deep imaging

Lecture 13: Beyond classification – visualization to object detection

Machine Learning and Imaging

BME 548L
Roarke Horstmeyer

Class project details

- Due this Friday at **11pm**
- Full details are here: https://deepimaging.github.io/proj-info/
- Can work alone or in a group (up to 4 people), required effort will scale with # of people
- Select a "base" dataset (online, or from a list I'll make)
- Simulate parameters of a physical (imaging) system with base dataset
- Train deep neural net with simulated dataset
- Report results

What you'll need to submit:

1) The project's source code

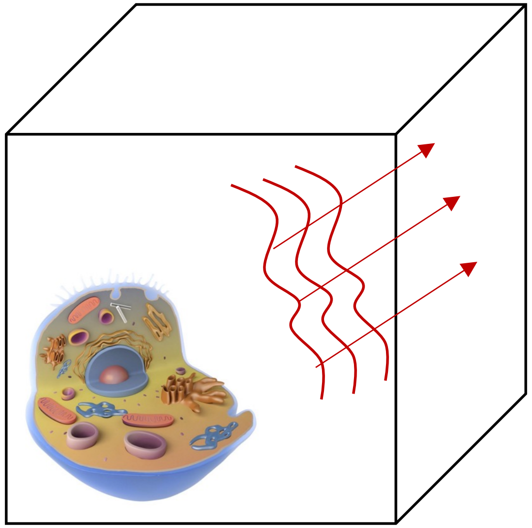
2) A short research-style paper (3 pages minimum, 5 pages maximum) that includes an introduction, results, a discussion section, references and at least 2 figures

3) A completed web template containing the main results from the research paper

4) A 7-10 minute presentation that each student will deliver to the class

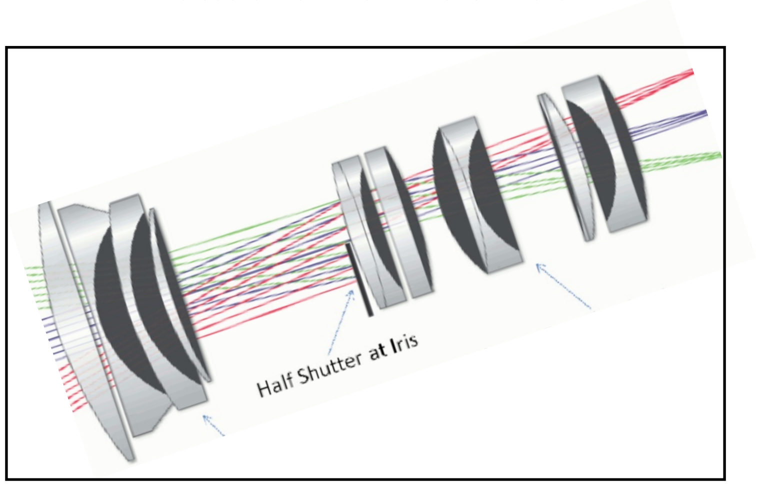
ML+Imaging pipeline + plan

Real World



Continuous
complex fields

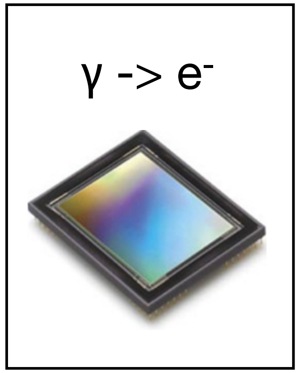
Measurement device



Black box transformations

- Convolution
- Fourier Transform

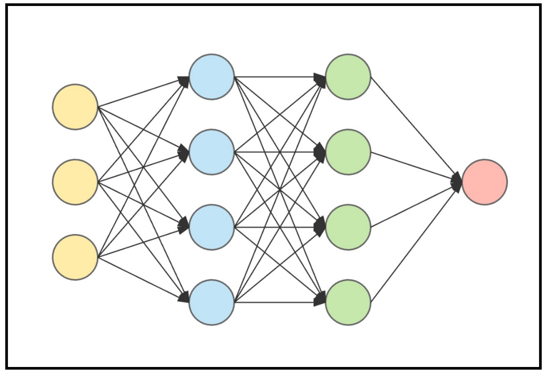
Digitization



Sampling Theorem

Discrete math &
Linear algebra

Machine Learning



Optimization

Linear classification

Logistic classifier

Neural networks

Convolutional NN’s

This week

After break

End of Class

How to examine and present your results: a few options at different stages

Options to examine your test data after processing:

- ROC curve, Precision-Recall
- Confusion matrix
- Sliding window visualization
- Layer visualizations
- Saliency maps etc.
- tSNE visualization

ROC curve and confusion matrix

- Can set threshold for f(x,W) wherever
- Leads to sliding window between FN and FP rate
- Need to summarize both statistics as a function of sliding window

Estimated label

f(x, W)

		+1	-1
Actual label y	+1	True positive	False negative
	-1	False positive	True negative

Missed an event

Predict event when there isn’t one

ROC curve and confusion matrix

TP Rate =

Sensitivity = TP / (TP + FN) = TP / Actual positives

False Positive Rate = FP / (TN + FP) = FP/ Actual negatives

Specificity = TN / (TN + FP) = TN / Actual negatives
= 1 – False Positive Rate

		Estimated label $f(x, W)$	
		+1	-1
Actual label y	+1	True positive	False negative
	-1	False positive	True negative

Missed an event

Predict event when there isn't one

ROC curve and confusion matrix

TP Rate =

Sensitivity = TP / (TP + FN) = TP / Actual positives

False Positive Rate = FP / (TN + FP) = FP/ Actual negatives

Specificity = TN / (TN + FP) = TN / Actual negatives
= 1 – False Positive Rate

Estimated label

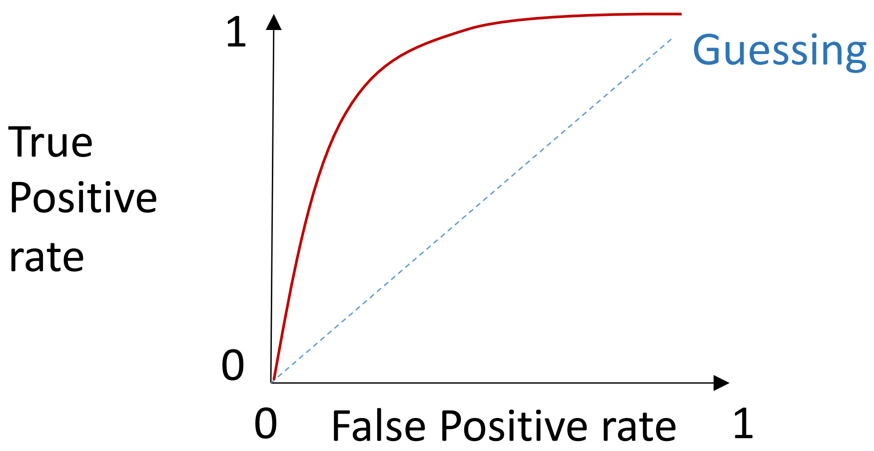
$f(x, W)$

		+1	-1
Actual label y	+1	True positive	False negative
	-1	False positive	True negative

Missed an event

Predict event when there isn't one

Receiver-Operator Curve



ROC curve and confusion matrix

		Estimated label f(x, W)	
		+1	-1
Actual label y	+1	True positive	False negative
	-1	False positive	True negative

Missed an event

Predict event when there isn't one

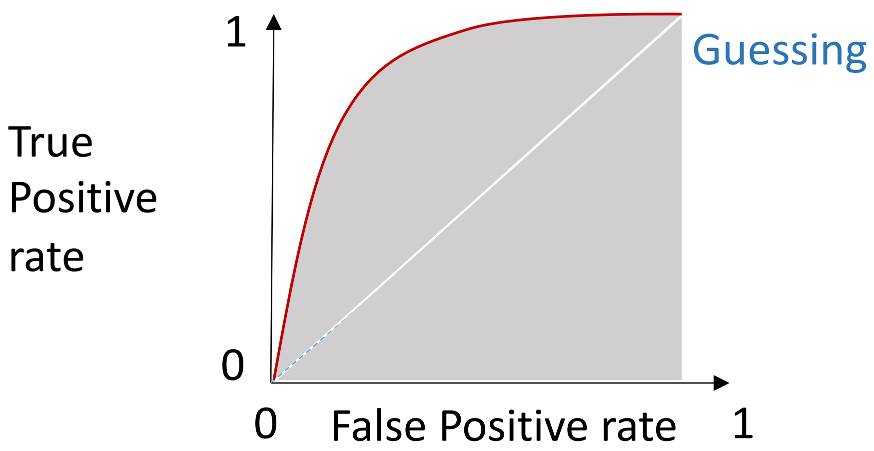
TP Rate =

Sensitivity = TP / (TP + FN) = TP / Actual positives

False Positive Rate = FP / (TN + FP) = FP/ Actual negatives

Specificity = TN / (TN + FP) = TN / Actual negatives
= 1 – False Positive Rate

Receiver-Operator Curve



Area under the curve (AUC): Integral of ROC curve

ROC curve and confusion matrix

Recall =
Sensitivity = TP / (TP + FN) = TP / Actual positives →

Estimated label

f(x, W)

Actual label y		+1	-1
	+1	True positive	False negative
	-1	False positive	True negative

Missed an event

Predict event when
there isn't one

Precision = TP / (TP + FP) = TP / Estimated positives

- Sometimes, you don't care about true negatives (just want to find events)
- In this case, use Precision and Recall

ROC curve and confusion matrix

Recall =
Sensitivity = TP / (TP + FN) = TP / Actual positives →

Estimated label
$f(x, W)$

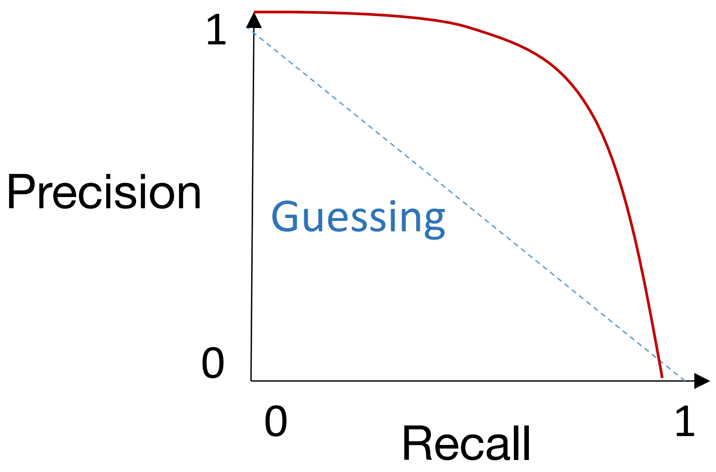
		+1	-1
Actual label y	+1	True positive	False negative
	-1	False positive	True negative

Missed an event

Predict event when there isn't one

Precision = TP / (TP + FP) = TP / Estimated positives

Precision-Recall curve



F1 Metric: $(1/\text{precision} + 1/\text{recall})^{-1}$

ROC curve and confusion matrix

Just 2 categories

Estimated label

f(x, W)

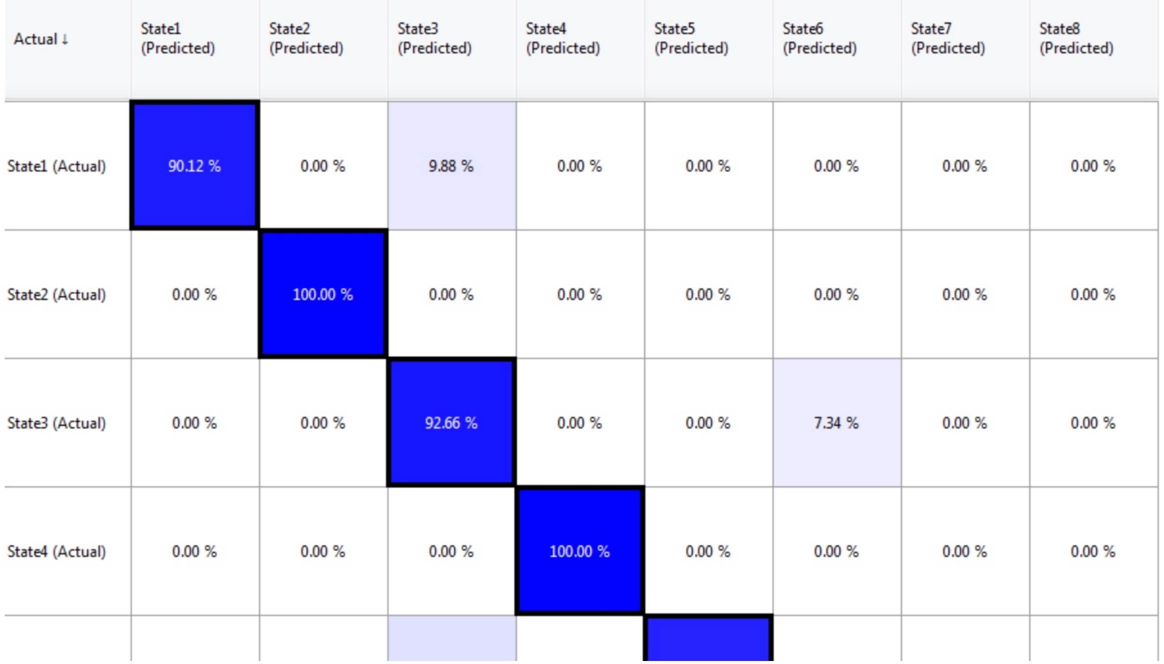
		+1	-1
y	+1	True positive	False negative
	-1	False positive	True negative

Confusion Matrix: 2+ categories

Estimated label

f(x, W)

y



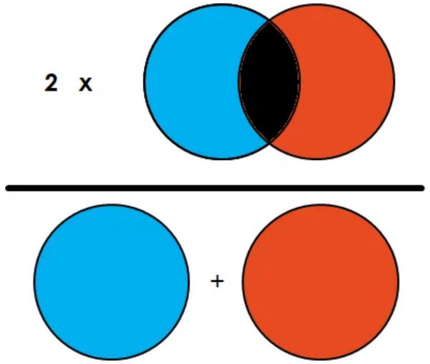
Actual ↓	State1 (Predicted)	State2 (Predicted)	State3 (Predicted)	State4 (Predicted)	State5 (Predicted)	State6 (Predicted)	State7 (Predicted)	State8 (Predicted)
State1 (Actual)	90.12 %	0.00 %	9.88 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
State2 (Actual)	0.00 %	100.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
State3 (Actual)	0.00 %	0.00 %	92.66 %	0.00 %	0.00 %	7.34 %	0.00 %	0.00 %
State4 (Actual)	0.00 %	0.00 %	0.00 %	100.00 %	0.00 %	0.00 %	0.00 %	0.00 %

Other performance metrics

- Overlap between segmented areas: Jaccard similarity coefficient
  - (also called Intersection over Union, IoU)

$$J = |R1 \cap R2| / |R1 \cup R2|$$

- Dice Coefficient (F1 score): 2 x (total area of overlap) / total number of pixels in both images

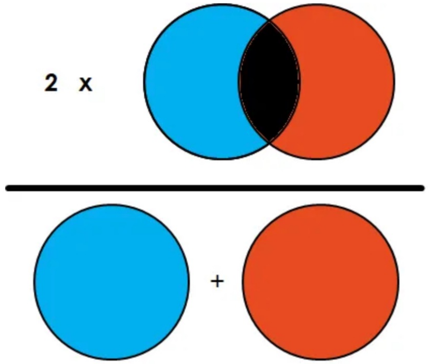


Other performance metrics

- Overlap between segmented areas: Jaccard similarity coefficient
  - (also called Intersection over Union, IoU)

$$J = |R1 \cap R2| / |R1 \cup R2|$$

- Dice Coefficient (F1 score): 2 x (total area of overlap) / total number of pixels in both images



- MSE, PSNR

- Structural Similarity (SSIM)

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

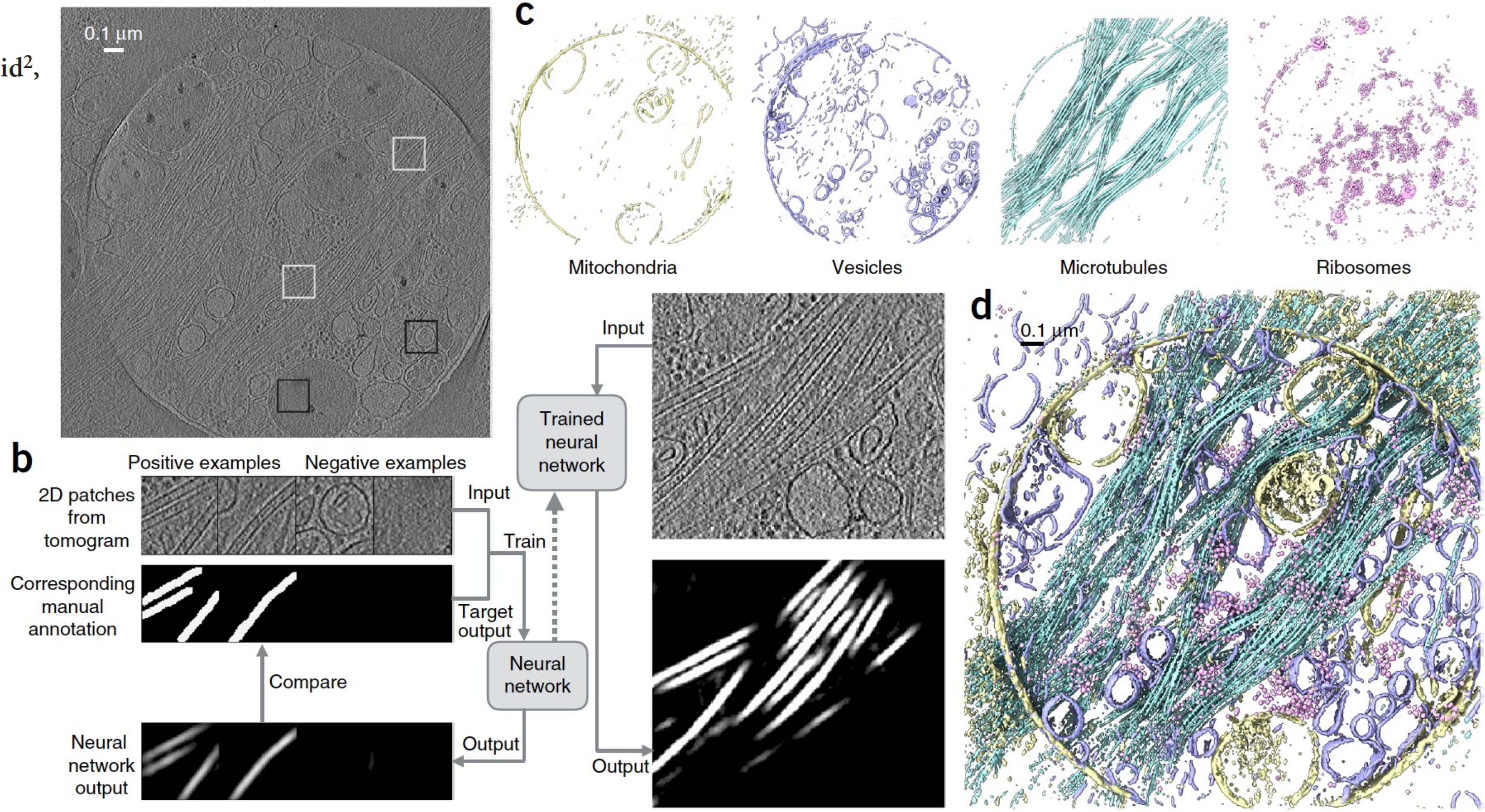
with:

- μ_x the average of x;
- μ_y the average of y;
- σ_x^2 the variance of x;
- σ_y^2 the variance of y;
- σ_{xy} the covariance of x and y;
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator;
- L the dynamic range of the pixel-values (typically this is $2^{\#bits\ per\ pixel} - 1$);
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

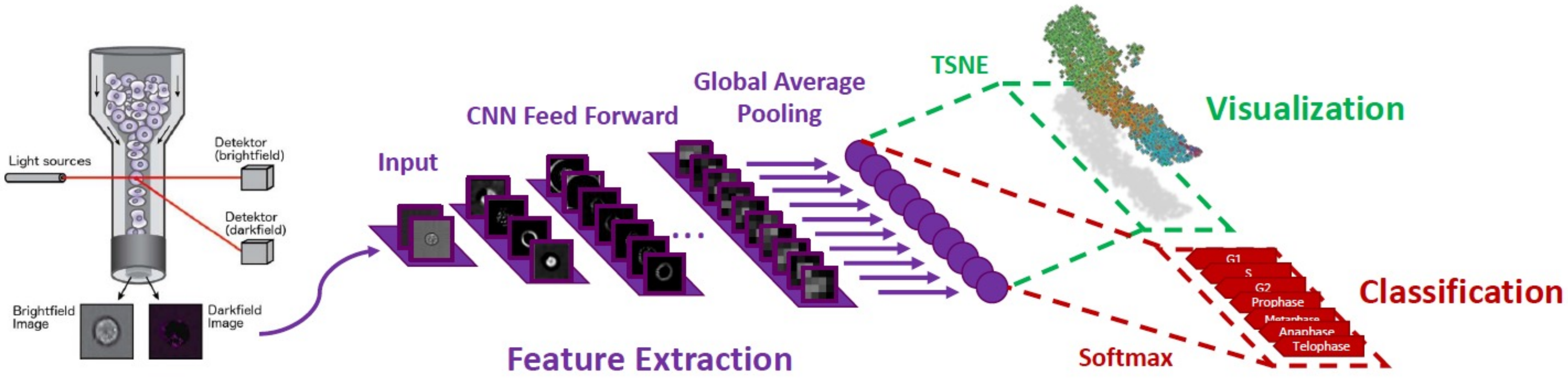
Examples of CNN’s for biomedical image analysis

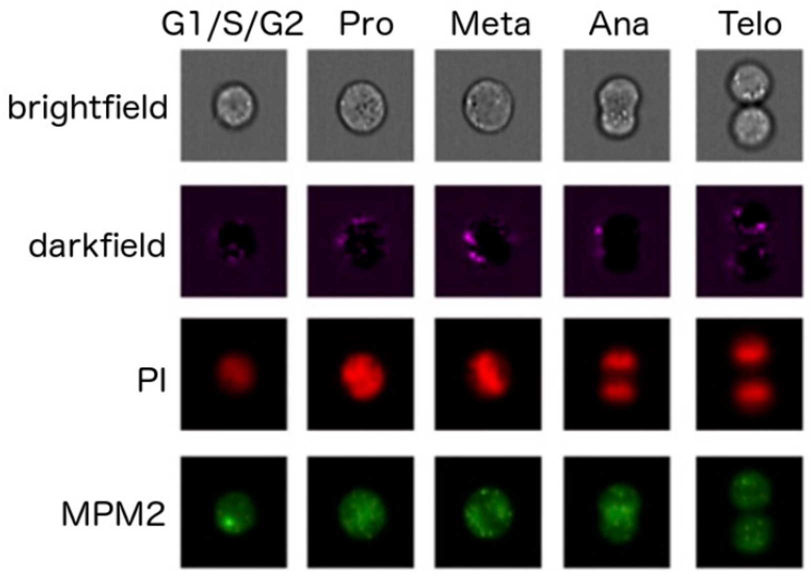
Convolutional neural networks for automated annotation of cellular cryo-electron tomograms

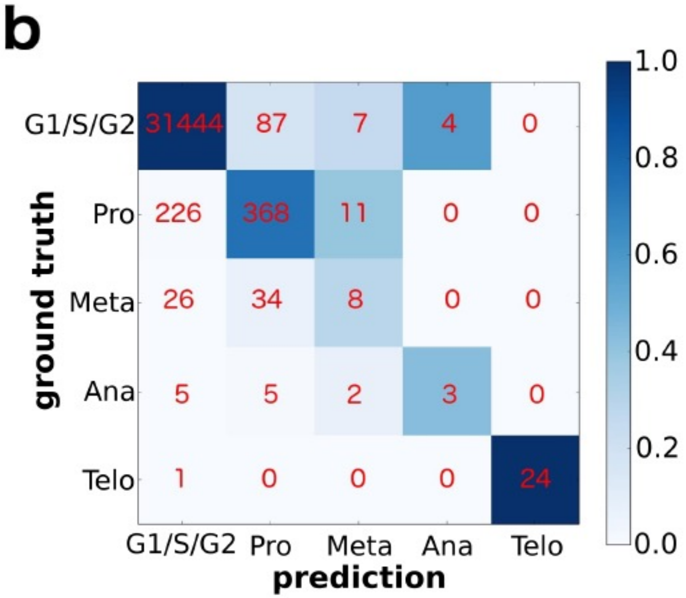
Muyuan Chen[1,2], Wei Dai[2,4], Stella Y Sun[2], Darius Jonasch[2], Cynthia Y He[3], Michael F Schmid[2], Wah Chiu[2] & Steven J Ludtke[2]



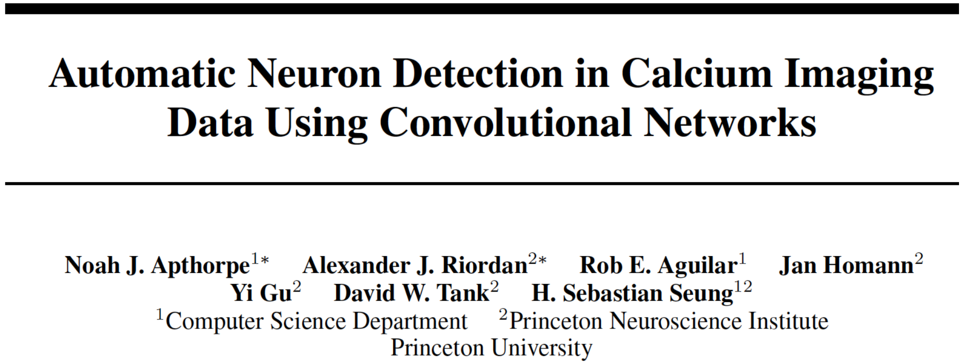
P. Eulenberg et al., “Reconstructing cell cycle and disease progression using deep learning”





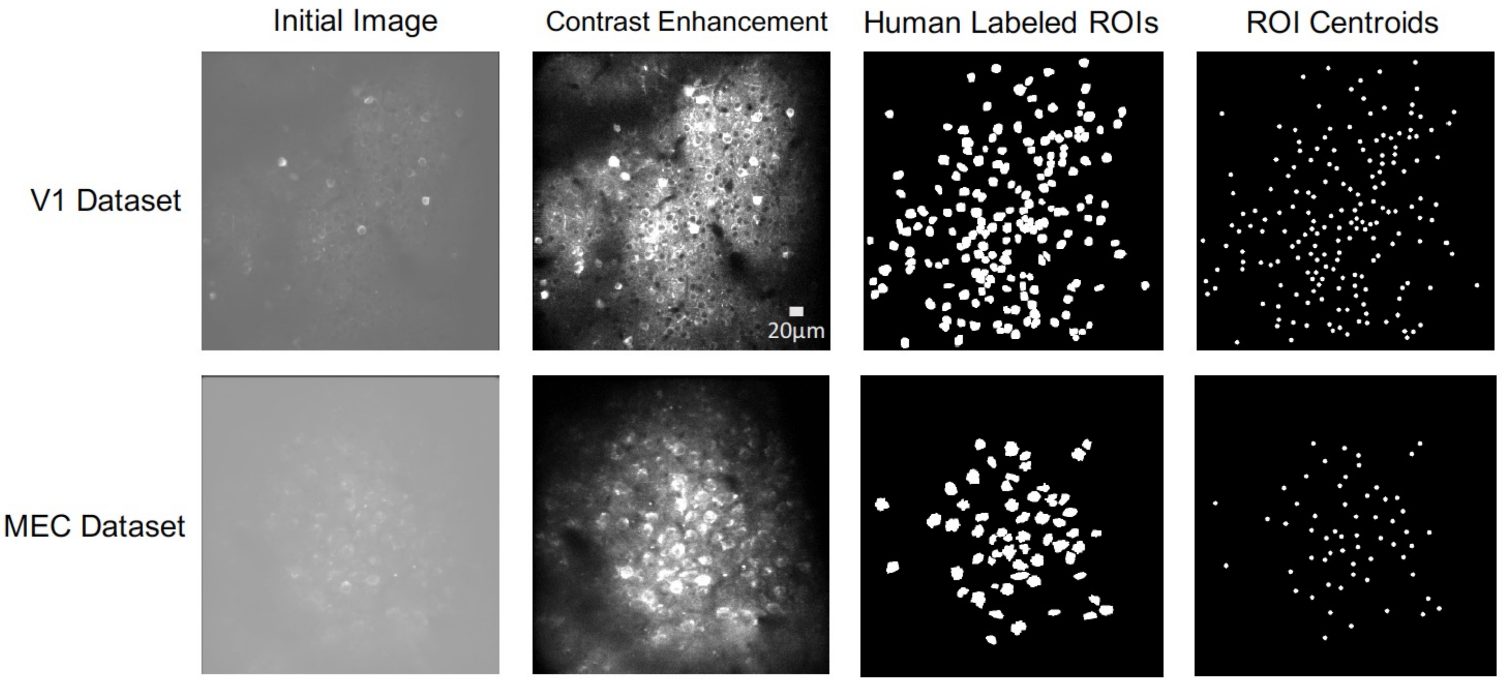


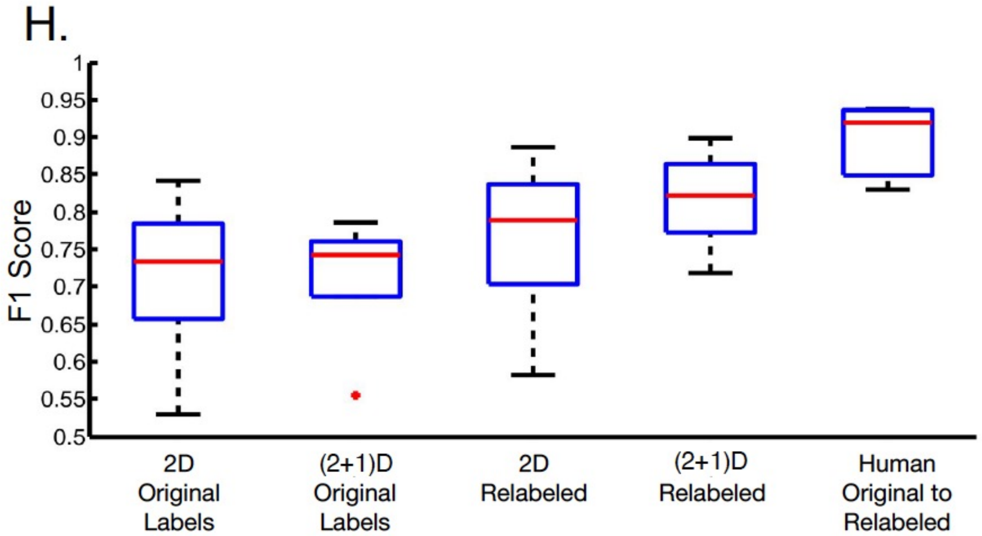
	G1/S/G2	Pro	Meta	Ana	Telo
G1/S/G2	31444	87	7	4	0
Pro	226	368	11	0	0
Meta	26	34	8	0	0
Ana	5	5	2	3	0
Telo	1	0	0	0	24

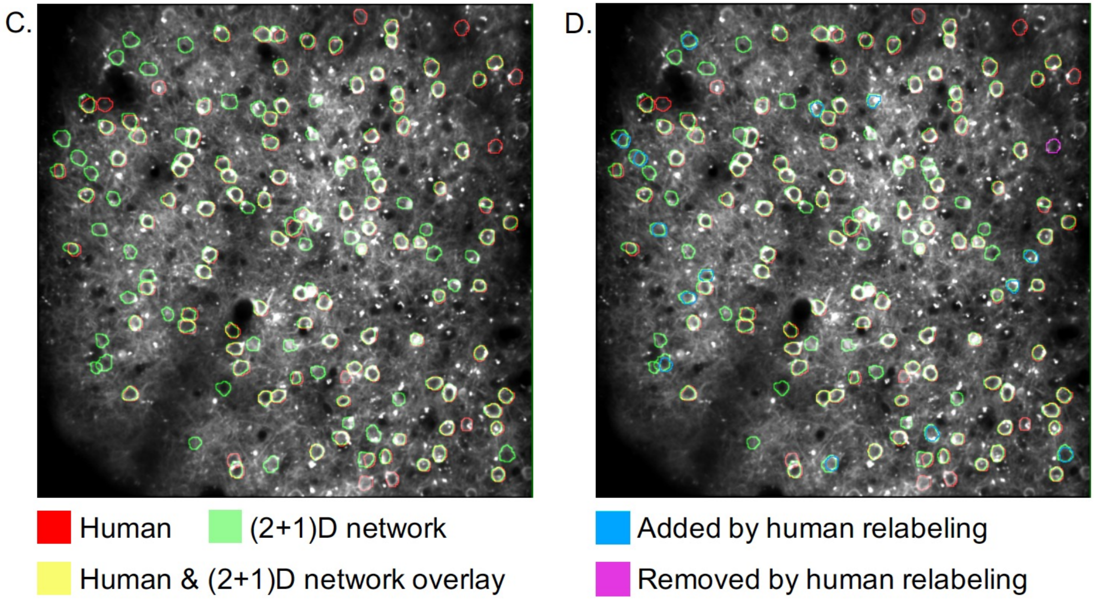


Automatic Neuron Detection in Calcium Imaging Data Using Convolutional Networks

Noah J. Apthorpe[1*] **Alexander J. Riordan**[2*] **Rob E. Aguilar**[1] **Jan Homann**[2]
Yi Gu[2] **David W. Tank**[2] **H. Sebastian Seung**[12]
[1]Computer Science Department [2]Princeton Neuroscience Institute
Princeton University

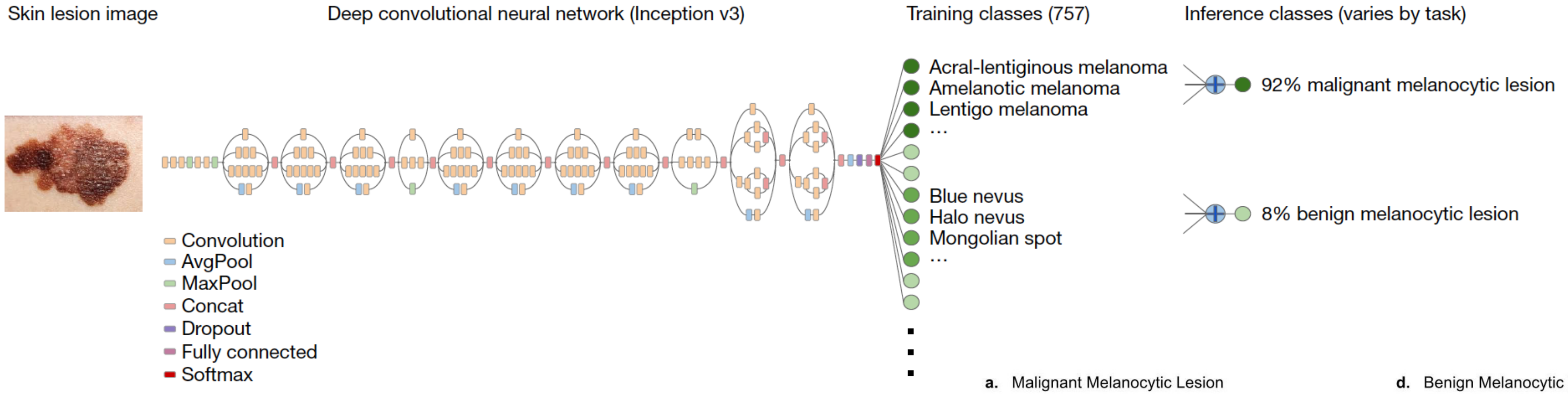


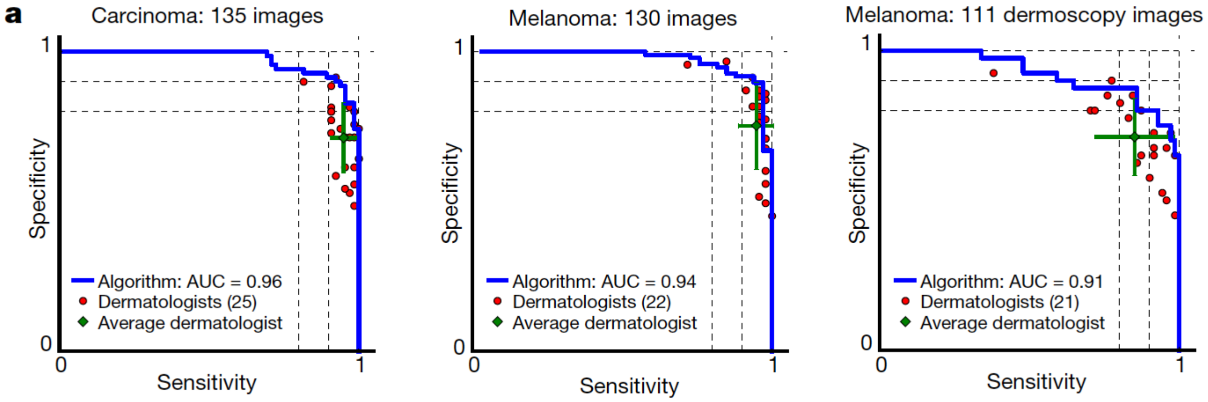


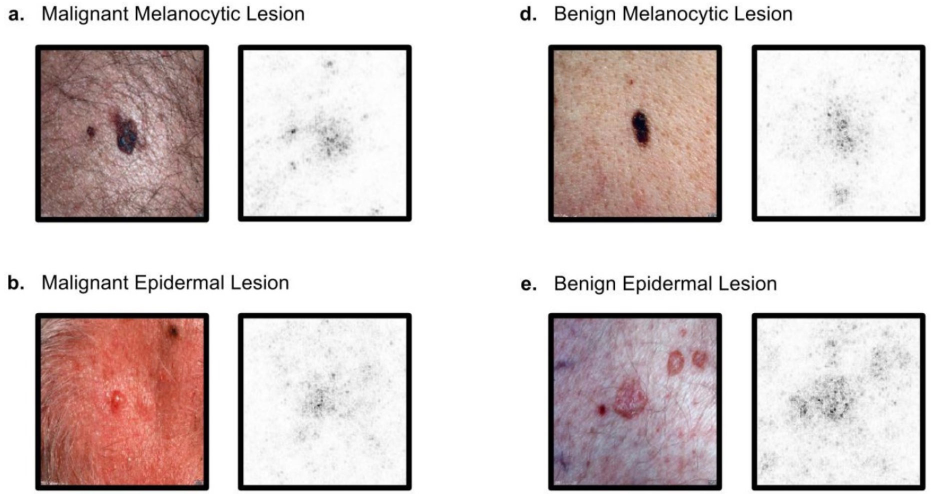


Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva[1]*, Brett Kuprel[1]*, Roberto A. Novoa[2,3], Justin Ko[2], Susan M. Swetter[2,4], Helen M. Blau[5] & Sebastian Thrun[6]

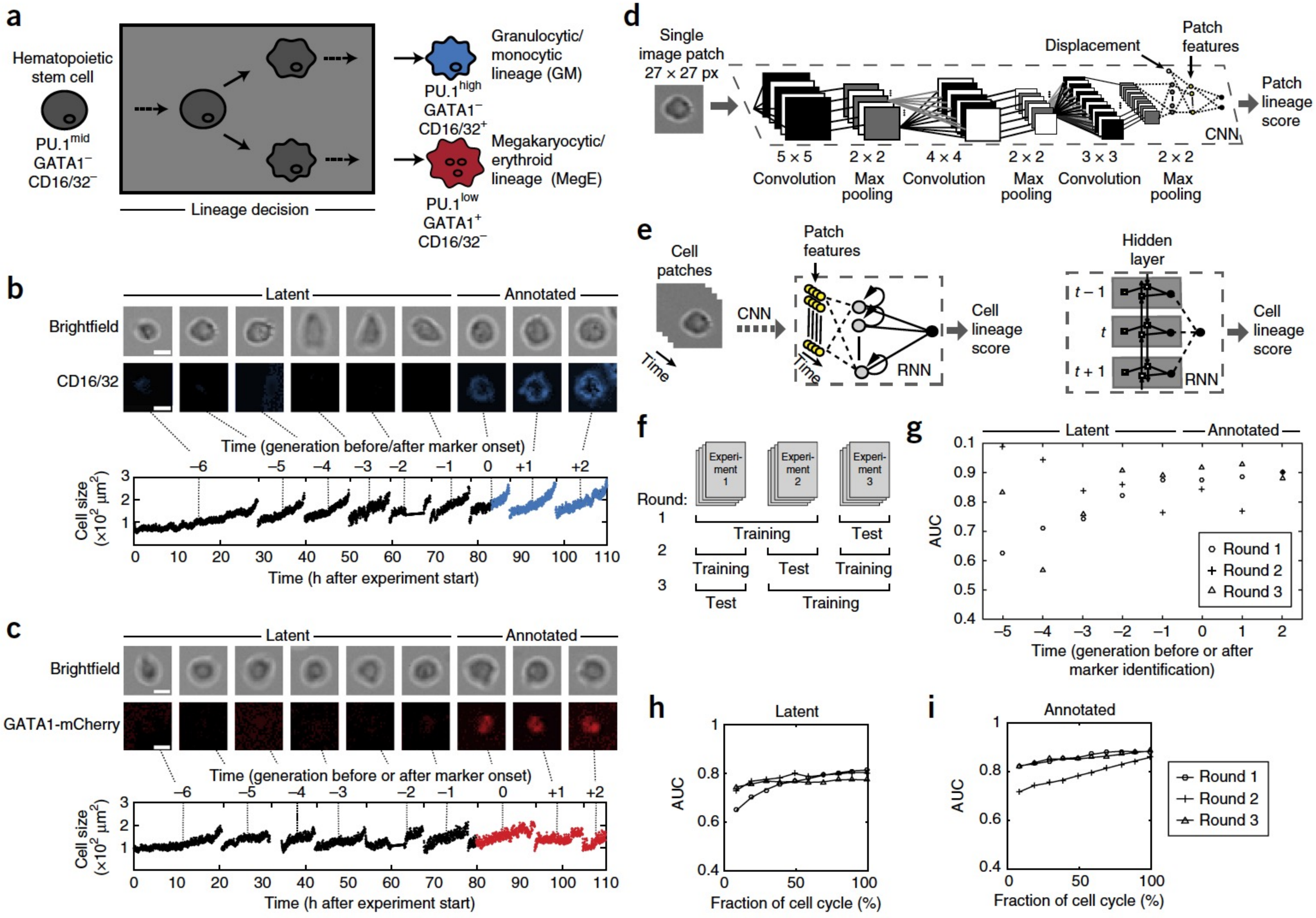






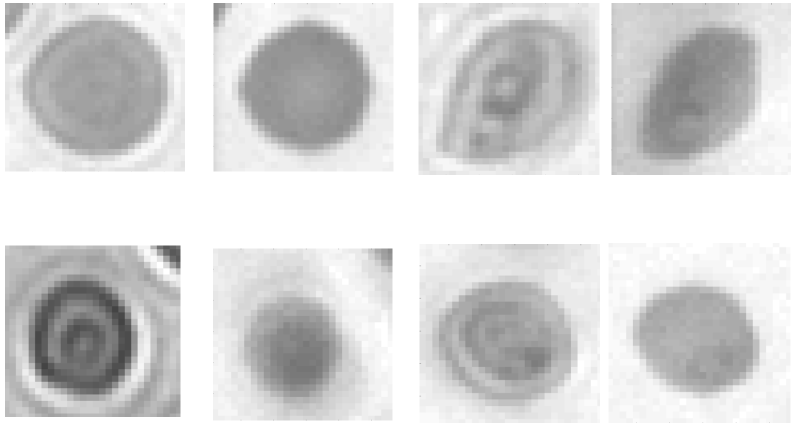
Prospective identification of hematopoietic lineage choice by deep learning

Felix Buggenthin[1,6], Florian Buettner[1,2,6], Philipp S Hoppe[3,4], Max Endele[3], Manuel Kroiss[1,5], Michael Strasser[1], Michael Schwarzfischer[1], Dirk Loeffler[3,4], Konstantinos D Kokkaliaris[3,4], Oliver Hilsenbeck[3,4], Timm Schroeder[3,4], Fabian J Theis[1,5] & Carsten Marr[1]



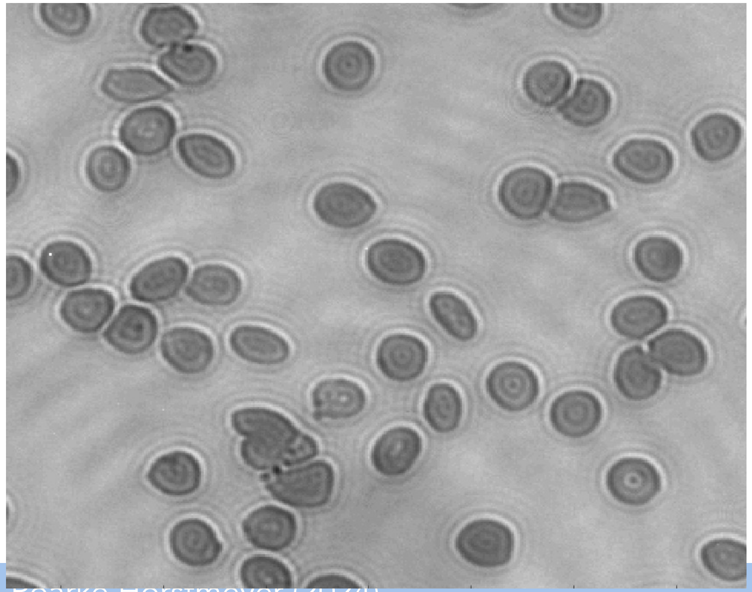
Beyond statistics, how can we visualize performance for classification?

Training dataset



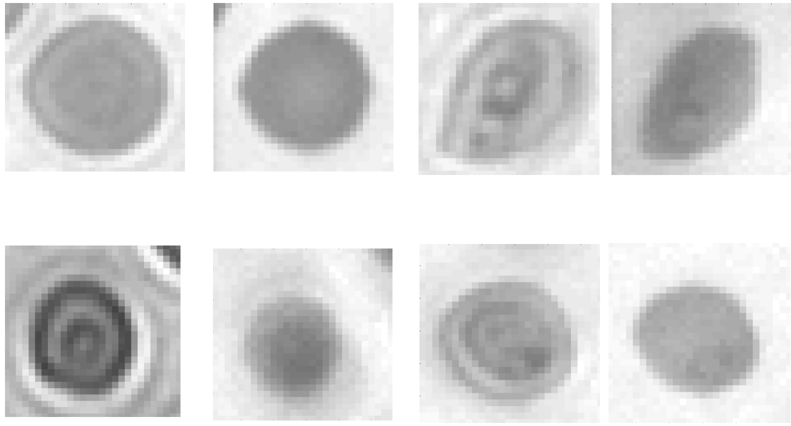
Trained CNN

Real data is big…what to do??



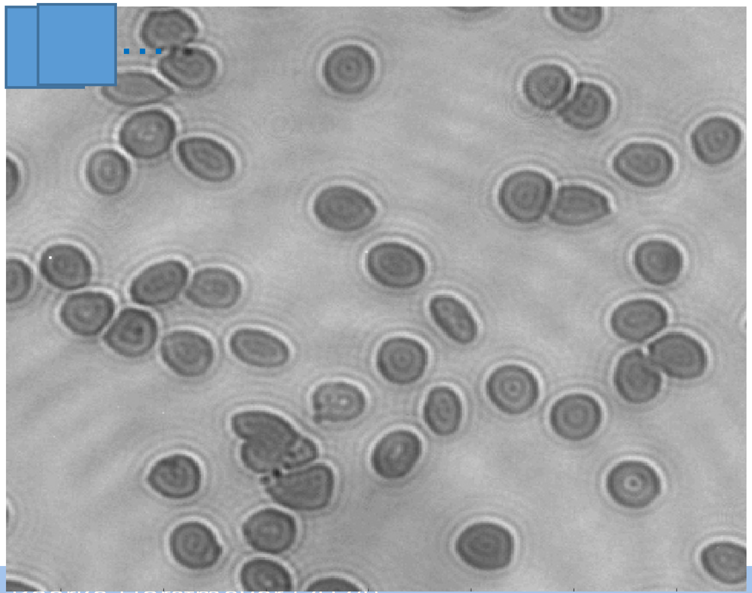
Beyond statistics, how can we visualize performance for classification?

Training dataset



Trained CNN

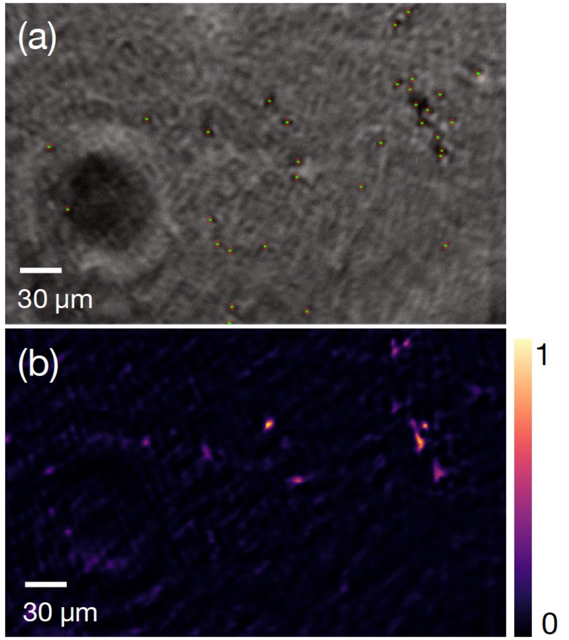
Real data is big…what to do??



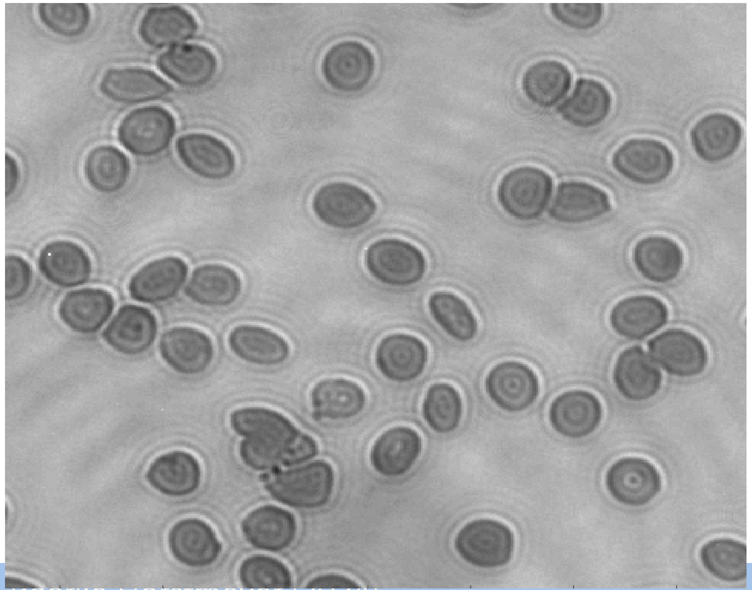
Use sliding window!

Beyond statistics, how can we visualize performance for classification?

Trained CNN

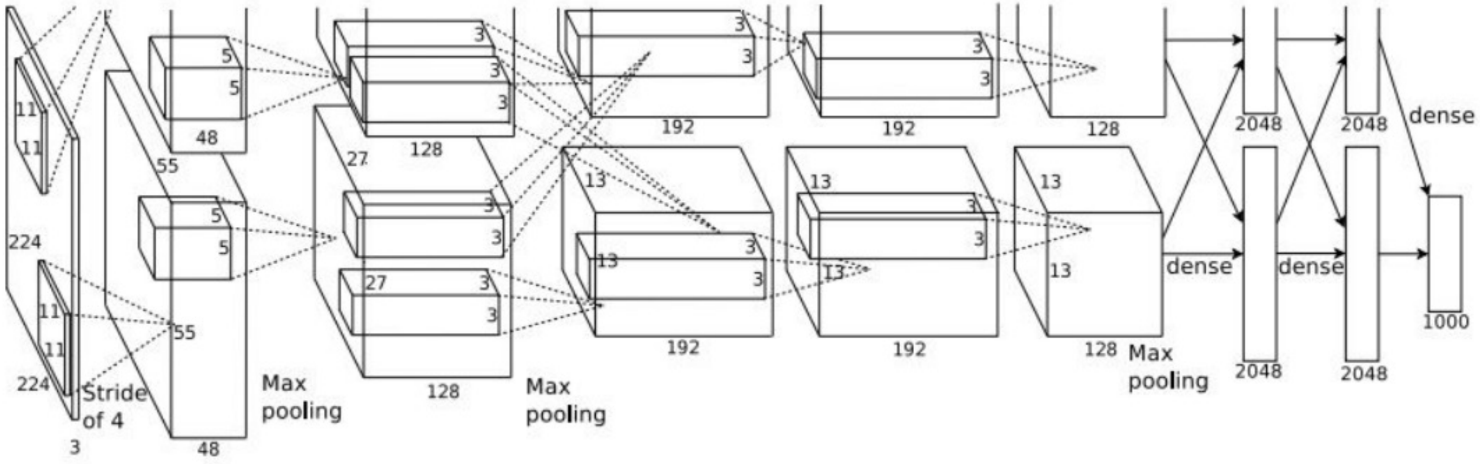


Use sliding window!



How can we visualize what's in the network?

Input Image:
3 x 224 x 224

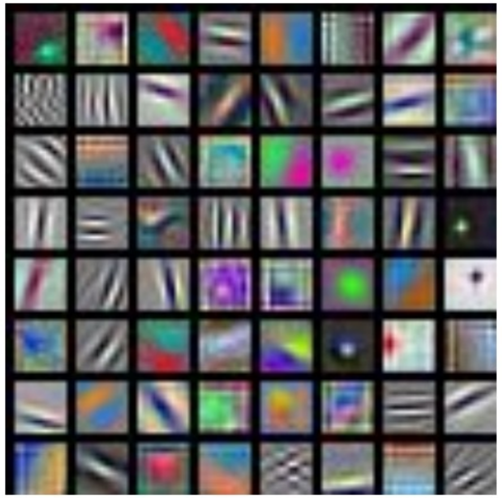


Class Scores:
1000 numbers

What are the intermediate features looking for?

Stanford CS231n: http://cs231n.stanford.edu/

First Layer: Visualize Filters

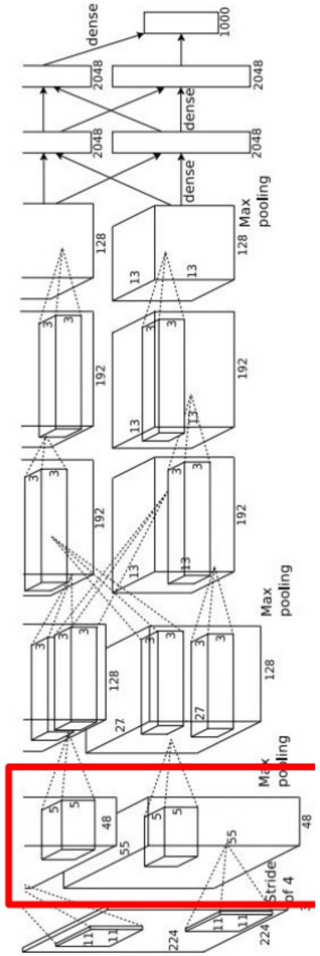


AlexNet:
64 x 3 x 11 x 11

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

Stanford CS231n: http://cs231n.stanford.edu/

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

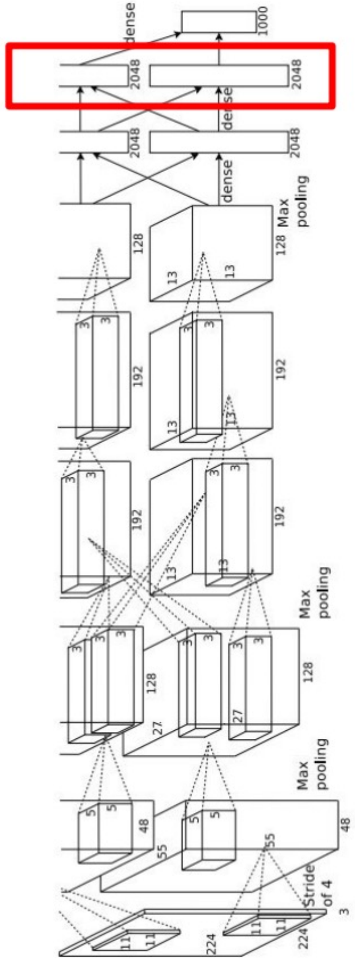
20 x 16 x 7 x 7

layer 3 weights

20 x 20 x 7 x 7

Stanford CS231n: http://cs231n.stanford.edu/

Last Layer



FC7 layer

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

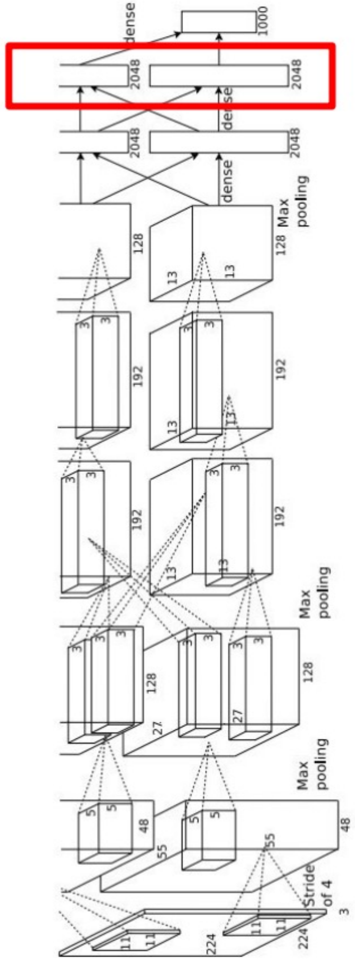
Run the network on many images, collect the feature vectors

Stanford CS231n: http://cs231n.stanford.edu/

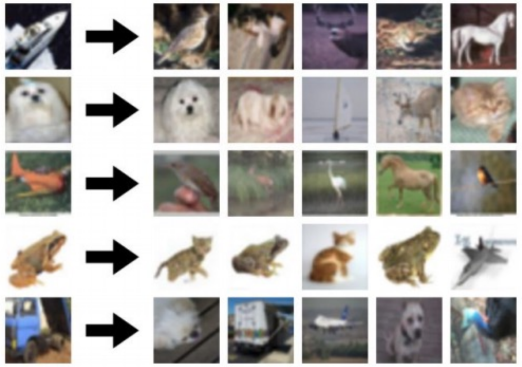
Last Layer

Nearest Neighbors

FC7 layer



Recall: Nearest neighbors in pixel space



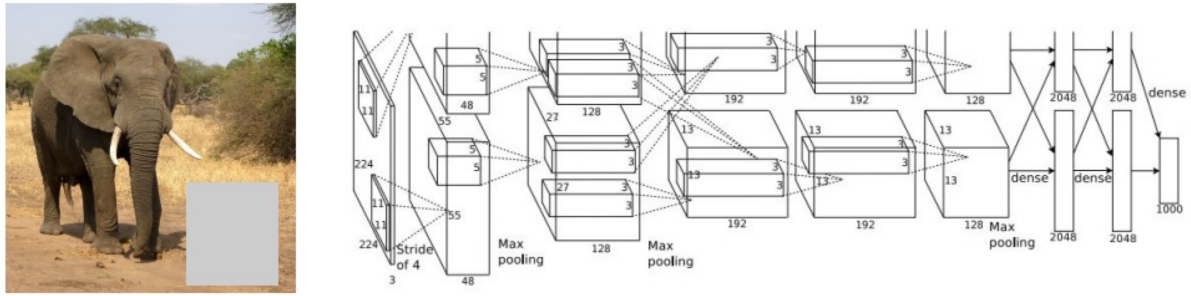
Test image L2 Nearest neighbors in feature space

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

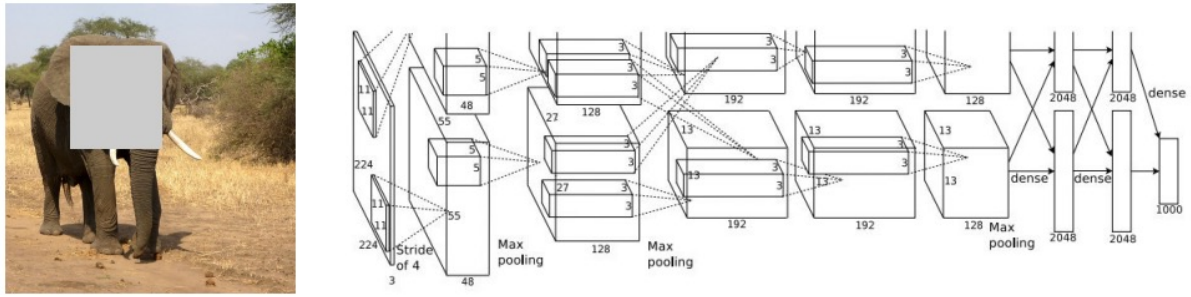
Stanford CS231n: http://cs231n.stanford.edu/

Which pixels matter: Saliency vs Occlusion

Mask part of the image before feeding to CNN, check how much predicted probabilities change



P(elephant) = 0.95



P(elephant) = 0.75

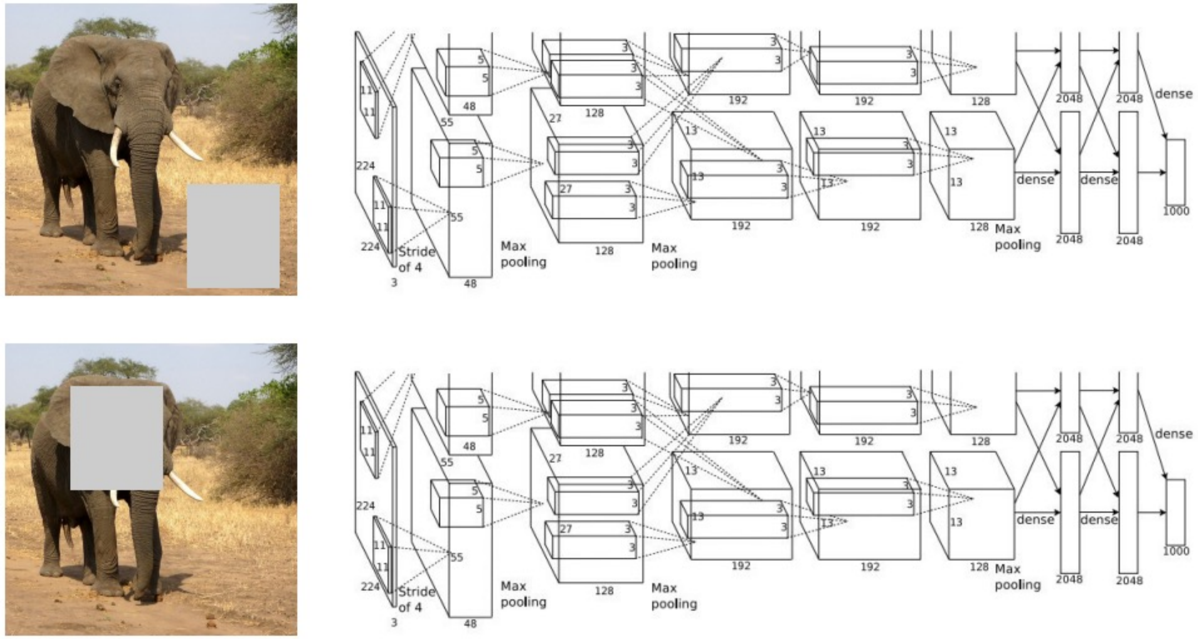
Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

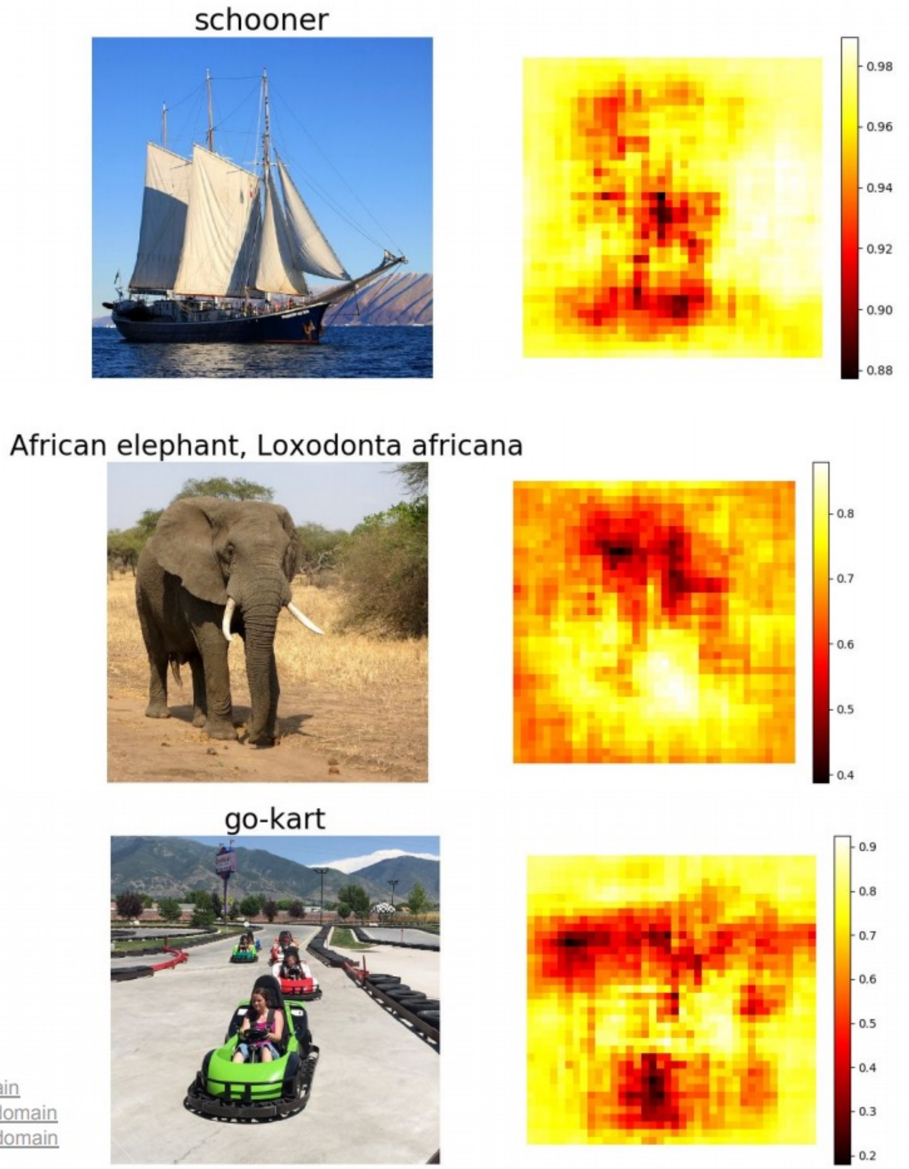
Boat image is CC0 public domain
Elephant image is CC0 public domain
Go-Karts image is CC0 public domain

Stanford CS231n: http://cs231n.stanford.edu/

Which pixels matter: Saliency vs Occlusion

Mask part of the image before feeding to CNN, check how much predicted probabilities change

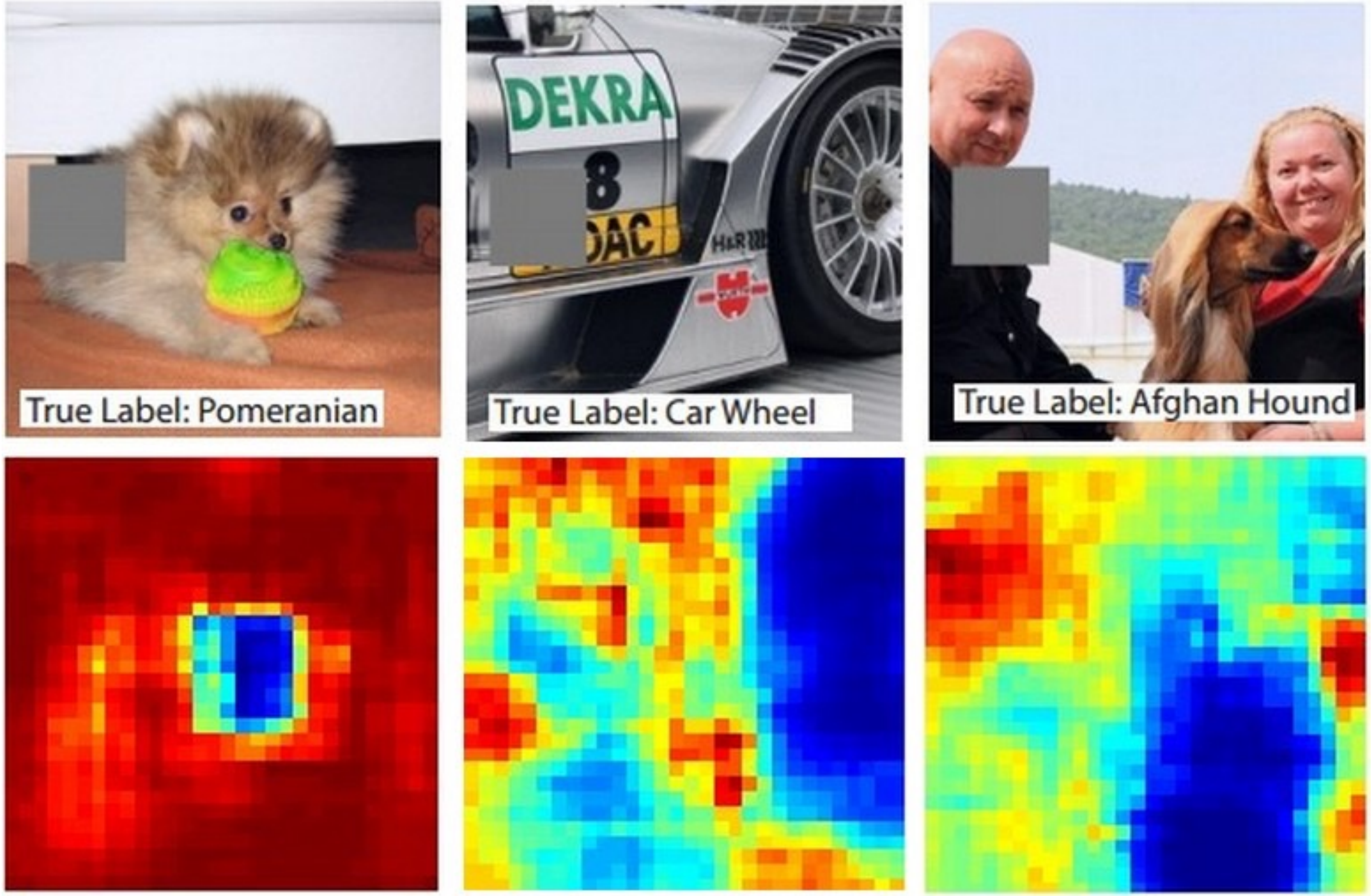




Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Boat image is CC0 public domain
Elephant image is CC0 public domain
Go-Karts image is CC0 public domain

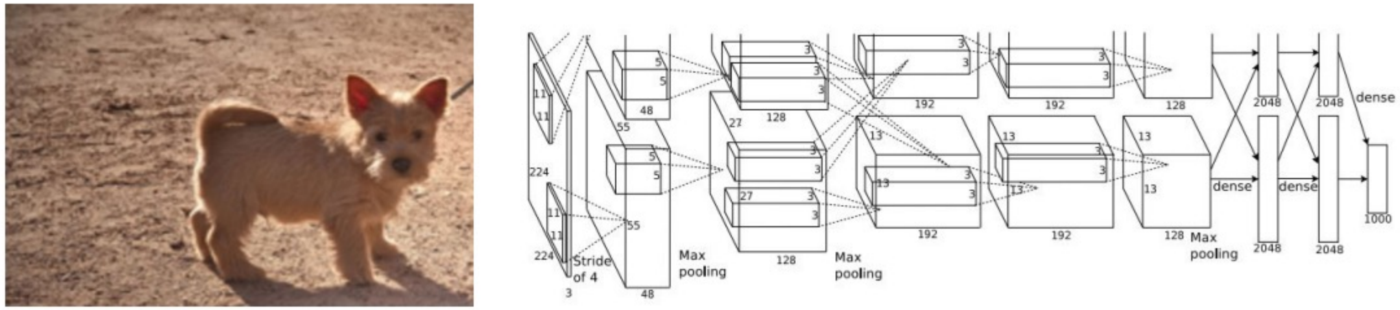
Stanford CS231n: http://cs231n.stanford.edu/



Stanford CS231n: http://cs231n.stanford.edu/

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



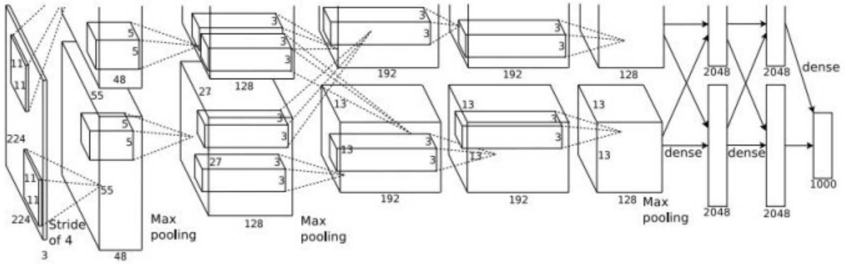
Dog

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Stanford CS231n: http://cs231n.stanford.edu/

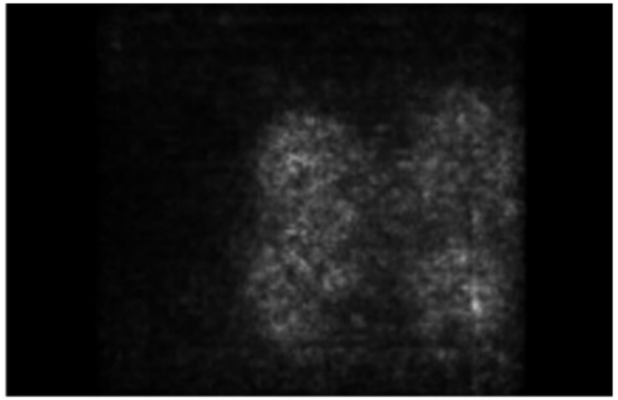
Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



Dog

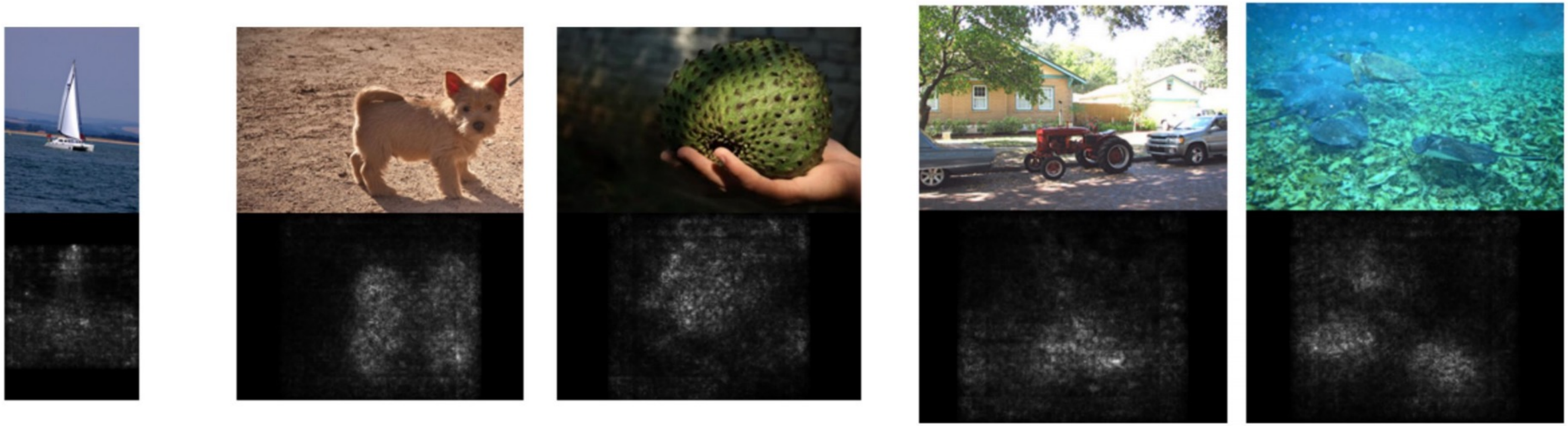
Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Stanford CS231n: http://cs231n.stanford.edu/

Saliency Maps

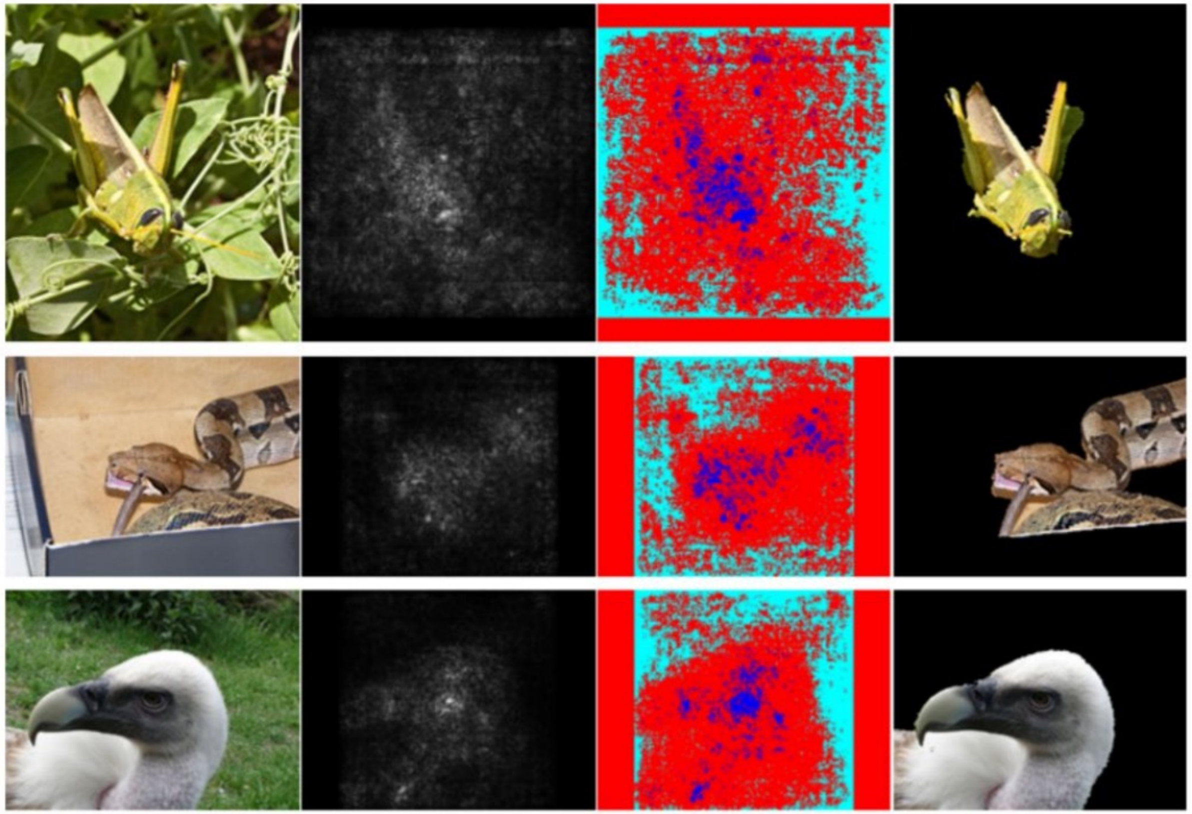


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Stanford CS231n: http://cs231n.stanford.edu/

Saliency Maps: Segmentation without supervision

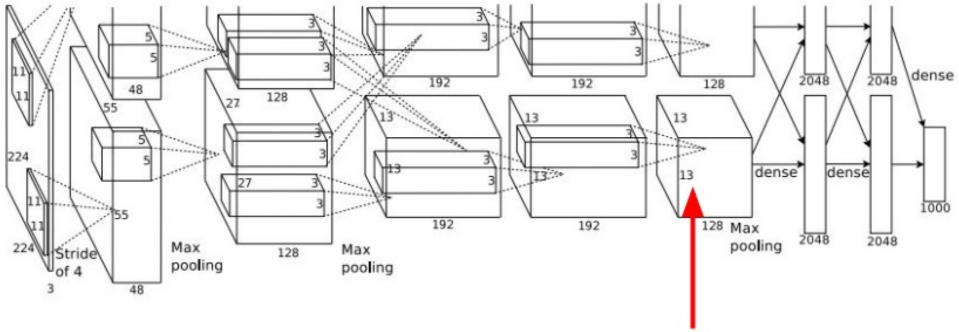
Use GrabCut on saliency map



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.
Rother et al. "Grabcut: Interactive foreground extraction using iterated graph cuts", ACM TOG 2004

Stanford CS231n: http://cs231n.stanford.edu/

Intermediate Features via (guided) backprop



Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

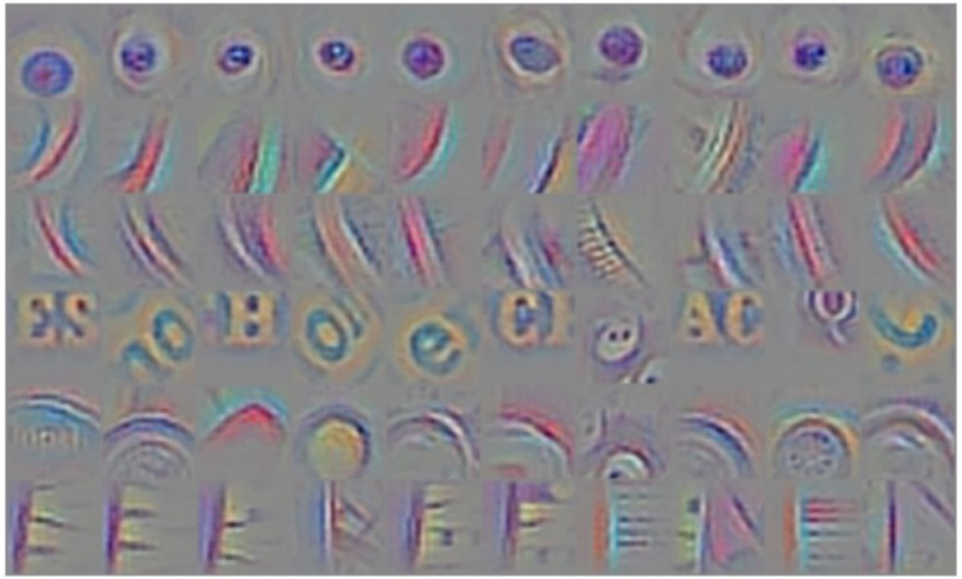
Compute gradient of neuron value with respect to image pixels

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

Stanford CS231n: http://cs231n.stanford.edu/

Intermediate features via (guided) backprop

Maximally activating patches
(Each row is a different neuron)



Guided Backprop

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

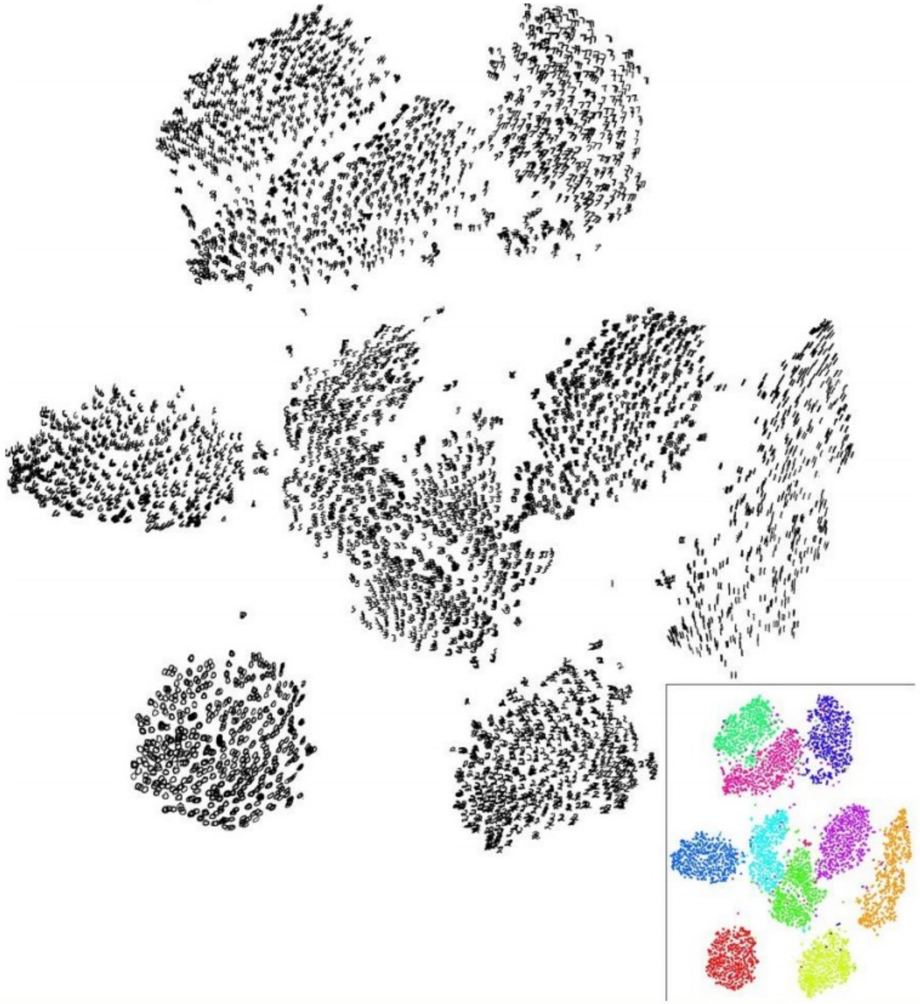
Stanford CS231n: http://cs231n.stanford.edu/

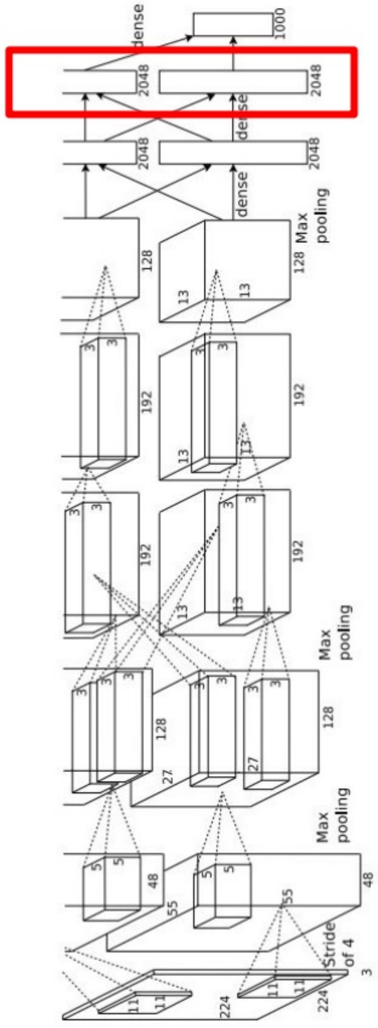
Last Layer: Dimensionality Reduction

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

Simple algorithms:
- K-means clustering
- Principle Component Analysis (PCA)

- More Complex: **t-SNE**



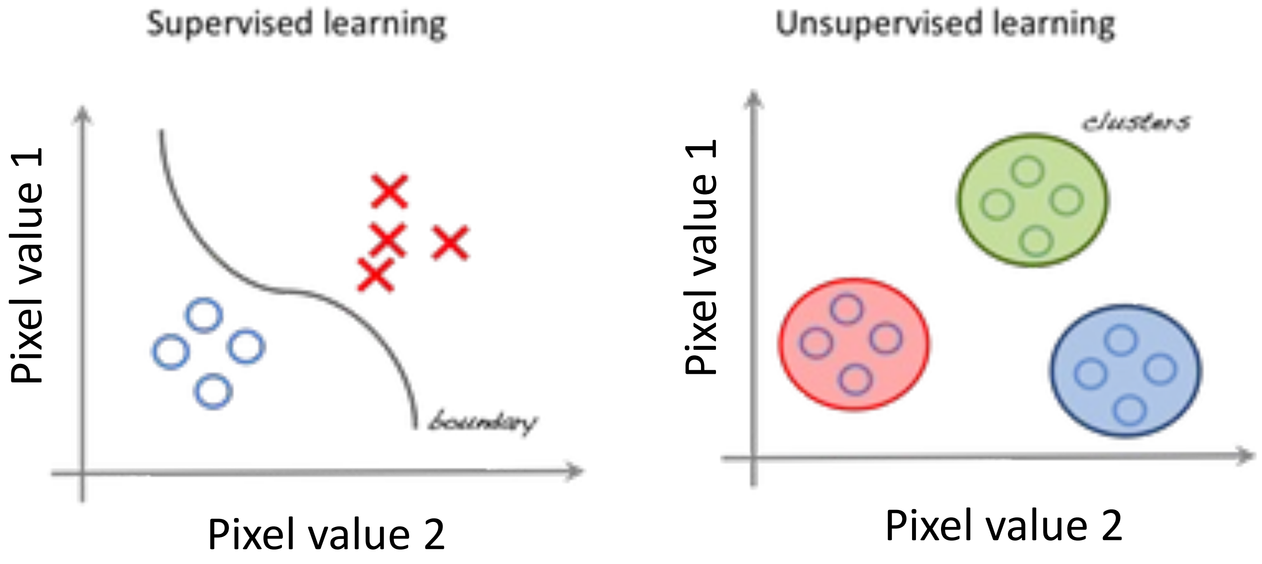


Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008
Figure copyright Laurens van der Maaten and Geoff Hinton, 2008. Reproduced with permission.

Stanford CS231n: http://cs231n.stanford.edu/

Dimensionality reduction is connected to “unsupervised learning”

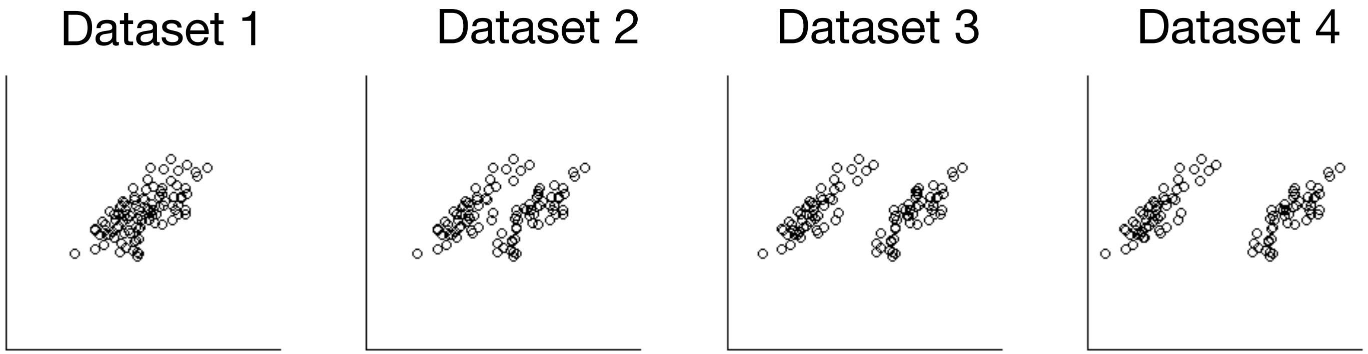
Definition of Unsupervised Learning:
Learning useful structure *without* labeled classes, optimization criterion, feedback signal, or any other information beyond the raw data



Unsupervised learning in a nutshull

Mathematical tools for finding patterns in data:

- Eigenvector decomposition
- Principal component analysis
- Singular value decomposition

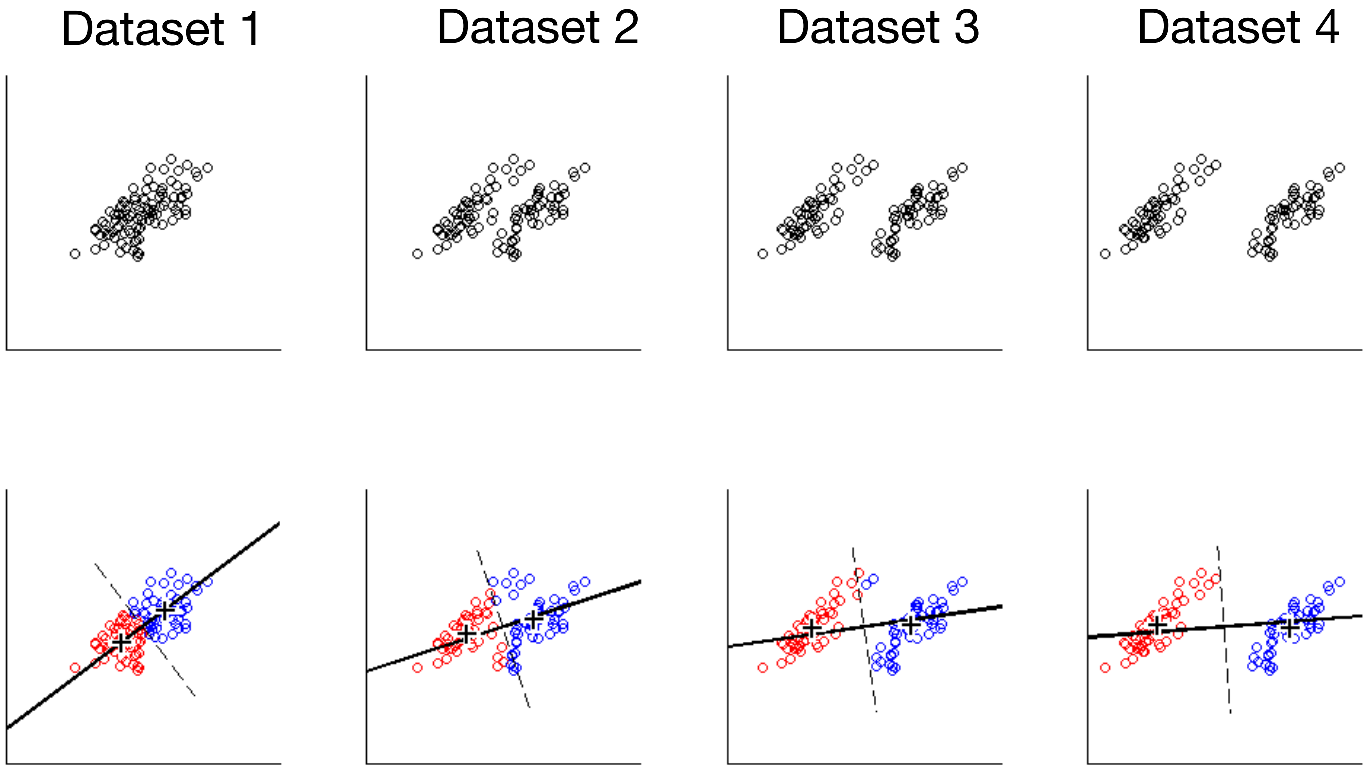


https://stats.stackexchange.com/questions/183236/what-is-the-relation-between-k-means-clustering-and-pca

Unsupervised learning in a nutshull

Mathematical tools for finding patterns in data:

- Eigenvector decomposition
- Principal component analysis
- Singular value decomposition

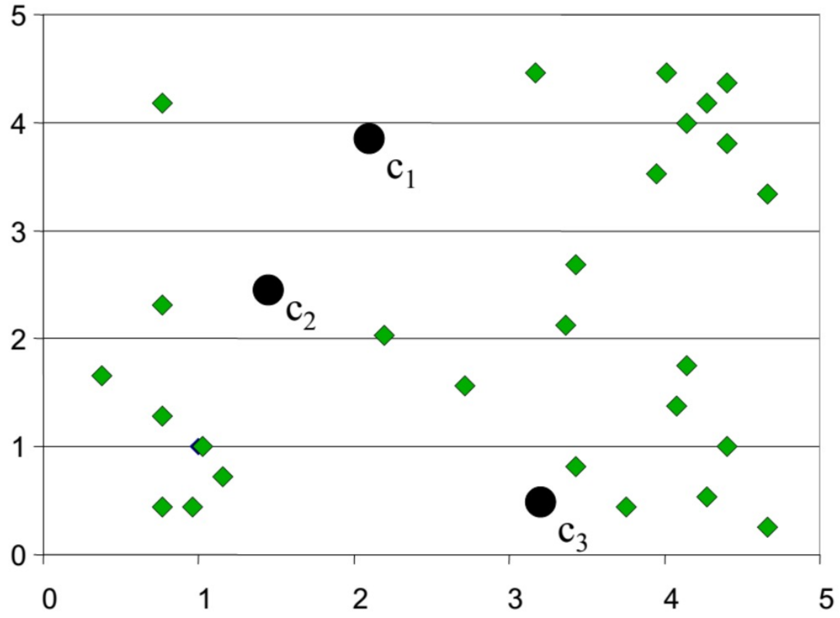


https://stats.stackexchange.com/questions/183236/what-is-the-relation-between-k-means-clustering-and-pca

K-Means Clustering

- Given *k*, the *k-means* algorithm works as follows:
  1. Choose *k* (random) data points (seeds) to be the initial centroids, cluster centers
  2. Assign each data point to the closest centroid
  3. Re-compute the centroids using the current cluster memberships
  4. If a convergence criterion is not met, repeat steps 2 and 3

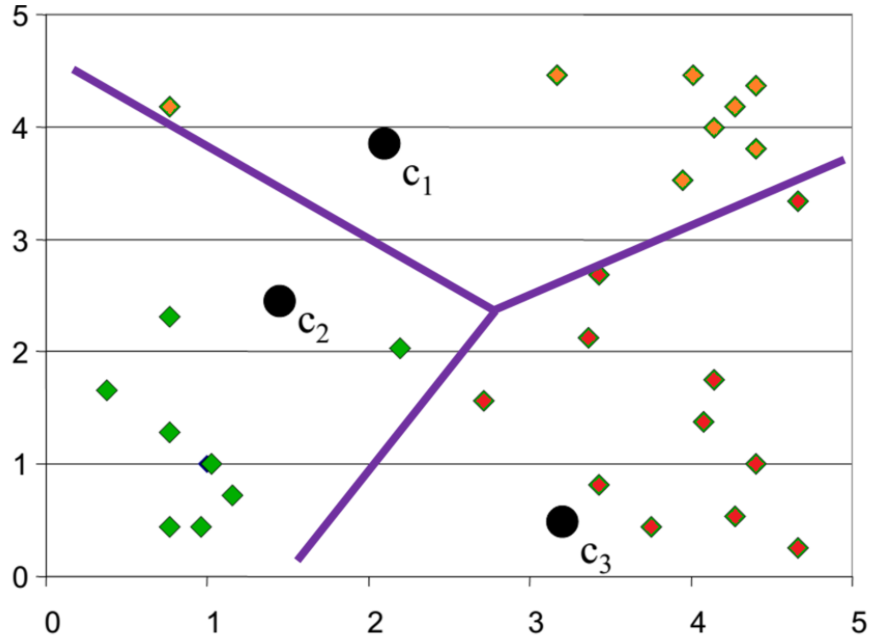
Randomly initialize seeds



K-Means Clustering

- Given *k*, the *k-means* algorithm works as follows:
  1. Choose *k* (random) data points (seeds) to be the initial centroids, cluster centers
  2. Assign each data point to the closest centroid
  3. Re-compute the centroids using the current cluster memberships
  4. If a convergence criterion is not met, repeat steps 2 and 3

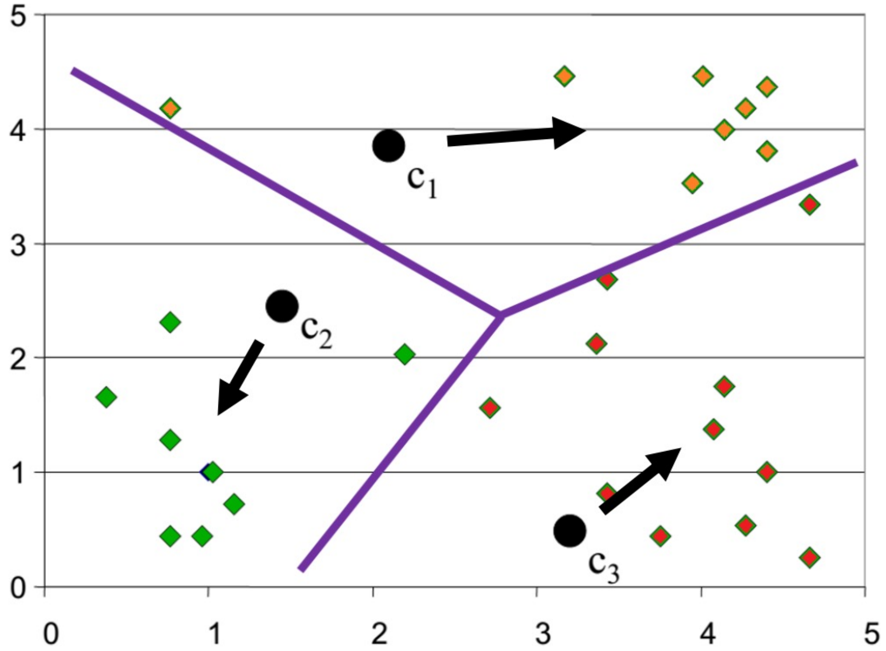
Determine cluster membership for each data point



K-Means Clustering

- Given *k*, the *k-means* algorithm works as follows:
  1. Choose *k* (random) data points (seeds) to be the initial centroids, cluster centers
  2. Assign each data point to the closest centroid
  3. Re-compute the centroids using the current cluster memberships
  4. If a convergence criterion is not met, repeat steps 2 and 3

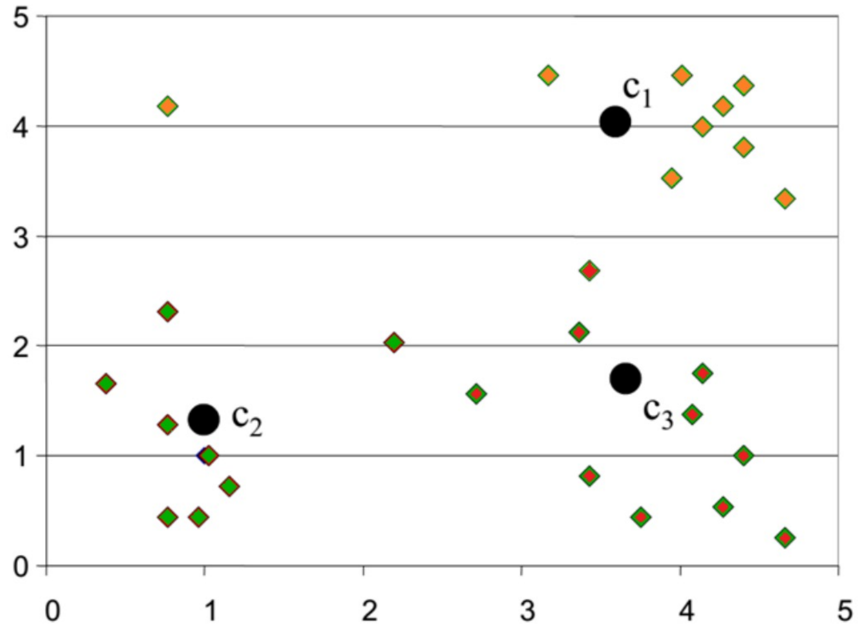
Compute and update new cluster center



K-Means Clustering

- Given *k*, the *k-means* algorithm works as follows:
  1. Choose *k* (random) data points (seeds) to be the initial centroids, cluster centers
  2. Assign each data point to the closest centroid
  3. Re-compute the centroids using the current cluster memberships
  4. If a convergence criterion is not met, repeat steps 2 and 3

Result of first iteration



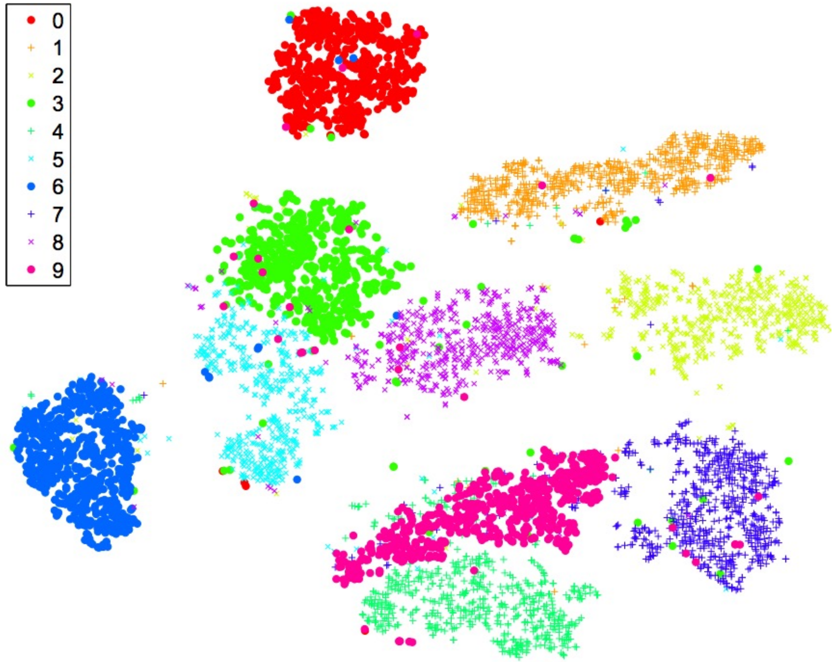
TSNE for data visualization

- Reduce data dimensions to enable visualization in 2D or 3D
  - nD -> 2D or 3D
  - Preserve local structure of data to highlight groups
  - Unsupervised – clusters unlabeled data

TSNE for data visualization

- Reduce data dimensions to enable visualization in 2D or 3D
  - nD -> 2D or 3D
  - Preserve local structure of data to highlight groups
  - Unsupervised – clusters unlabeled data

Applied to MNIST digits

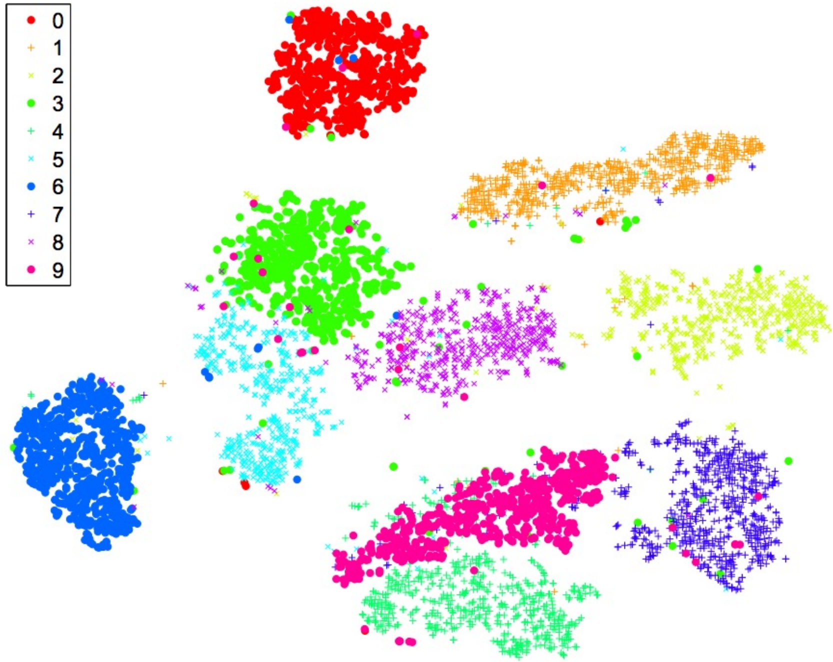


Stanford CS231n: http://cs231n.stanford.edu/

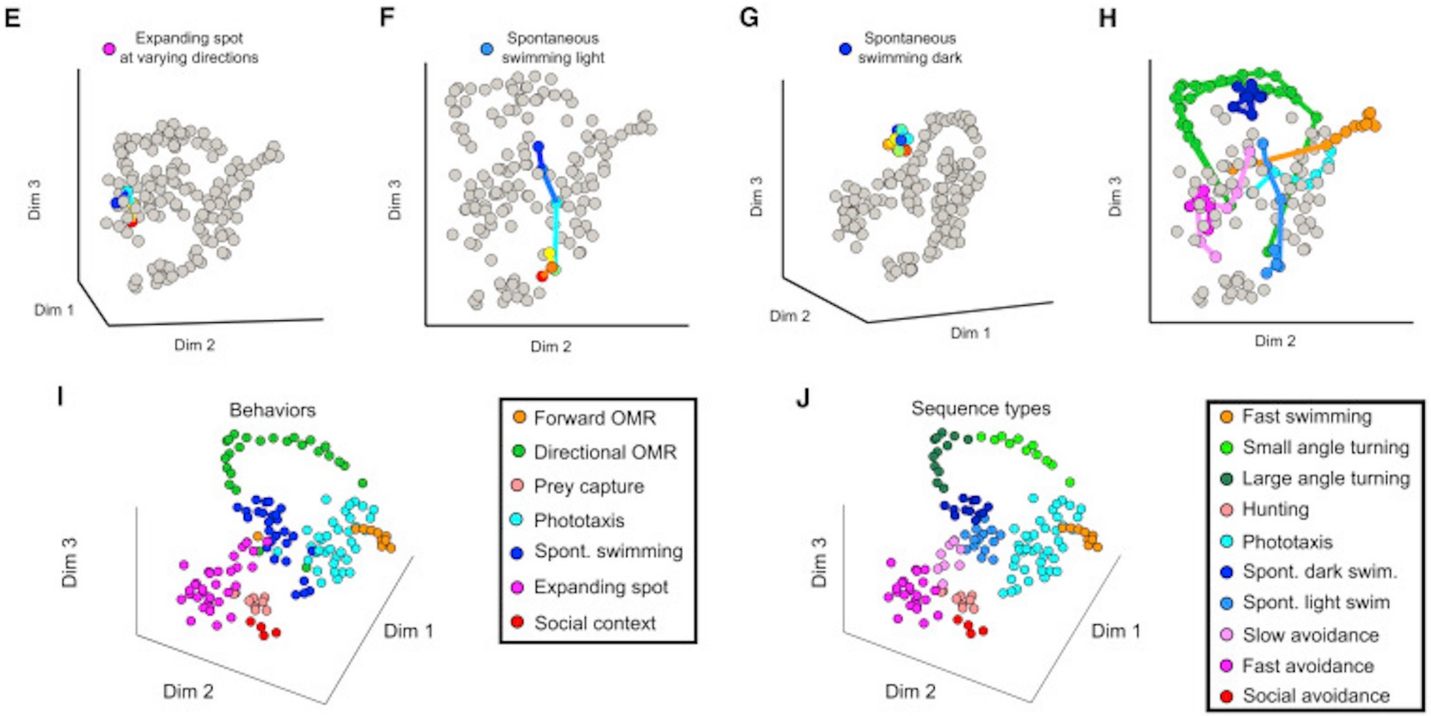
TSNE for data visualization

- Reduce data dimensions to enable visualization in 2D or 3D
  - nD -> 2D or 3D
  - Preserve local structure of data to highlight groups
  - Unsupervised – clusters unlabeled data

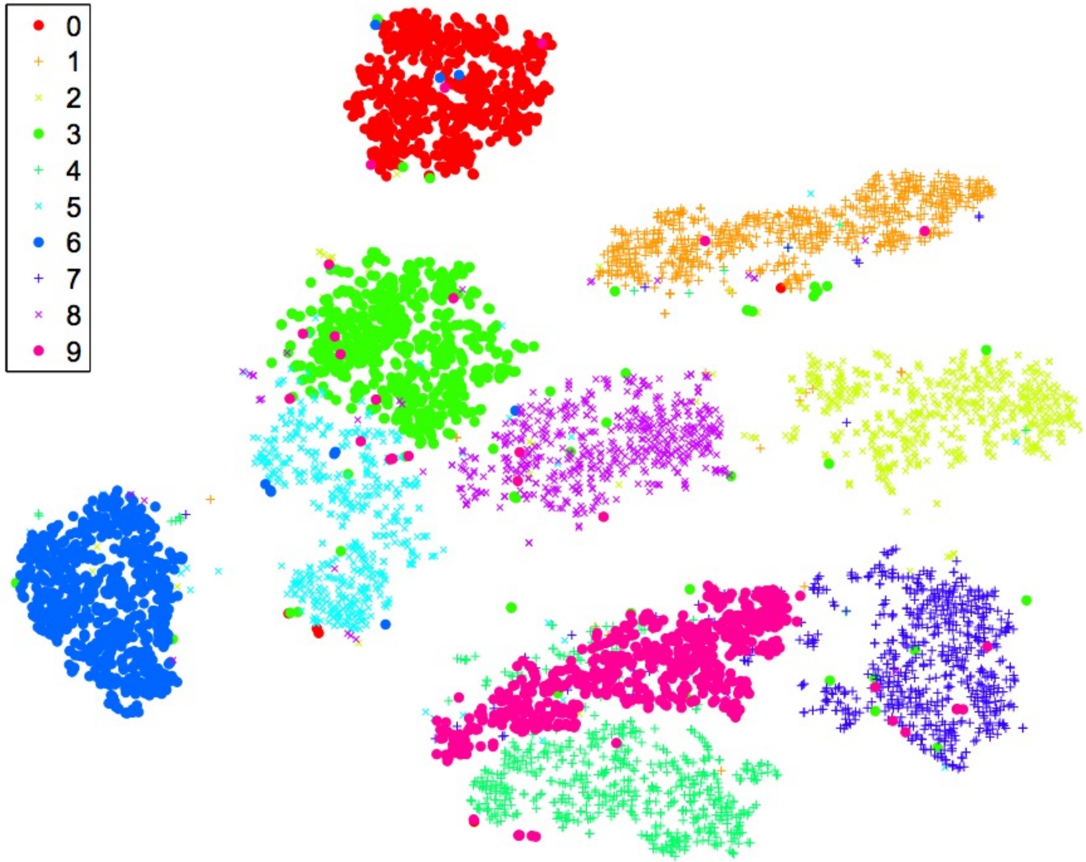
Applied to MNIST digits



Applied to movies of zebrafish behavior



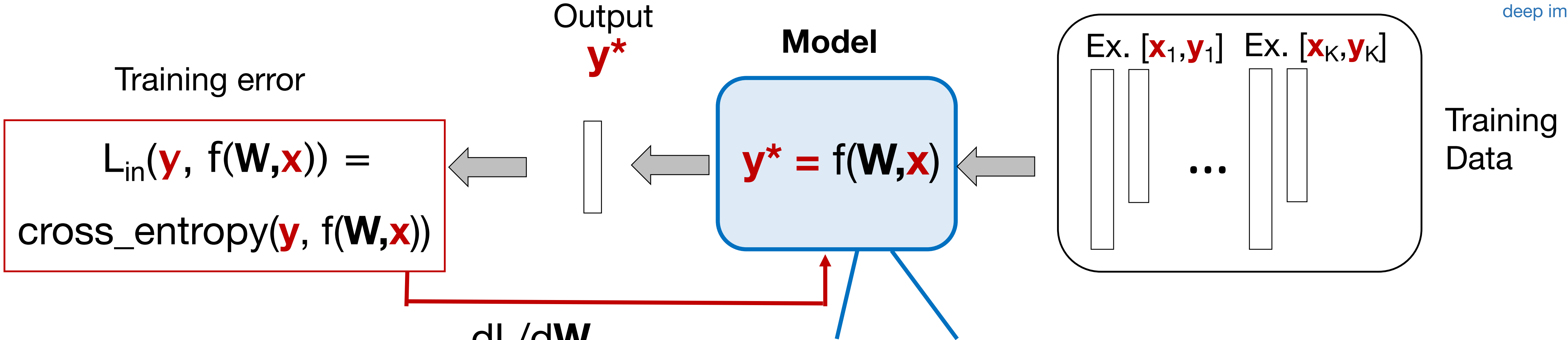
Aside about clustering data – why do we need deep learning at all?



Isn't this good enough?

Stanford CS231n: http://cs231n.stanford.edu/

Next step: let's consider other automated tasks besides image classification!



Output $\mathbf{y}^*$

Model

$\mathbf{y}^* = f(\mathbf{W},\mathbf{x})$

Training error

$L_{in}(\mathbf{y}, f(\mathbf{W},\mathbf{x})) =$

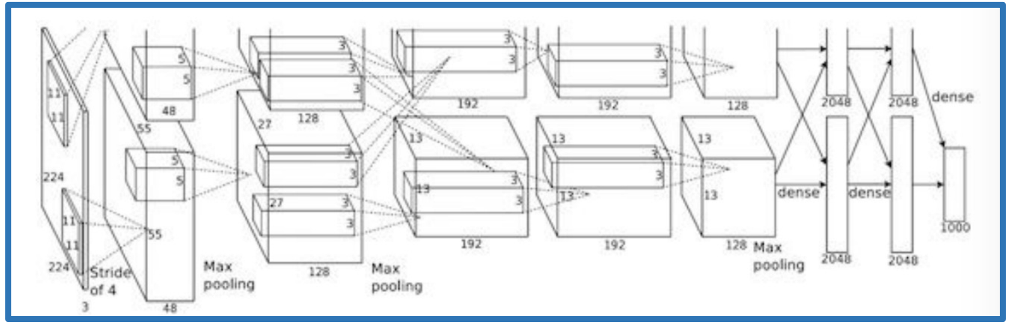
$\text{cross\_entropy}(\mathbf{y}, f(\mathbf{W},\mathbf{x}))$

$dL/d\mathbf{W}$

Ex. $[\mathbf{x}_1,\mathbf{y}_1]$ Ex. $[\mathbf{x}_K,\mathbf{y}_K]$

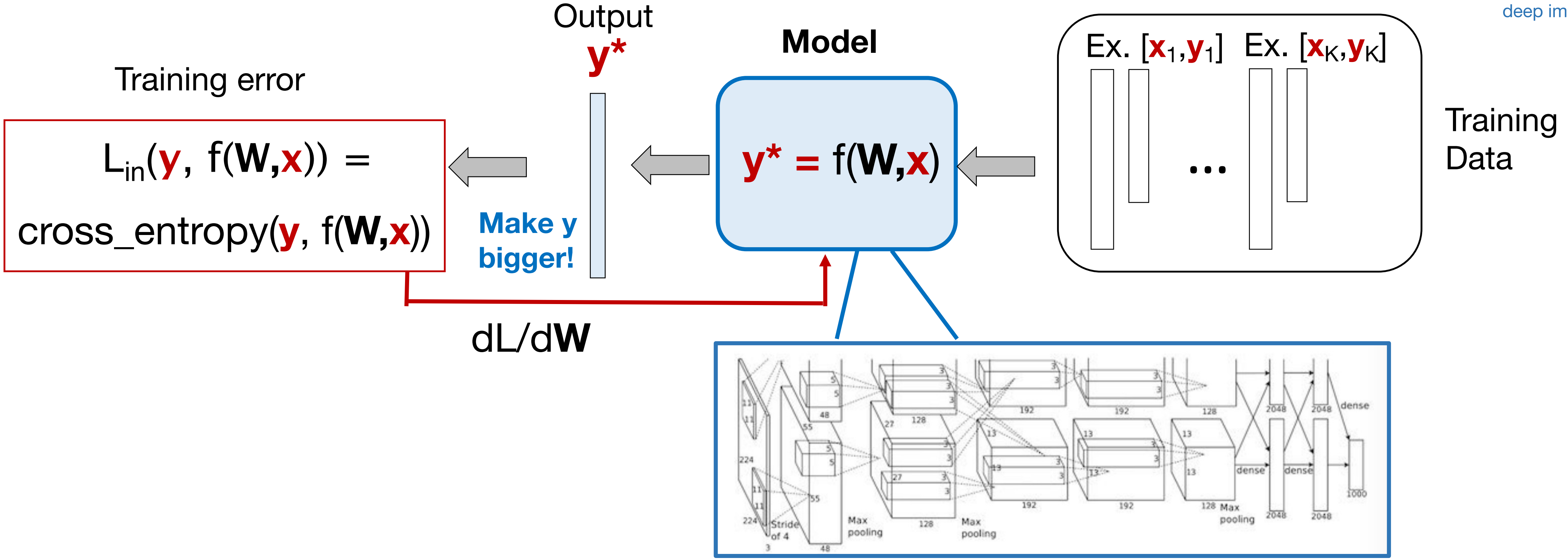
…

Training Data



Dimensional analysis for classification:

Input $\mathbf{x}$: $\sim R^{1000}$
Output $\mathbf{y}^*$: $\sim R^2 - R^{10}$



Output

$\mathbf{y}^*$

Model

Training error

$L_{in}(\mathbf{y}, f(\mathbf{W},\mathbf{x})) =$

$\text{cross\_entropy}(\mathbf{y}, f(\mathbf{W},\mathbf{x}))$

Make y bigger!

$\mathbf{y}^* = f(\mathbf{W},\mathbf{x})$

Ex. $[\mathbf{x}_1,\mathbf{y}_1]$ Ex. $[\mathbf{x}_K,\mathbf{y}_K]$

…

Training Data

$dL/d\mathbf{W}$

Dimensional analysis for classification:

Input **x**: $\sim R^{1000}$

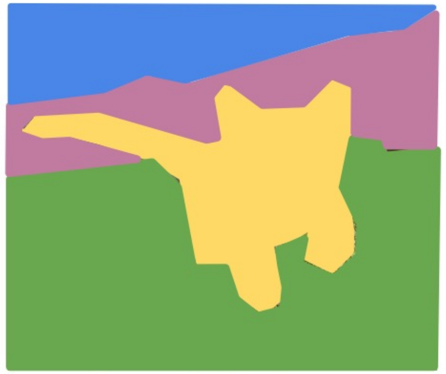
Output **y***: $\sim R^2 - R^{10}$

This class – let's make **y*** bigger!

- Object detection
- Segmentation
- Creating 3D volumes
- Better resolution

Other Computer Vision Tasks

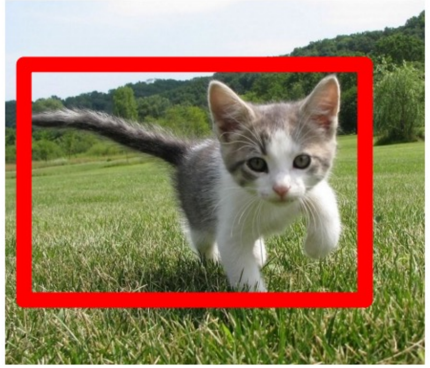
Semantic Segmentation



GRASS, **CAT**, **TREE**, **SKY**

No objects, just pixels

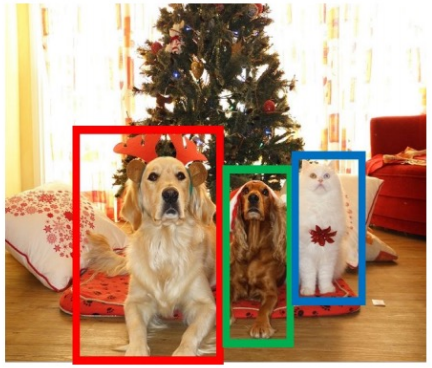
Classification + Localization



CAT

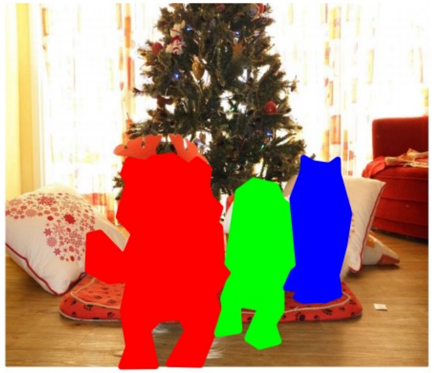
Single Object

Object Detection



DOG, **DOG**, **CAT**

Instance Segmentation



DOG, **DOG**, **CAT**

Multiple Object

This image is CC0 public domain

Super-resolution

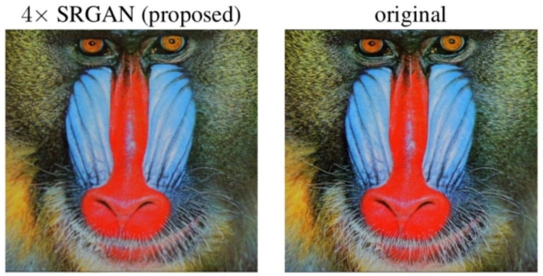


Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

Stanford CS231n - http://cs231n.stanford.edu