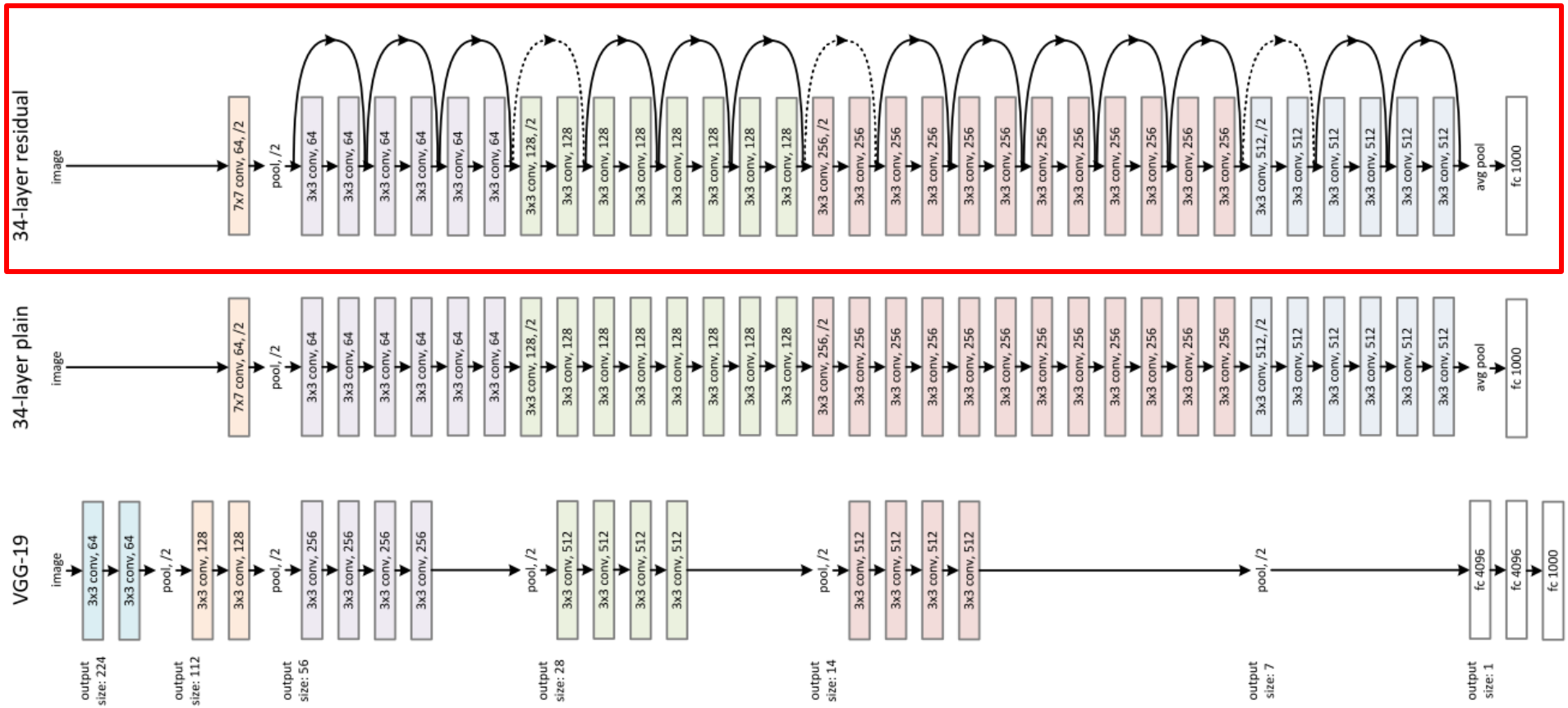


Case study

Skip connections:

Help in backpropagation

Residual Networks



He et. al. Deep Residual Learning for Image Recognition, 2015

Common network models

- Pytorch

```
import torchvision.models as models
resnet18 = models.resnet18()
alexnet = models.alexnet()
vgg16 = models.vgg16()
squeezenet = models.squeezenet1_0()
densenet = models.densenet161()
inception = models.inception_v3()
googlenet = models.googlenet()
shufflenet = models.shufflenet_v2_x1_0()
mobilenet = models.mobilenet_v2()
resnext50_32x4d = models.resnext50_32x4d()
wide_resnet50_2 = models.wide_resnet50_2()
mnasnet = models.mnasnet1_0()
```

TORCHVISION.MODELS

The models subpackage contains definitions of models for addressing differ
semantic segmentation, object detection, instance segmentation, person ke

Classification

The models subpackage contains definitions for the following model archite

- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNet v2
- ResNeXt
- Wide ResNet
- MNASNet

https://pytorch.org/docs/stable/torchvision/models.html

Common network models

- Keras

Available models

Model	Size	Top-1 Accuracy	Top-5 Accuracy	F
Xception	88 MB	0.790	0.945	
VGG16	528 MB	0.713	0.901	
VGG19	549 MB	0.713	0.900	
ResNet50	98 MB	0.749	0.921	
ResNet101	171 MB	0.764	0.928	
ResNet152	232 MB	0.766	0.931	
ResNet50V2	98 MB	0.760	0.930	
ResNet101V2	171 MB	0.772	0.938	
ResNet152V2	232 MB	0.780	0.942	
InceptionV3	92 MB	0.779	0.937	
InceptionResNetV2	215 MB	0.803	0.953	
MobileNet	16 MB	0.704	0.895	
MobileNetV2	14 MB	0.713	0.901	
DenseNet121	33 MB	0.750	0.923	
DenseNet169	57 MB	0.762	0.932	
DenseNet201	80 MB	0.773	0.936	
NASNetMobile	23 MB	0.744	0.919	

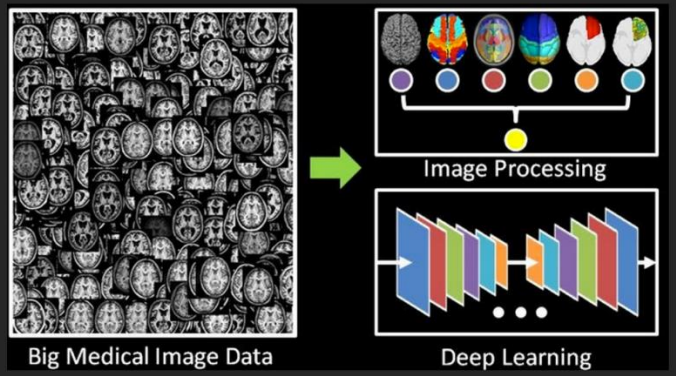
https://keras.io/api/applications/

Interpretability of Deep Neural Networks

Safety of AI models

Autonomous Driving

Trust of AI decision



Medical Diagnosis

Policy and Regulation

Right to the explanation for algorithmic decisions

Interpretability of Deep Neural Networks

Debate on if interpretability is necessary

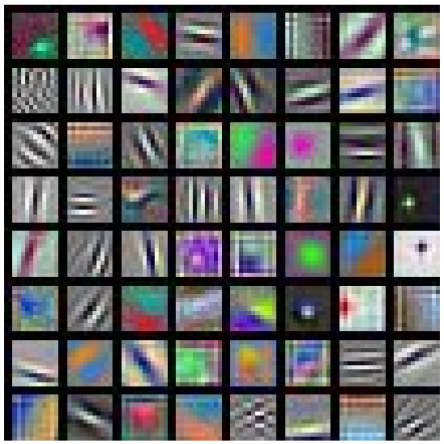
For: Rich Caruana, Patrice Simard

Against: Kilian Weinberger, Yann LeCun

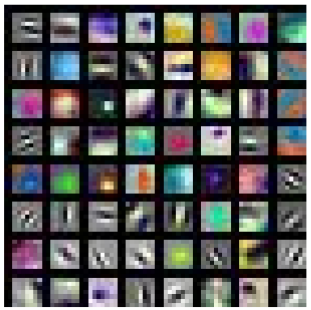
Interpretable Machine Learning Symposium

https://www.youtube.com/watch?v=93Xv8vJ2acI

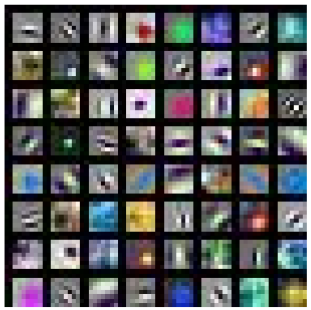
First Layer: Visualize Filters



AlexNet:
64 x 3 x 11 x 11

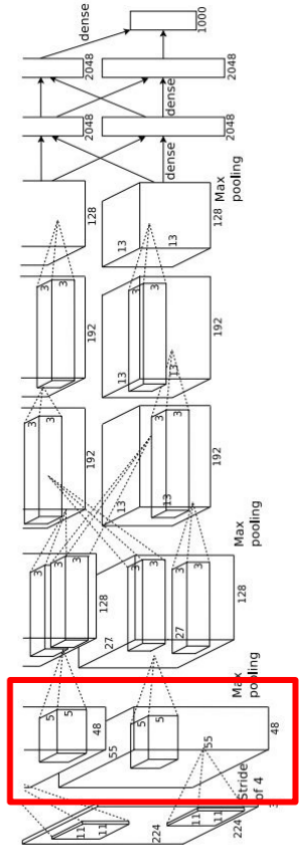


ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

Visualize the filters/kernels (raw weights)

We can visualize filters at higher layers, but not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)

layer 1 weights

16 x 3 x 7 x 7

layer 2 weights

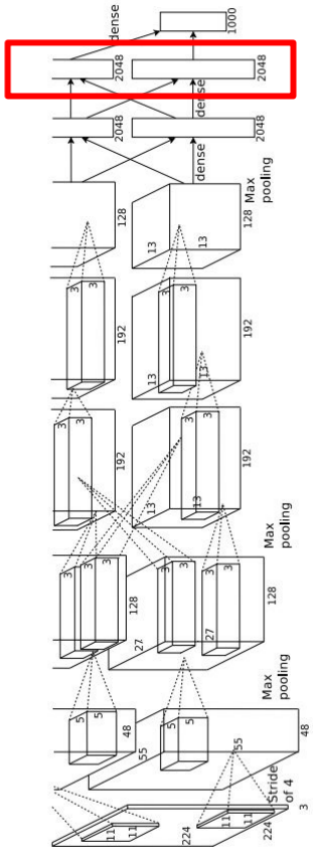
20 x 16 x 7 x 7

layer 3 weights

20 x 20 x 7 x 7

Last Layer

FC7 layer

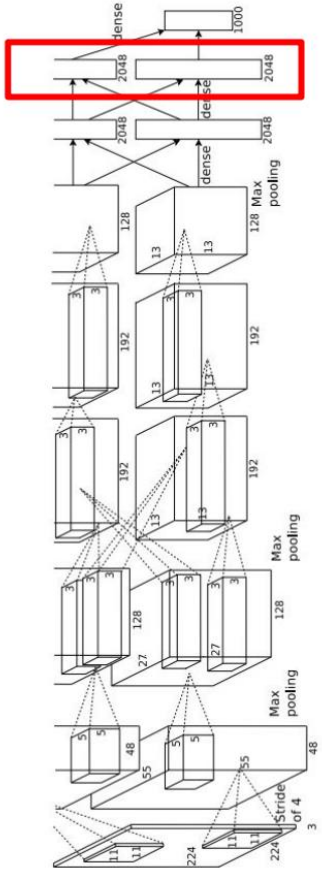


4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors

Last Layer: Nearest Neighbors

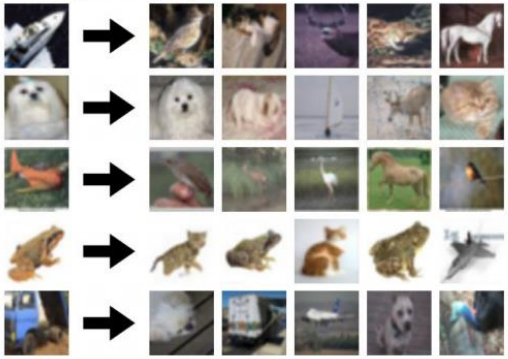
4096-dim vector



Test image

L2 Nearest neighbors in feature space

Recall: Nearest neighbors in pixel space



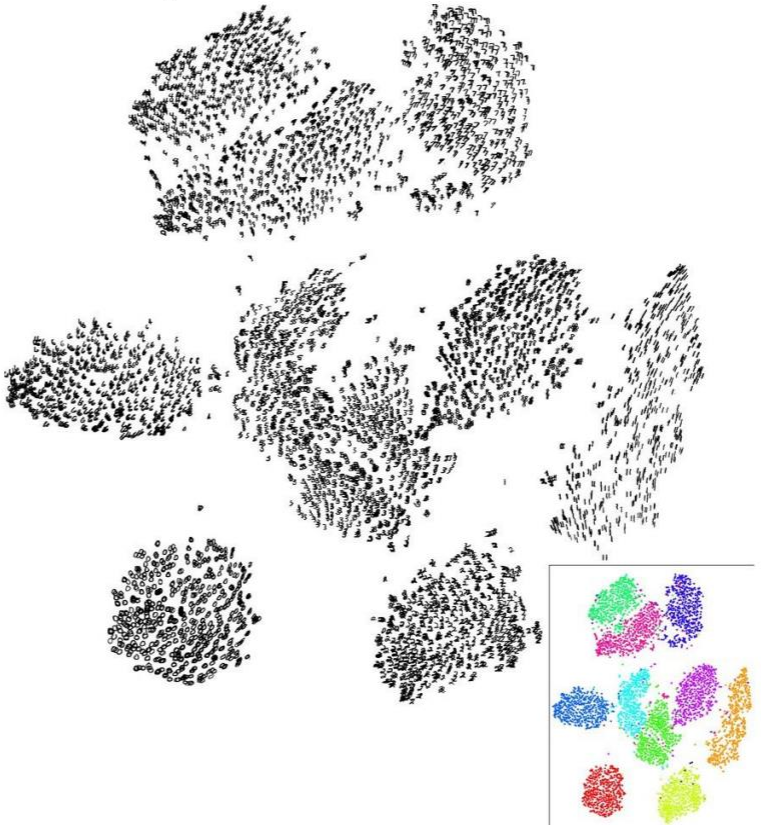
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figures reproduced with permission.

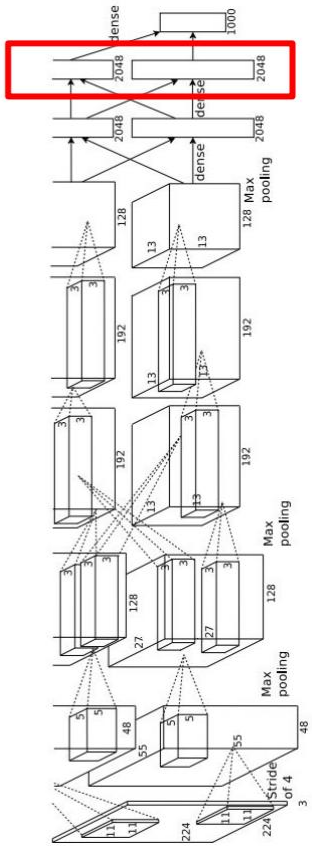
Last Layer: Dimensionality Reduction

Visualize the “space” of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions

Simple algorithm: Principle Component Analysis (PCA)

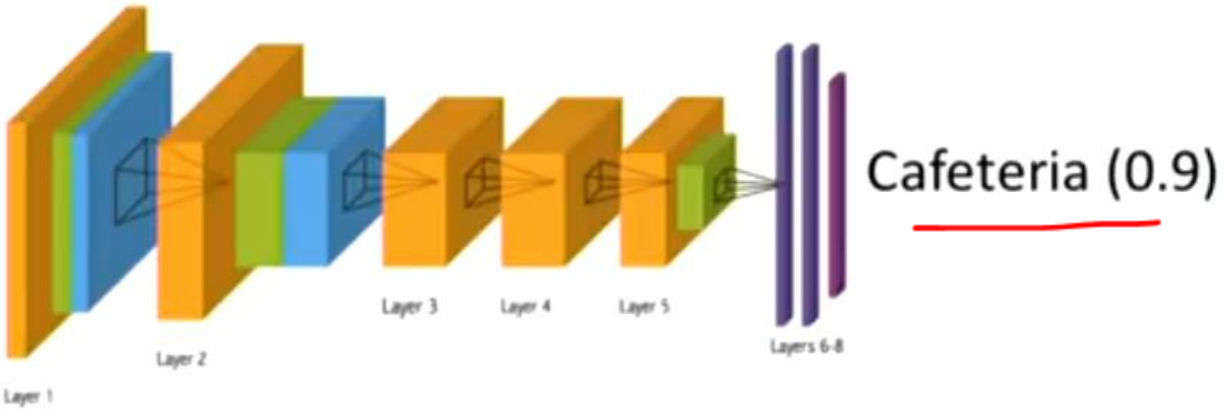
More complex: **t-SNE**





Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

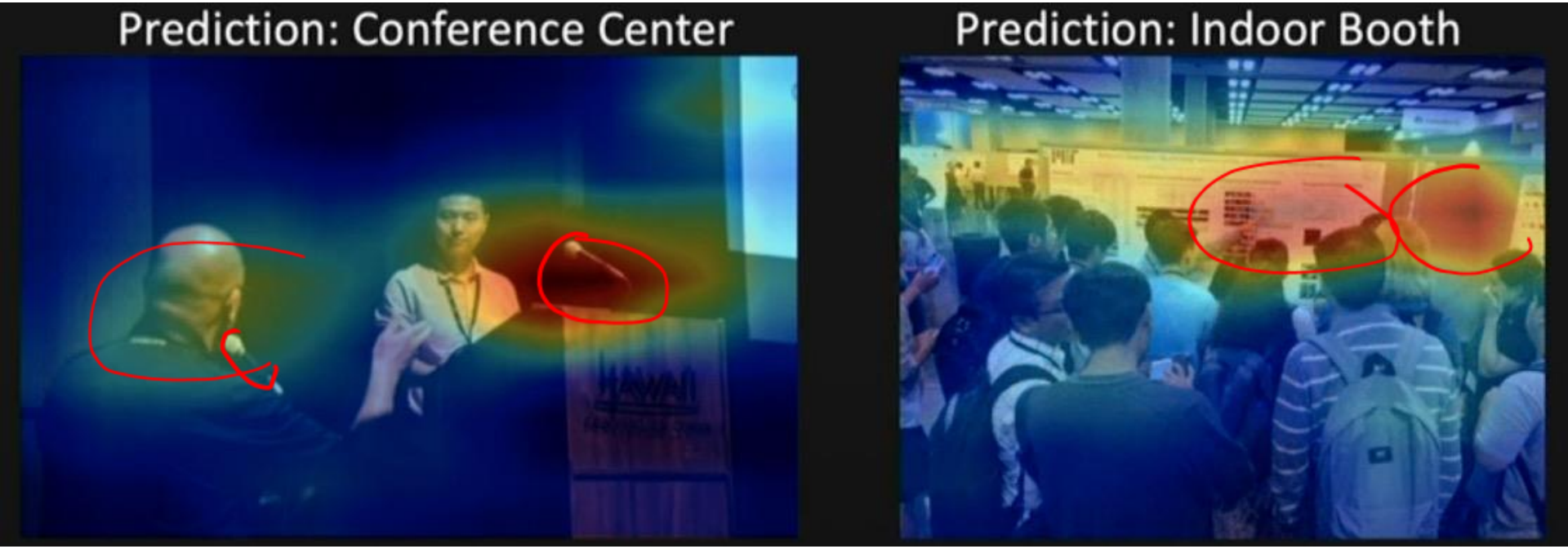
Why the network gives such prediction?



Credit: Bolei Zhou

Class Activation Map

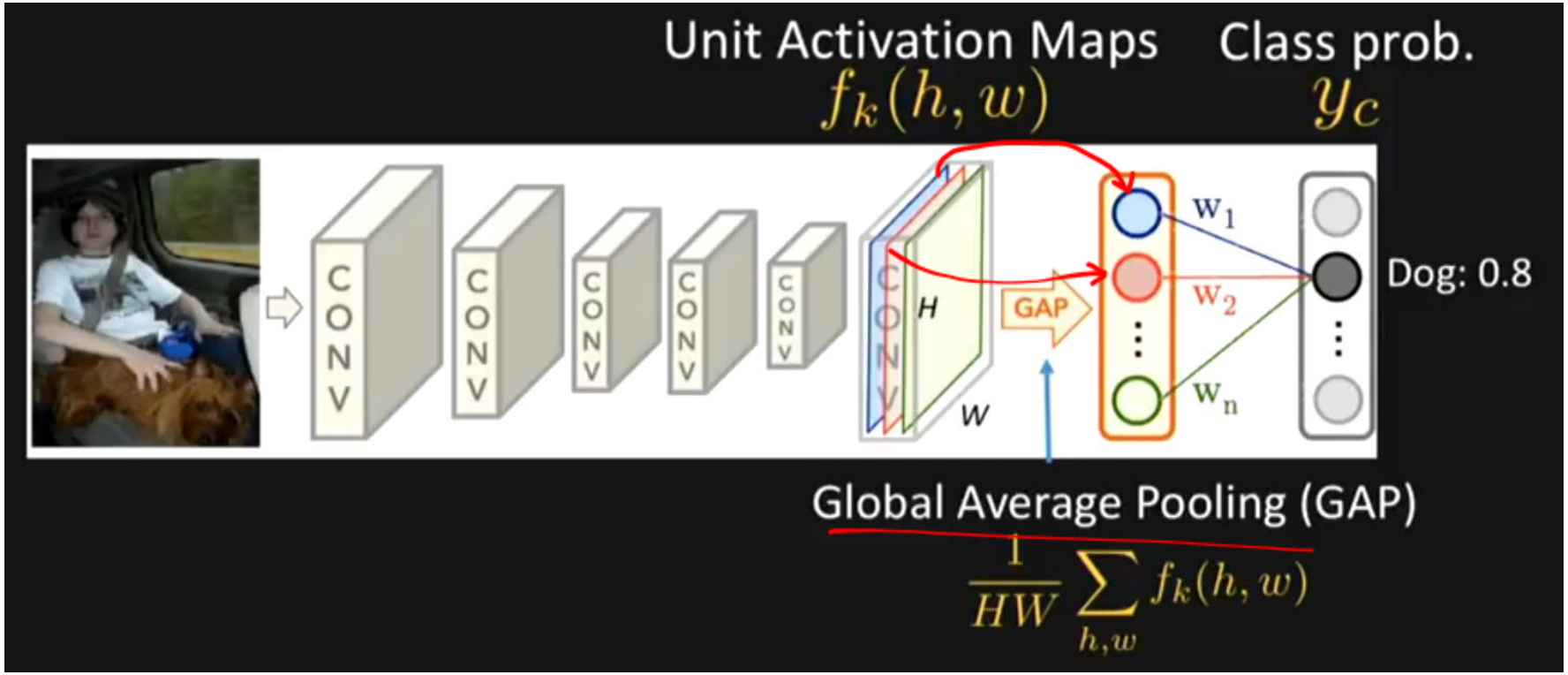
- Explain Prediction of Deep Neural Network



Credit: Bolei Zhou

Zhou, Bolei, et al. "Learning deep features for discriminative localization." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

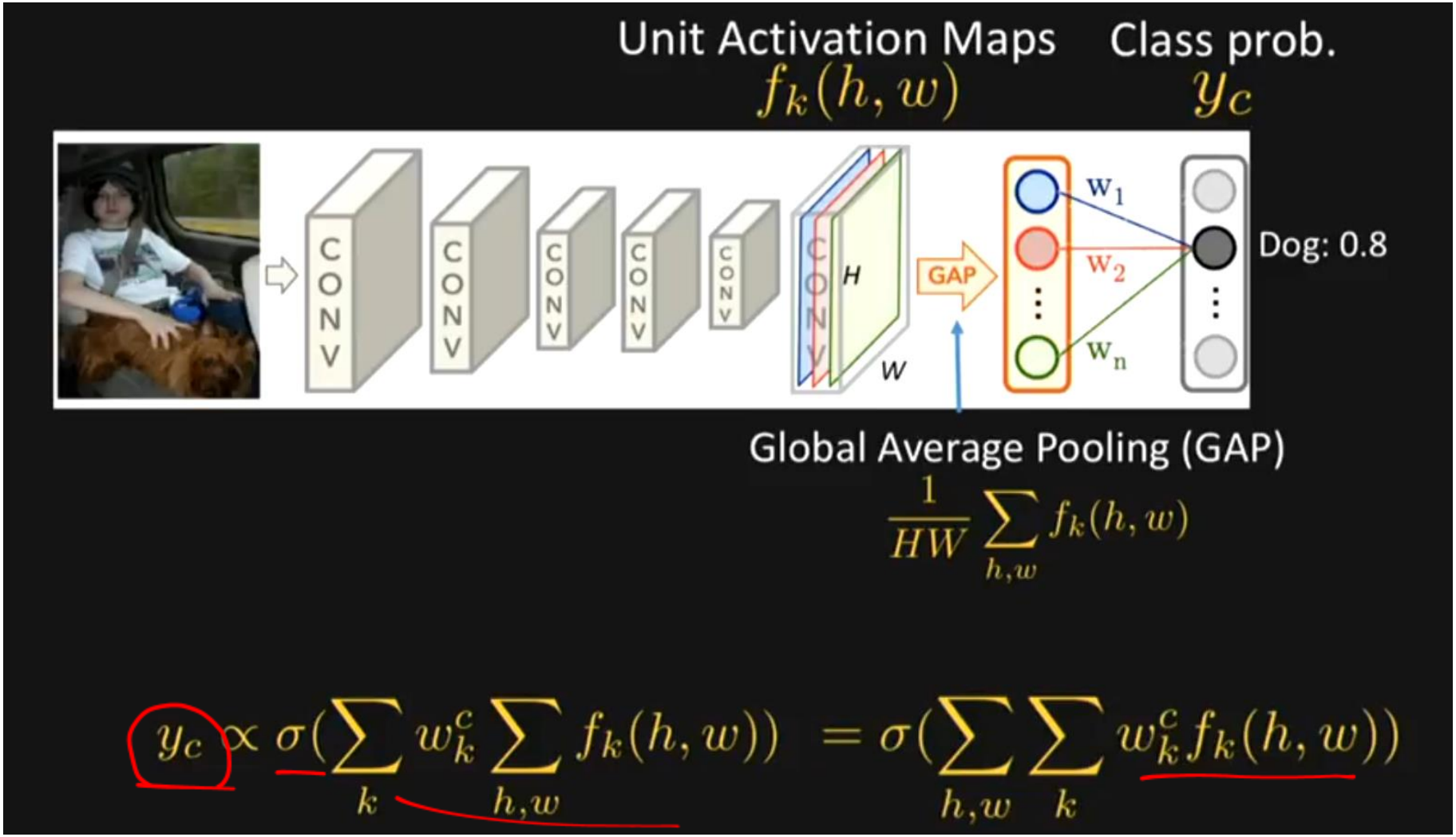
What the CNN is looking



Credit: Bolei Zhou

Zhou, Bolei, et al. "Learning deep features for discriminative localization." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

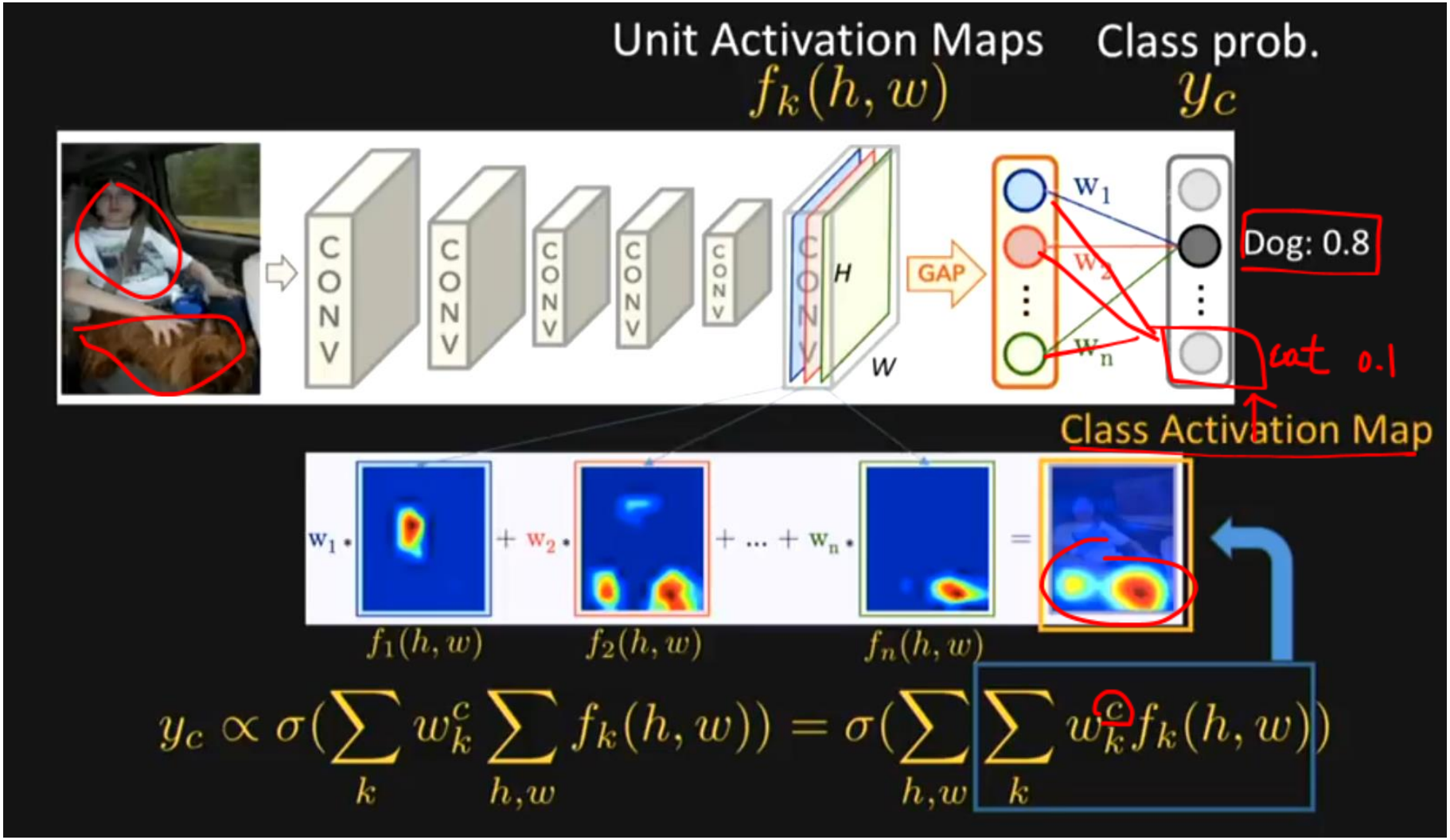
What the CNN is looking



Zhou, Bolei, et al. "Learning deep features for discriminative localization." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Credit: Bolei Zhou

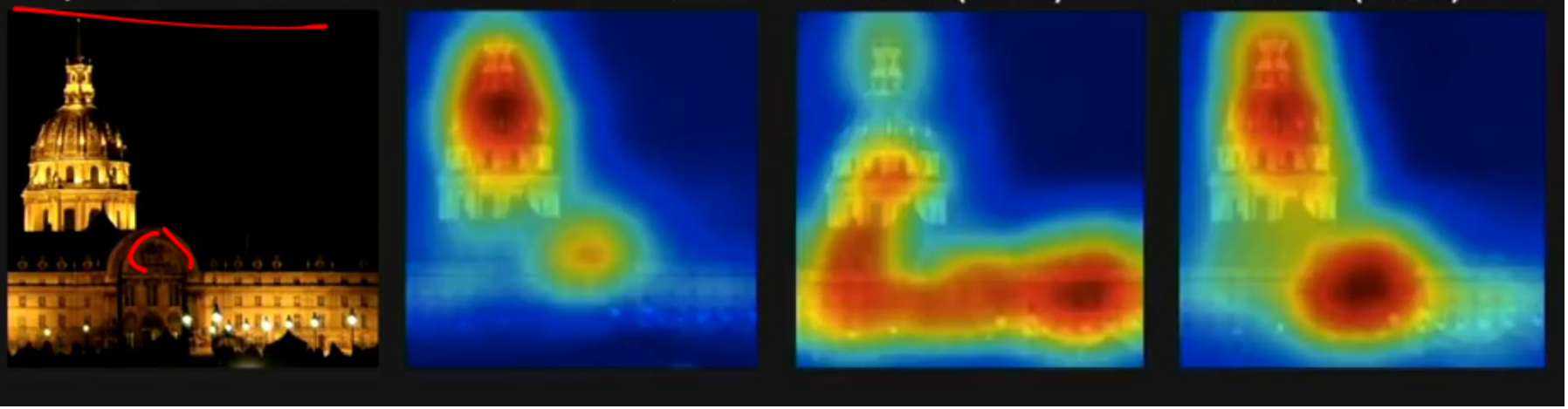
What the CNN is looking



Zhou, Bolei, et al. "Learning deep features for discriminative localization." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Credit: Bolei Zhou

Class Activation Mapping: Explain Prediction of Deep Neural Network



Credit: Bolei Zhou

Credit: Bolei Zhou

Explain the failure cases

GT: House
Prediction: Sushi bar

Credit: Bolei Zhou

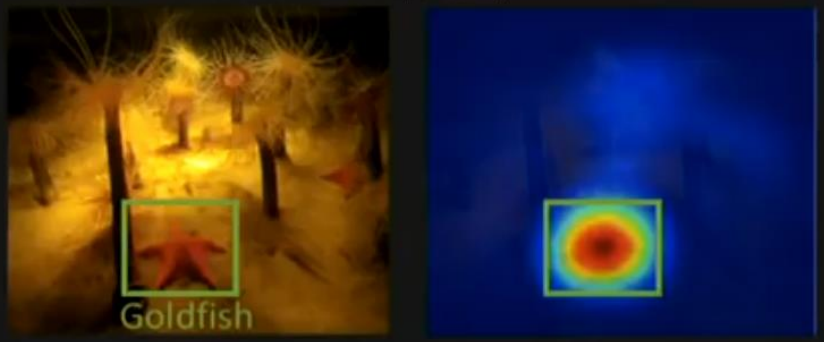
Explain the failure cases

Credit: Bolei Zhou

Using CAM for localization

Evaluation on Weakly-Supervised Localization

Prediction: Starfish (0.83)



Prediction: Tricycle (0.92)

Method	Supervision	Localization Accuracy(%)
Backpropagation	weakly	53.6
Our method	weakly	**62.9**
AlexNet	full	65.8

Result on ImageNet Localization Benchmark

Credit: Bolei Zhou

Limitation of CAM

- To apply CAM, any CNN-based network must change its architecture, where GAP is a must before the output layer
  - i.e., architectural changes and hence re-training is needed

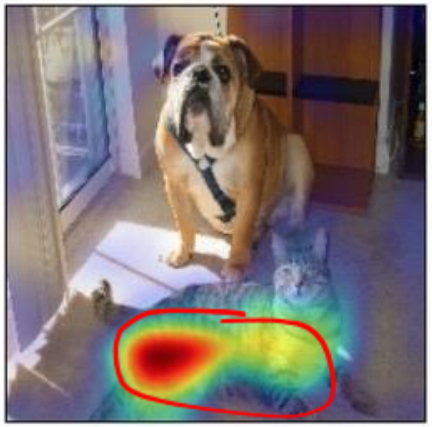
Grad-CAM (Gradient-weighted Class Activation Mapping)

- Grad-CAM (Gradient-weighted Class Activation Mapping) generalizes CAM for a wide variety of CNN-based architectures
  - i.e., without requiring architectural changes or re-training
- Characteristics
  - Without GAP layer, we need a way to define weights - w_k^c
  - Grad-CAM uses the gradients of any target class (e.g., “dog” in a classification network) flowing into the final convolutional layer, and derive summary statistics out of it to represent the weights (importance)

Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." Proceedings of the IEEE International Conference on Computer Vision. 2017.

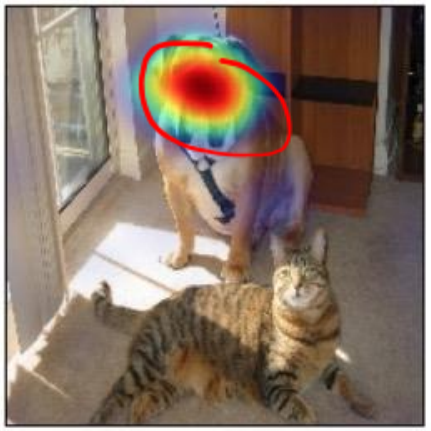
Grad-CAM

(a) Original Image



(c) Grad-CAM 'Cat'

(g) Original Image



(i) Grad-CAM 'Dog'

Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." Proceedings of the IEEE International Conference on Computer Vision. 2017.

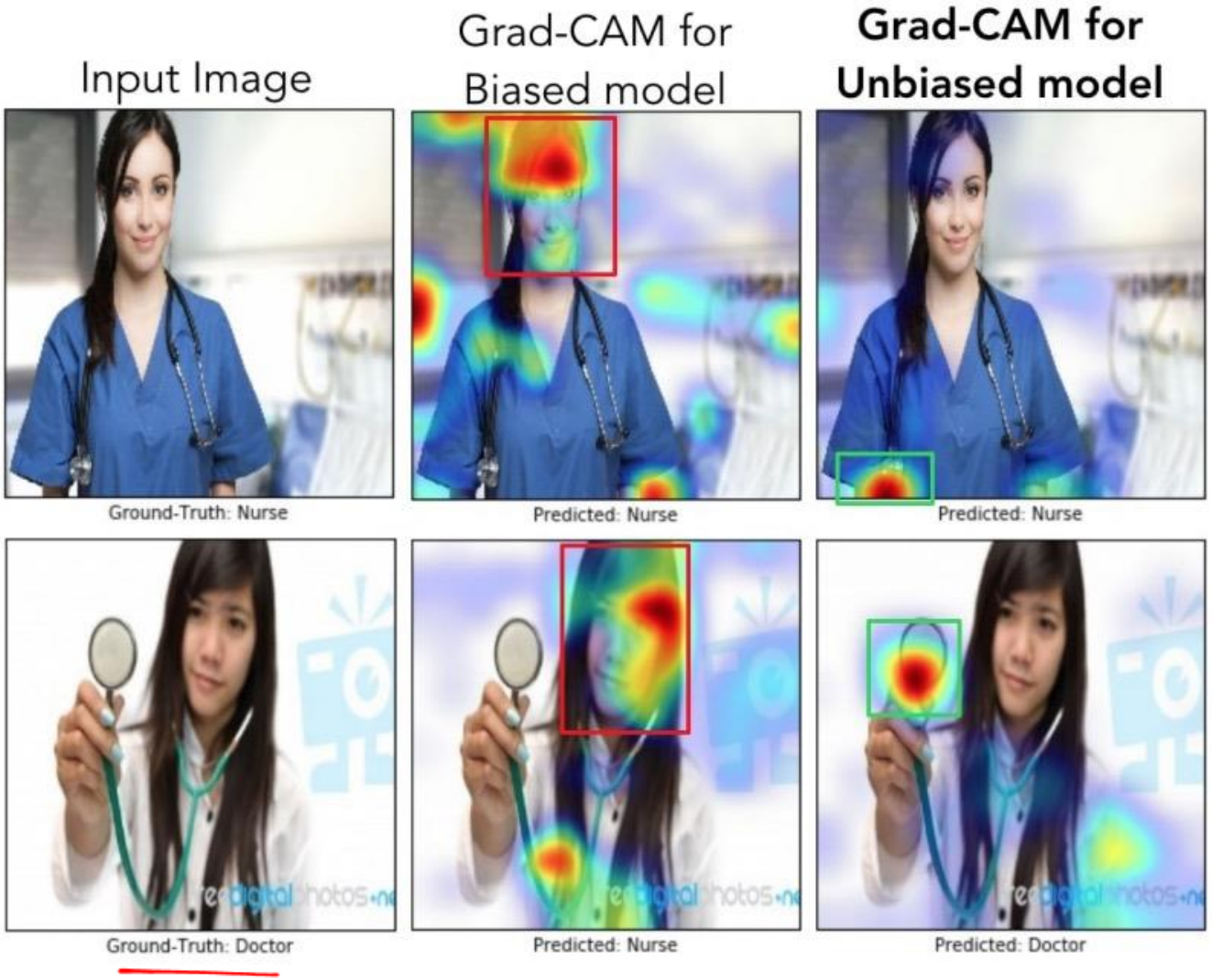
Grad-CAM for Weakly-supervised Localization

		Classification		Localization	
		Top-1	Top-5	Top-1	Top-5
VGG-16	Backprop [51]	30.38	10.89	61.12	51.46
	c-MWP [58]	30.38	10.89	70.92	63.04
	Grad-CAM (ours)	30.38	10.89	**56.51**	46.41
	CAM [59]	33.40	12.20	57.20	**45.14**
AlexNet	c-MWP [58]	44.2	20.8	92.6	89.2
	Grad-CAM (ours)	44.2	20.8	68.3	56.6
GoogleNet	Grad-CAM (ours)	31.9	11.3	60.09	49.34
	CAM [59]	31.9	11.3	60.09	49.34

Table 1: Classification and localization error % on ILSVRC-15 val (lower is better) for VGG-16, AlexNet and GoogleNet. We see that Grad-CAM achieves superior localization errors without compromising on classification performance.

Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." Proceedings of the IEEE International Conference on Computer Vision. 2017.

Grad-CAM for Identifying Bias in Dataset



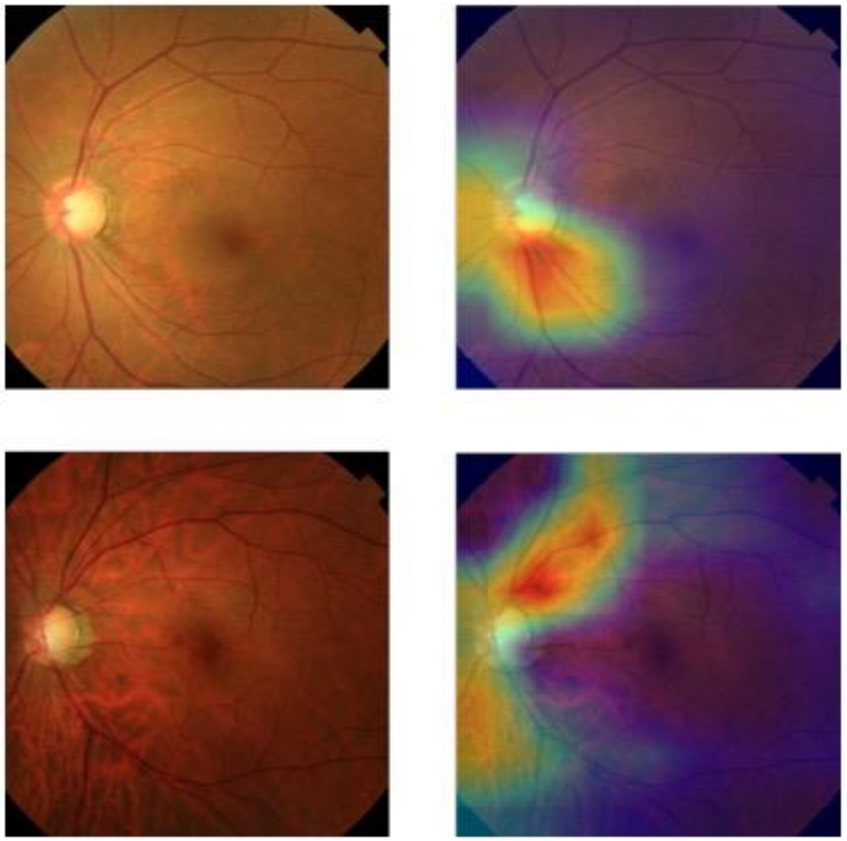
First row:
The biased model (model1) was looking at the face of the person to decide if the person was a nurse, whereas the unbiased model was looking at the short sleeves to make the decision.

Second row:
The biased model made the wrong prediction (misclassifying a doctor as a nurse) by looking at the face and the hairstyle, whereas the unbiased model made the right prediction looking at the white coat, and the stethoscope.

Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." Proceedings of the IEEE International Conference on Computer Vision. 2017.

Grad-CAM for Medical Image Analysis

- Grad-CAM for glaucoma diagnosis and localization



(a) Correctly localized suspicious areas.

Kim, Mijung, et al. "Web applicable computer-aided diagnosis of glaucoma using deep learning." arXiv preprint arXiv:1812.02405 (2018).

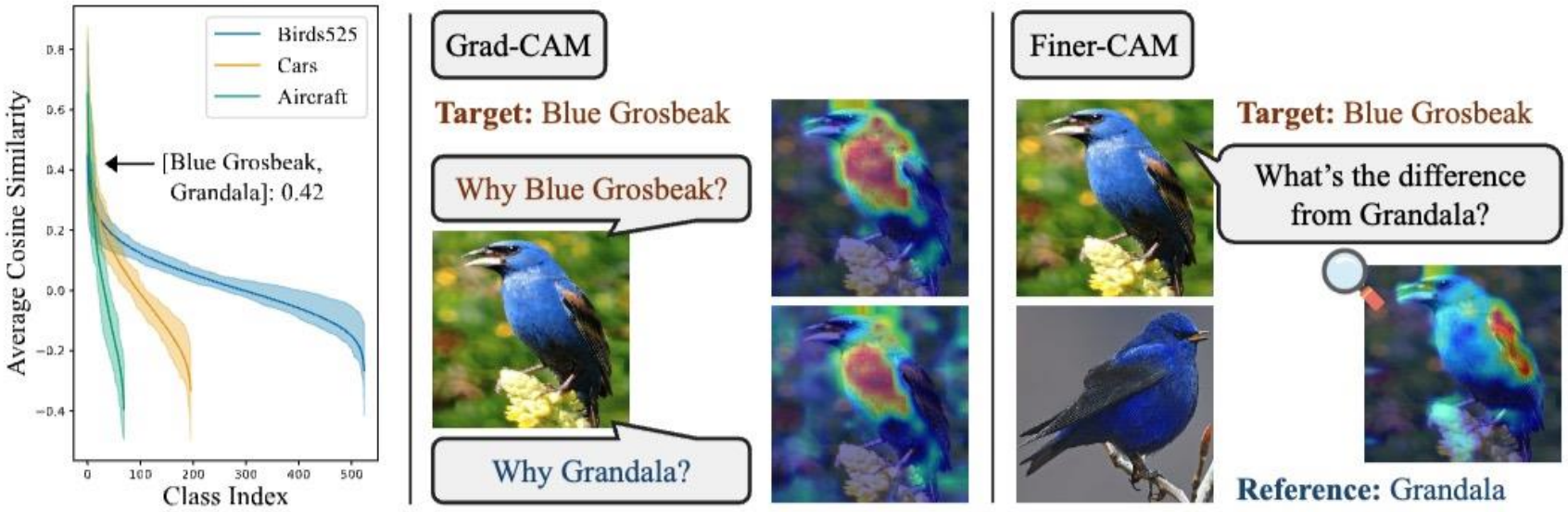


Figure 1. **Illustration of Finer-CAM. Left:** Sorted cosine similarity between the linear classifier weights of one class and the other classes, averaged across all classes. Many pairs of classes are highly similar, yet neural networks can effectively distinguish them to achieve high fine-grained classification accuracy. **Middle**: Standard CAM methods highlight main regions contributing to the target class's logit value, inadvertently including regions predictive of similar classes and overshadowing fine discriminative details. **Right**: We propose Finer-CAM to explicitly compare the target class with similar classes and spot the difference, enabling accurate localization of discriminative details.

Zhang et al. Finer-CAM: Fine-grained Visual Interpretability through Class-Specific Gradient Refinements
https://arxiv.org/pdf/2501.11309

More reading

- Bau, David, et al. "Network dissection: Quantifying interpretability of deep visual representations." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

- Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

- Chattopadhay, Aditya, et al. "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks." *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018.

- Zhang et al. Finer-CAM: Fine-grained Visual Interpretability through Class-Specific Gradient Refinements https://arxiv.org/pdf/2501.11309

Implementation

- CAM
  - https://github.com/zhoubolei/CAM
- Grad-CAM
  - https://github.com/ramprs/grad-cam/
  - Demo: http://gradcam.cloudcv.org/
- TorchCAM: class activation explorer
  - https://github.com/frgfm/torch-cam

Thank you!

Question?

References and Slide Credits

- Many slides are adapted from the existing teaching or tutorial slides by Hung-yi Lee, Andrew Ng, Alexander Amini, Lex Fridman, and Stanford course - CS231n: Convolutional Neural Networks for Visual Recognition
- Special thanks to Dr. Hung-yi Lee for making his machine learning course slides and materials available
- Alexander Amini, MIT 6.S191 Introduction to Deep Learning: http://introtodeeplearning.com/
  - Youtube videos: https://www.youtube.com/watch?v=5tvmMX8r_OM&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=1
- Lex Fridman, MIT Deep Learning and Artificial Intelligence Lectures: https://deeplearning.mit.edu/ https://www.youtube.com/watch?v=O5xeyoRL95U