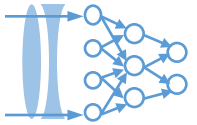
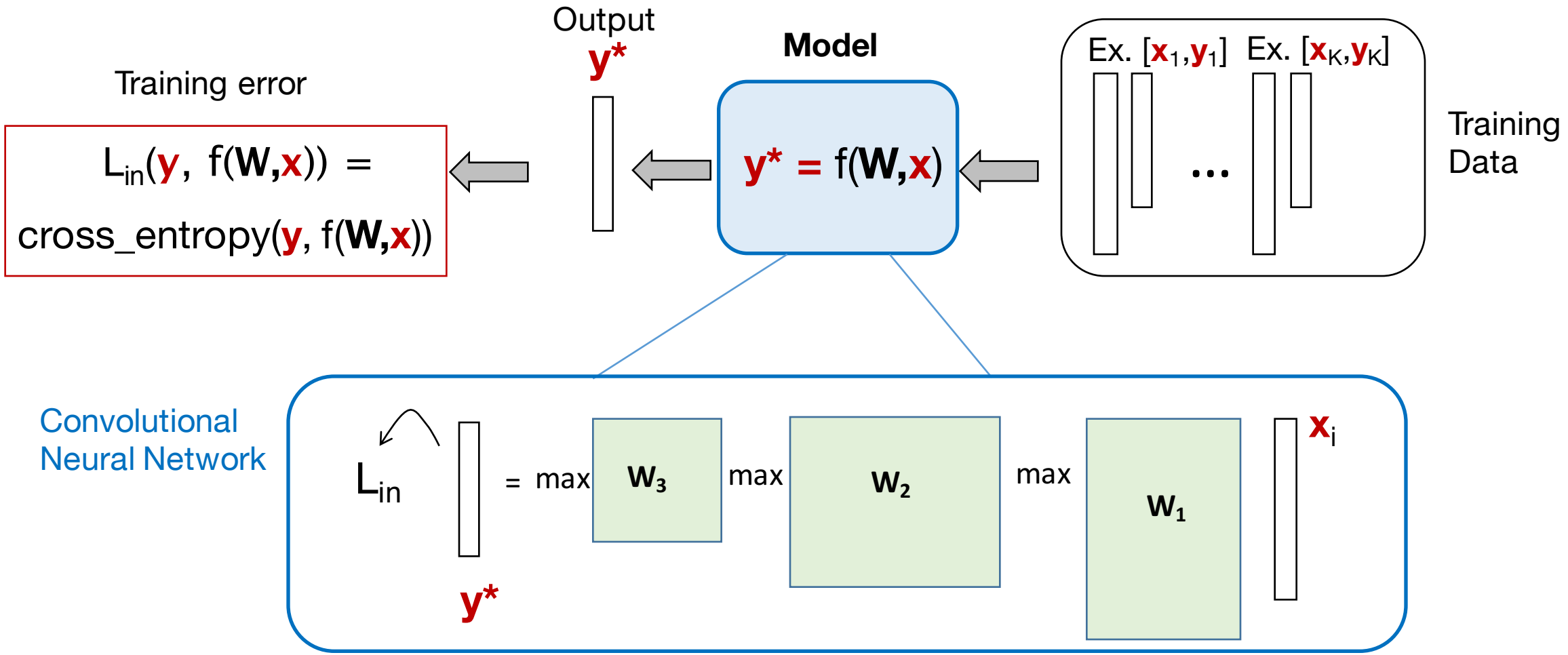


Lecture 13: CNN visualization and example applications

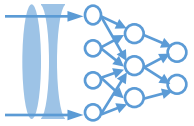
Machine Learning and Imaging

BME 590L
Roarke Horstmeyer

Our very basic convolutional neural network



Forward pass: from $\mathbf{x}_i$ and current $\mathbf{W}$'s, find L_{in}



Important components of a CNN

CNN Architecture

Architecture choices

- CONV size, stride, pad, depth
- ReLU & other nonlinearities
- POOL methods
- # of layers, dimensions per layer
- Fully connected layers

Loss function & optimization

- Type of loss function
- Regularization
- Gradient descent method
- SGD batch and step size

Optimization choices

Other specifics: Initialization, dropout, batch normalization, data normalization & augmentation

You've turned all the knobs and now it works!!!

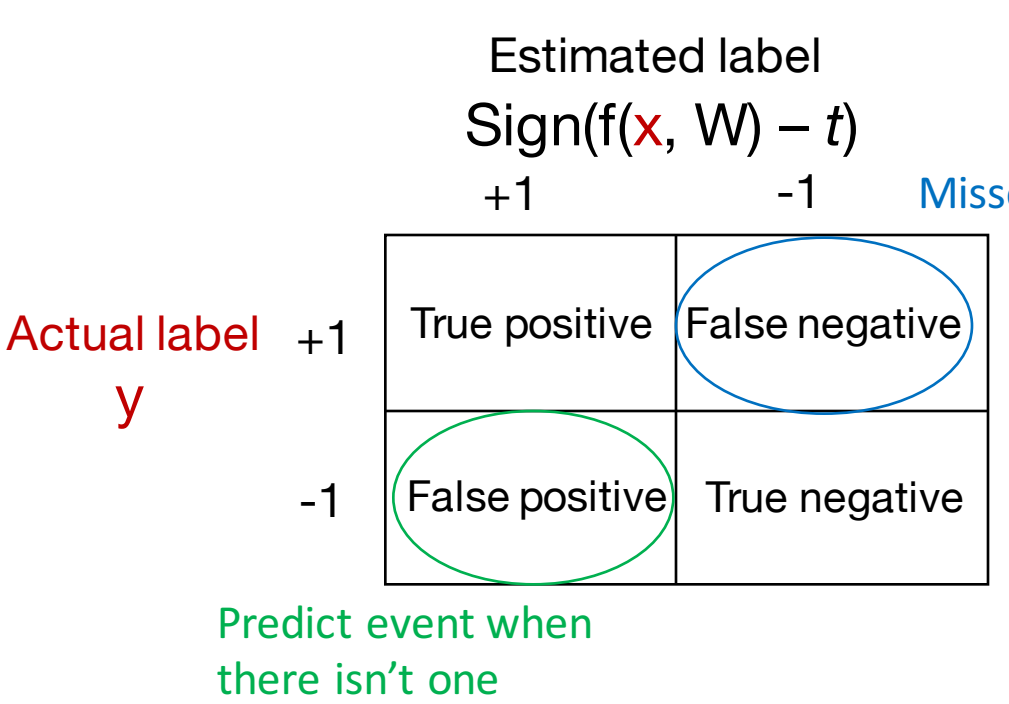
How to examine and present your results: a few options at different stages

Options to examine your test data after processing:

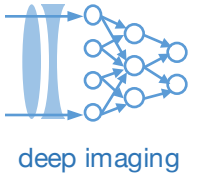
- ROC curve, Precision-Recall
- Confusion matrix
- Sliding window visualization
- Layer visualizations
- Saliency maps etc.
- tSNE visualization

ROC curve and confusion matrix

- Can set threshold *t* for evaluating f(x,W) as wherever you'd like
- Leads to sliding window between FN and FP rate
- Need to summarize both statistics as a function of sliding window



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

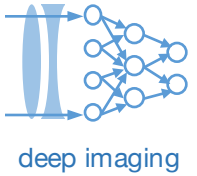
Estimated label
f(x, W)

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

ROC curve and confusion matrix



TP Rate =

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

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

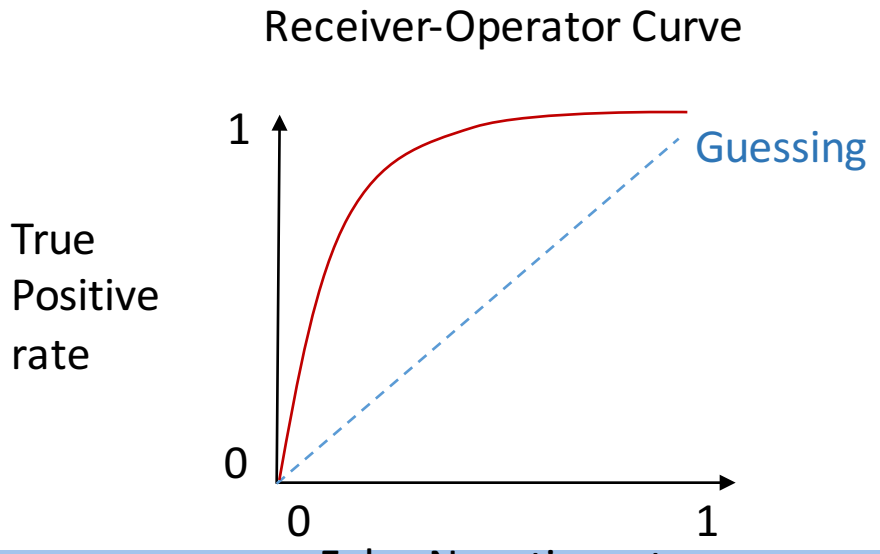
Specificity = TN / (TN + FP) = TN / Actual negatives
= 1 – False Positive Rate

Estimated label
f(x, W)

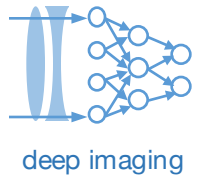
Actual label y	+1	-1
+1	True positive	False negative
-1	False positive	True negative

Missed an event

Predict event when there isn't one



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

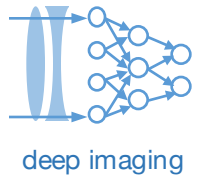
Missed an event

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 – example = neurons in tissue)
- 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)

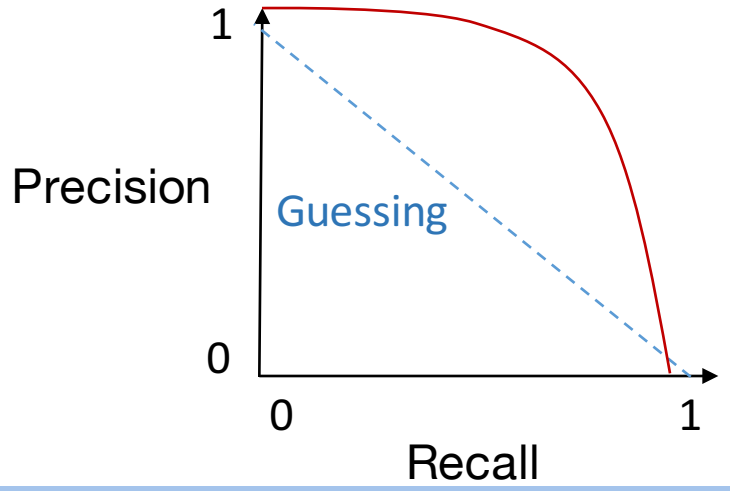
Actual label y	+1	-1
+1	True positive	False negative
-1	False positive	True negative

Missed an event

Predict event when there isn't one

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

Precision-Recall curve



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

ROC curve and confusion matrix

deep imaging

Just 2 categories

Estimated label

f(x, W)

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

Confusion Matrix: 2+ categories

Estimated label

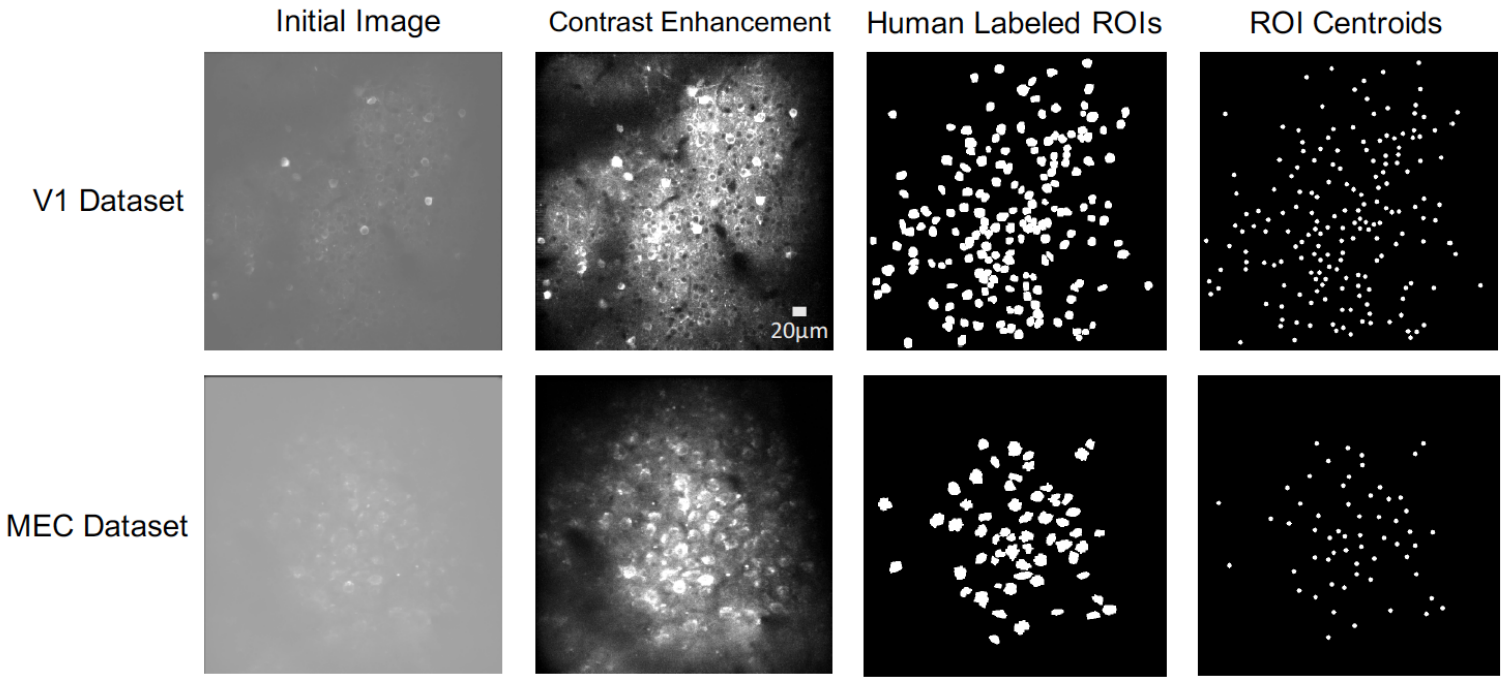
f(x, W)

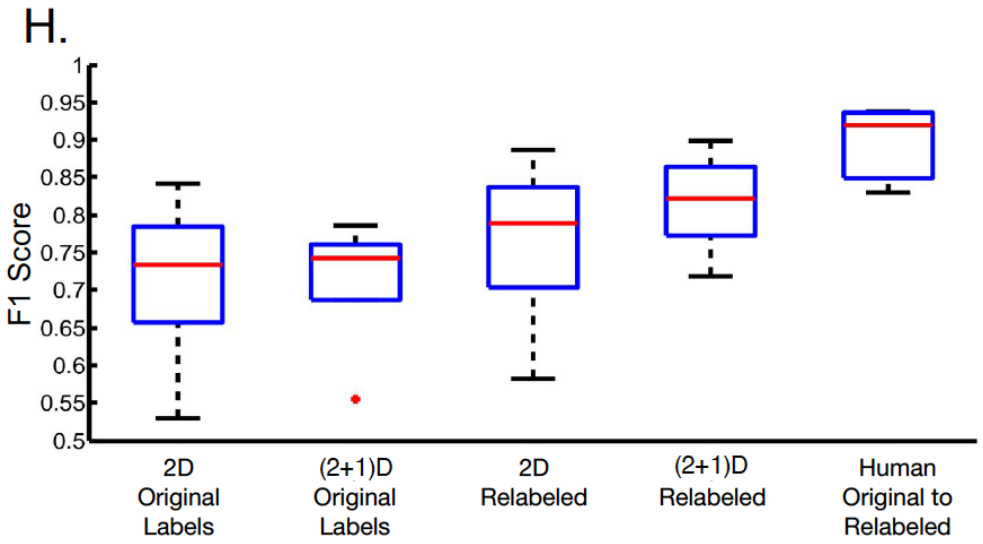
y

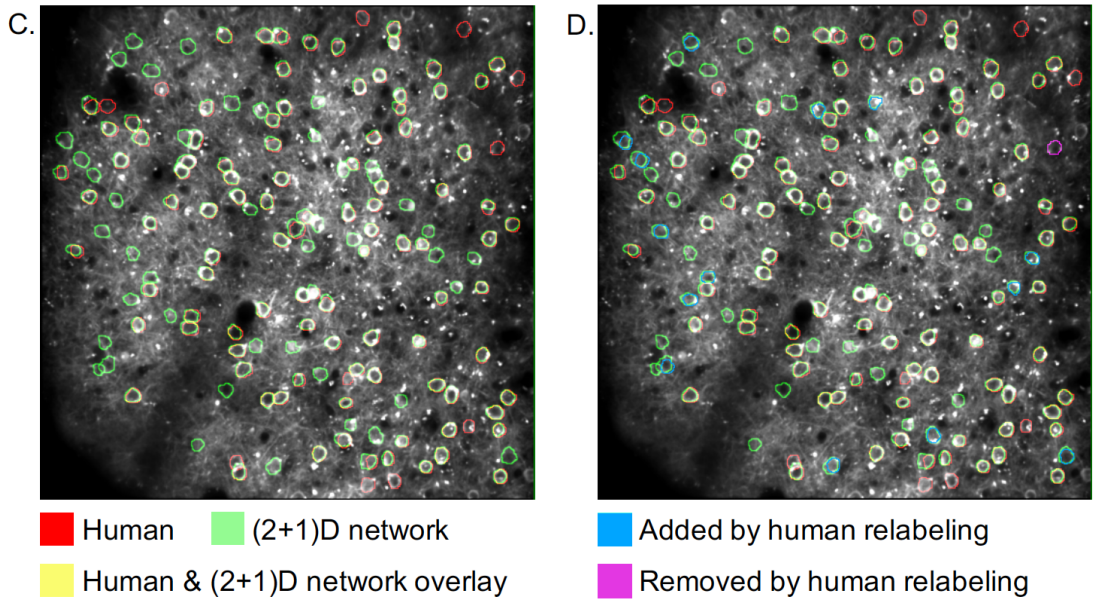
Actual ↓	State1 (Predicted)	State2 (Predicted)	State3 (Predicted)	State4 (Predicted)	State5 (Predicted)	State6 (Predicted)	State7 (Predicted)	State8 (Predicted)
State1 (Actual)	90.12 %	0.00 %	9.88 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
State2 (Actual)	0.00 %	100.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %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

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

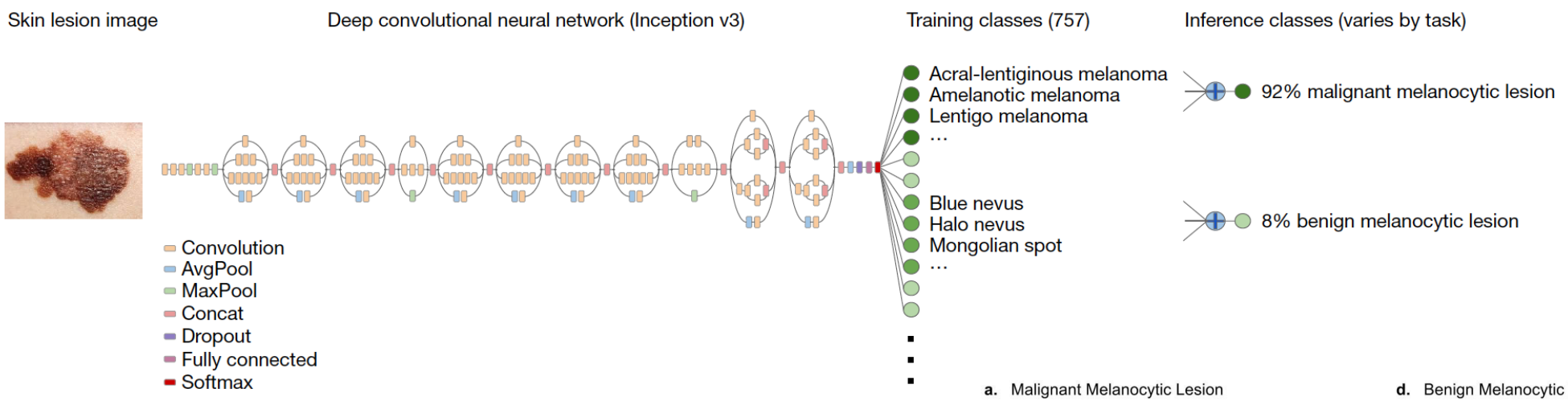


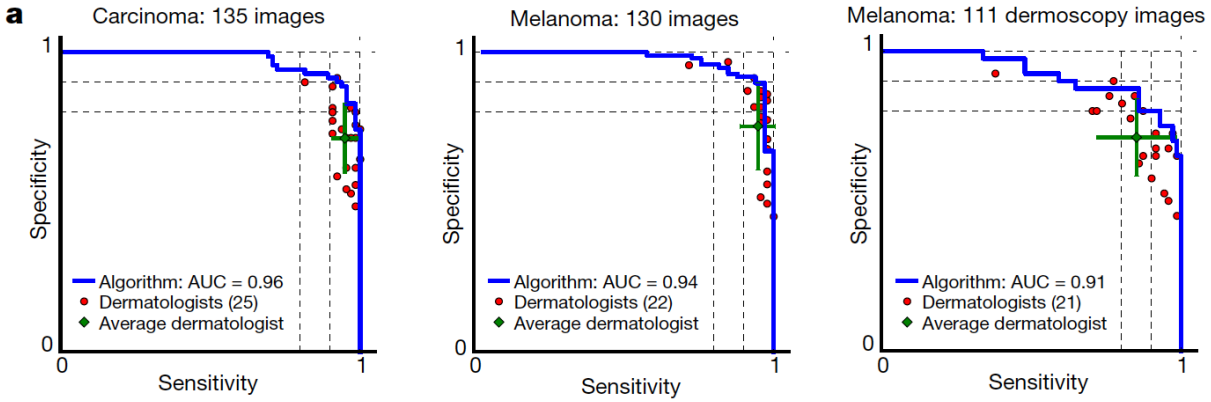


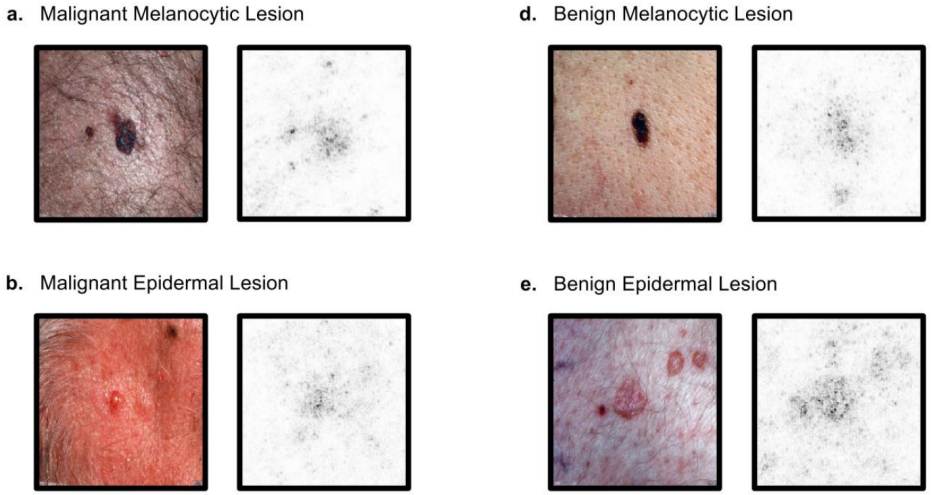


Dermatologist-level classification of skin cancer with deep neural networks

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]

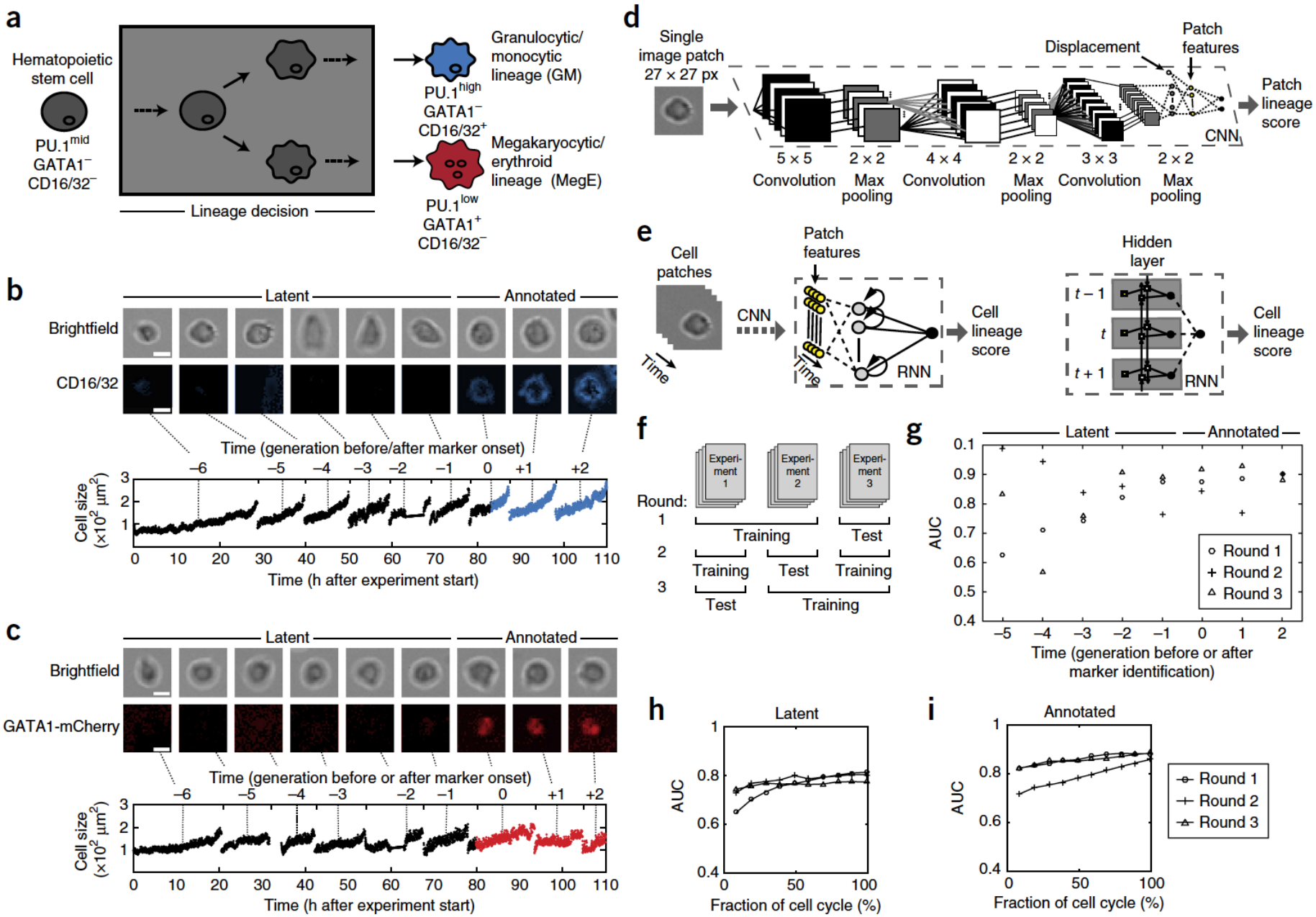




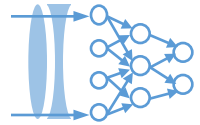


Prospective identification of hematopoietic lineage choice by deep learning

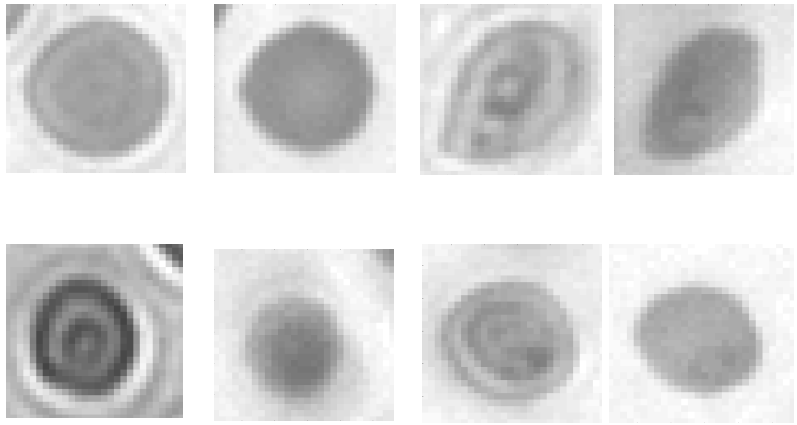
Felix Buggenthin[1,6], Florian Buettner[1,2,6], Philipp S Hoppe[3,4], Max Endele[3], Manuel Kroiss[1,5], Michael Strasser[1], Michael Schwarzfischer[1], Dirk Loeffler[3,4], Konstantinos D Kokkaliaris[3,4], Oliver Hilsenbeck[3,4], Timm Schroeder[3,4], Fabian J Theis[1,5] & Carsten Marr[1]



Beyond statistics, how can we visualize performance for classification?

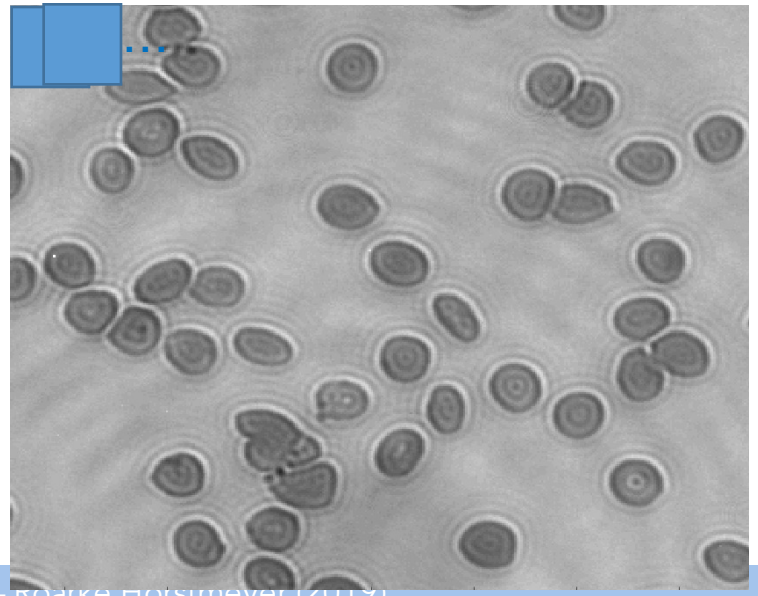


Training dataset



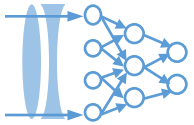
Trained CNN

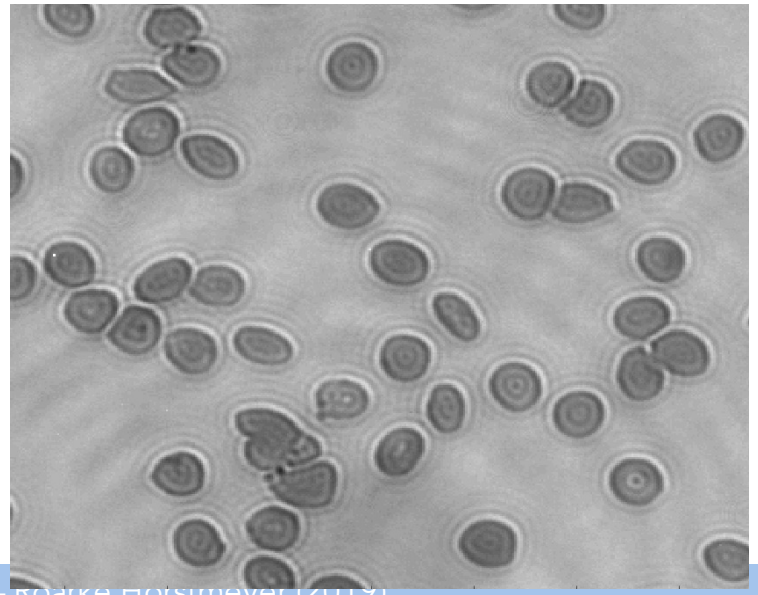
Real data is big…what to do??



Use sliding window!

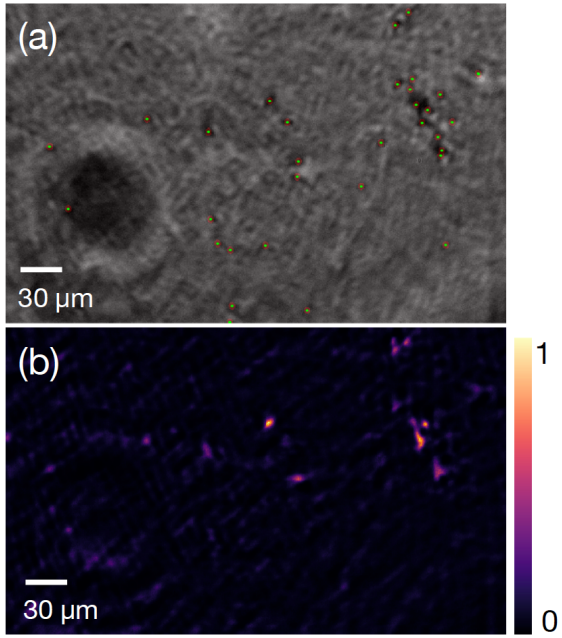
Beyond statistics, how can we visualize performance for classification?



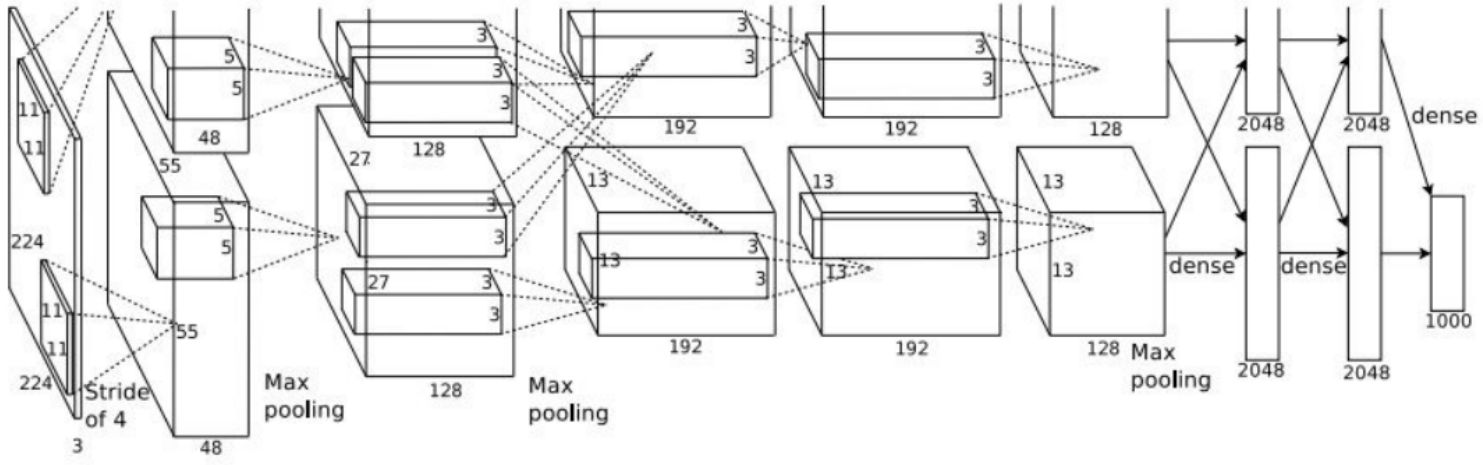


Use sliding window

Trained CNN



How can we visualize what's in the network?



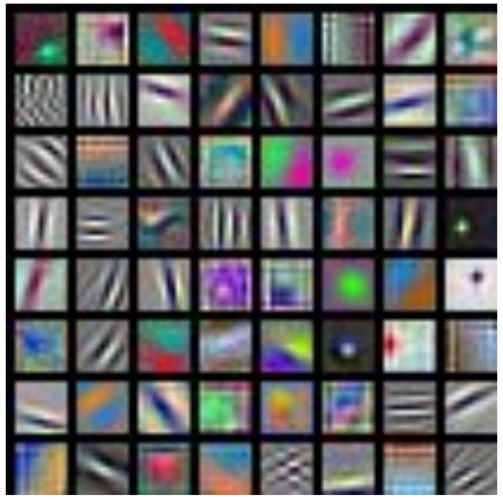
Input Image:
3 x 224 x 224

Class Scores:
1000 numbers

What are the intermediate features looking for?

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

First Layer: Visualize Filters

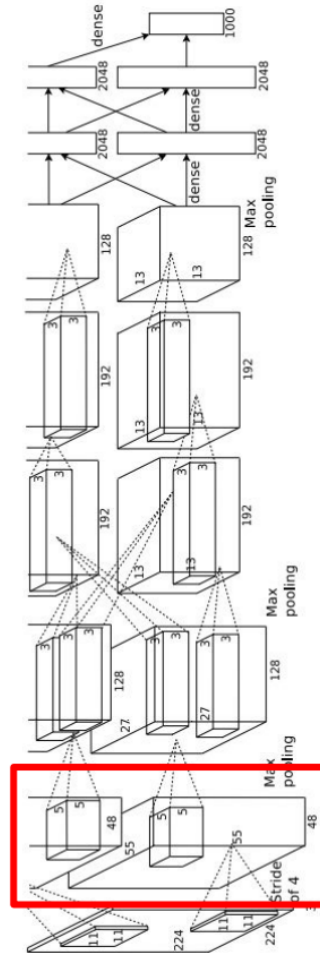


AlexNet:
64 x 3 x 11 x 11

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

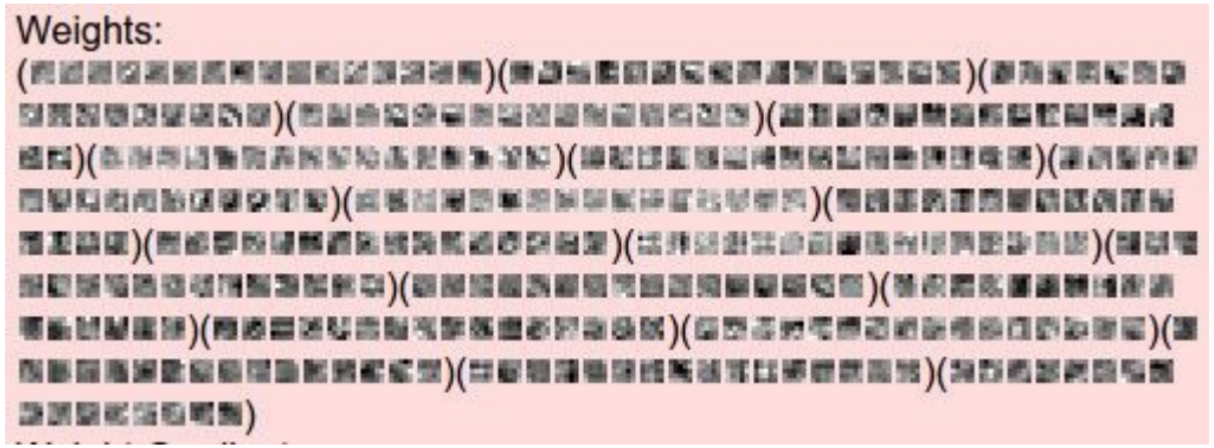
(these are taken from ConvNetJS CIFAR-10 demo)

Weights:

layer 1 weights

16 x 3 x 7 x 7

Weights:



layer 2 weights

20 x 16 x 7 x 7

Weights:

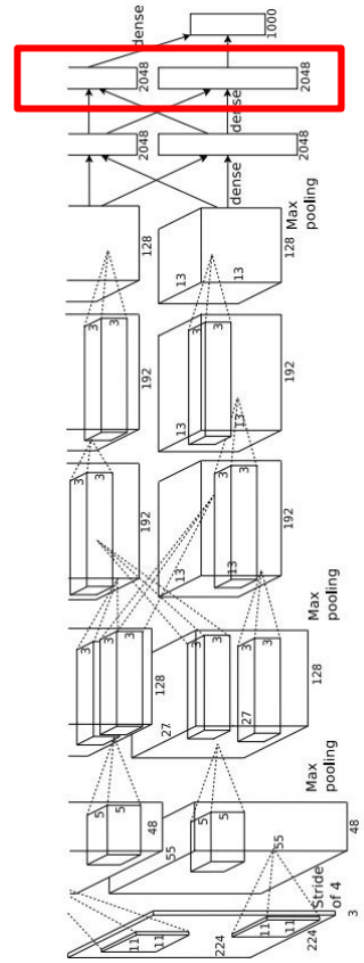
layer 3 weights

20 x 20 x 7 x 7

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

Last Layer

FC7 layer



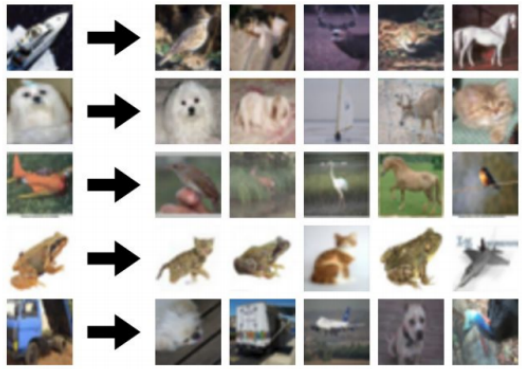
4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors

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

Last Layer

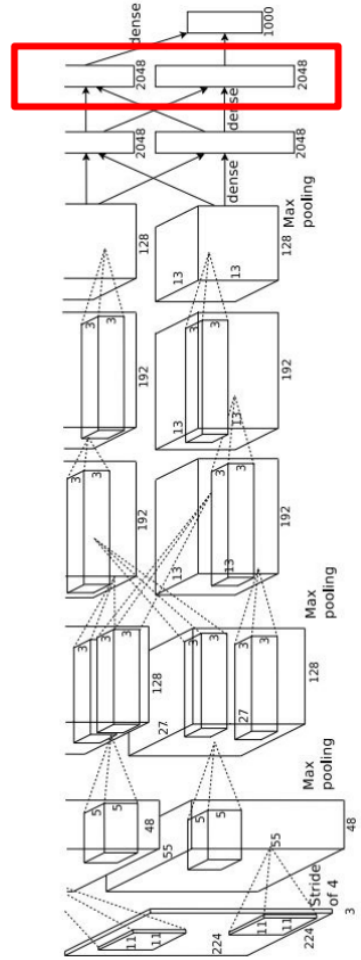
Recall: Nearest neighbors in pixel space



Nearest Neighbors

Test image L2 Nearest neighbors in feature space

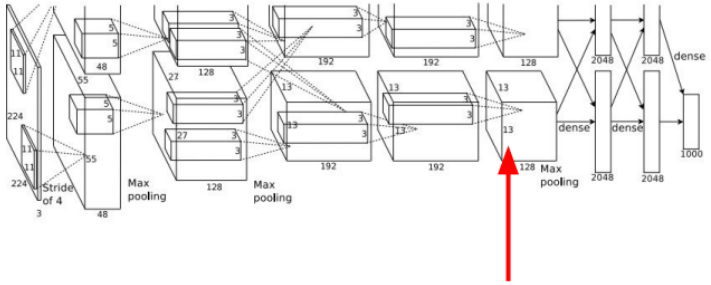
FC7 layer



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

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

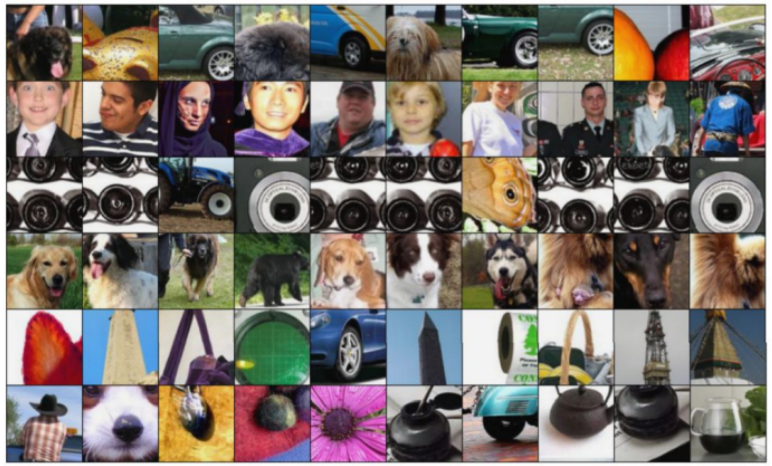
Maximally Activating Patches



Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations

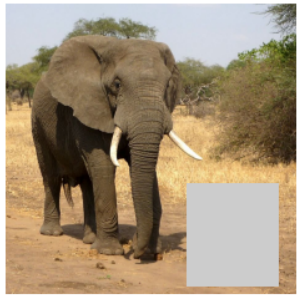


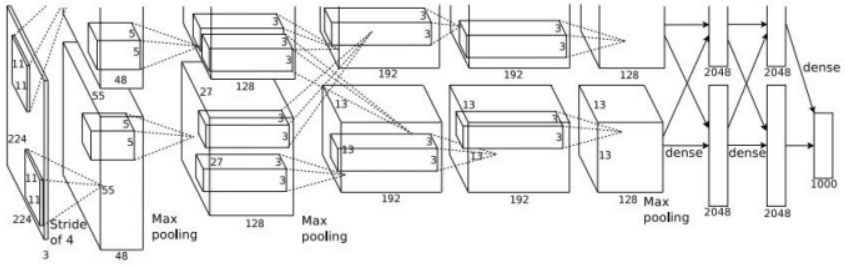
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

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

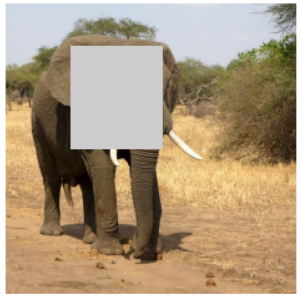
Which pixels matter: Saliency vs Occlusion

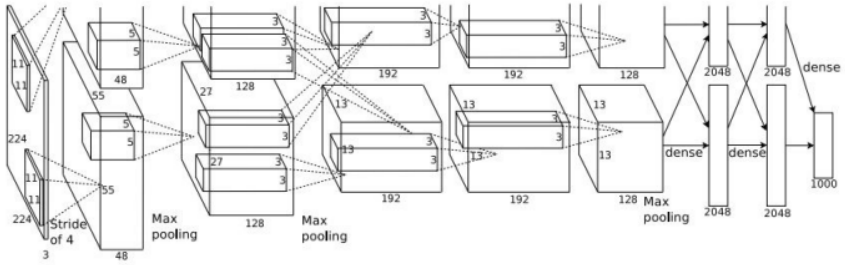
Mask part of the image before feeding to CNN, check how much predicted probabilities change





P(elephant) = 0.95





P(elephant) = 0.75

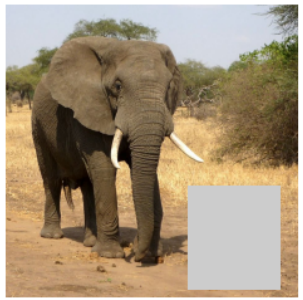
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

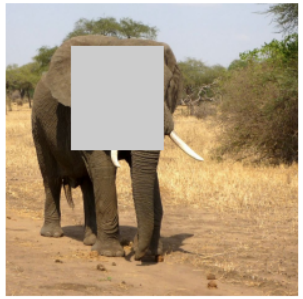
Boat image is CC0 public domain
Elephant image is CC0 public domain
Go-Karts image is CC0 public domain

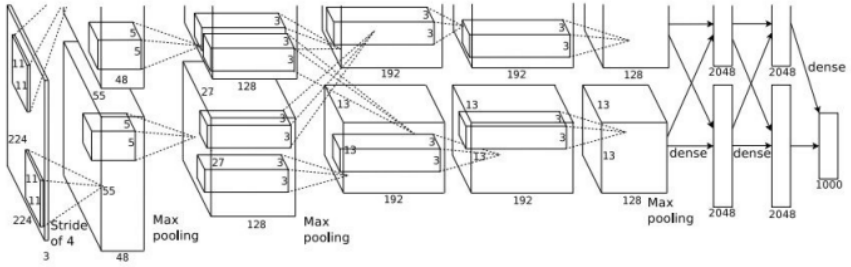
Stanford CS231n: http://cs231n.stanford.edu/

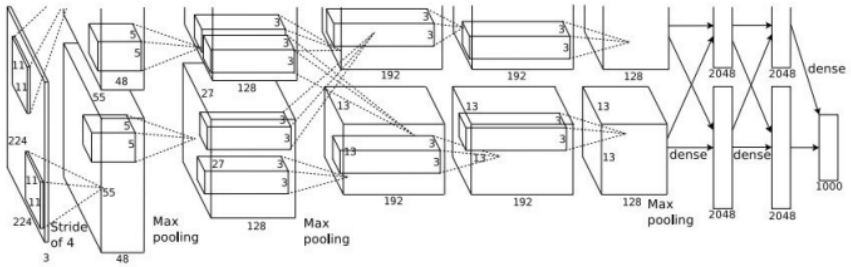
Which pixels matter: Saliency vs Occlusion

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

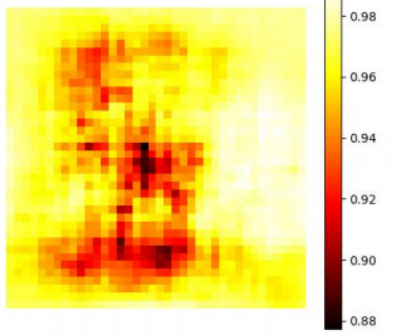




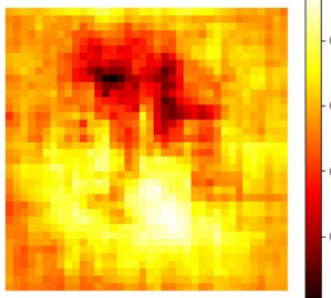




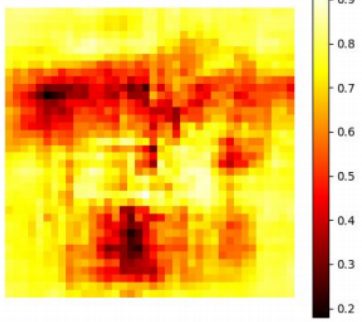
schooner



African elephant, Loxodonta africana



go-kart



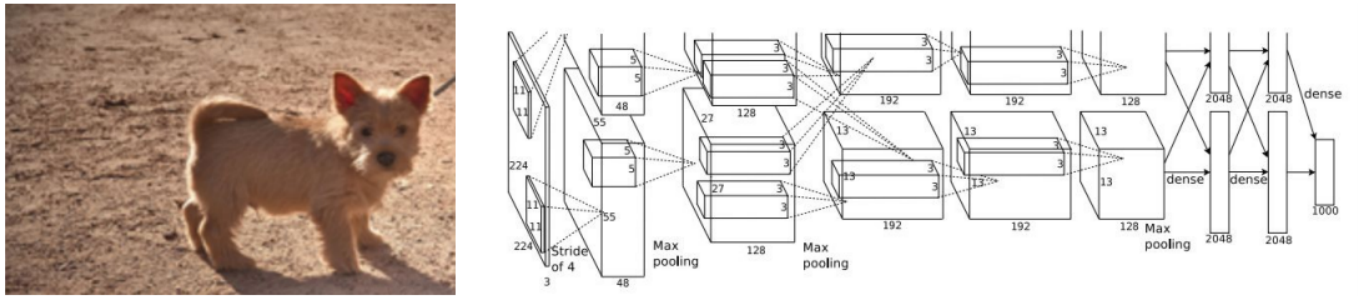
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

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

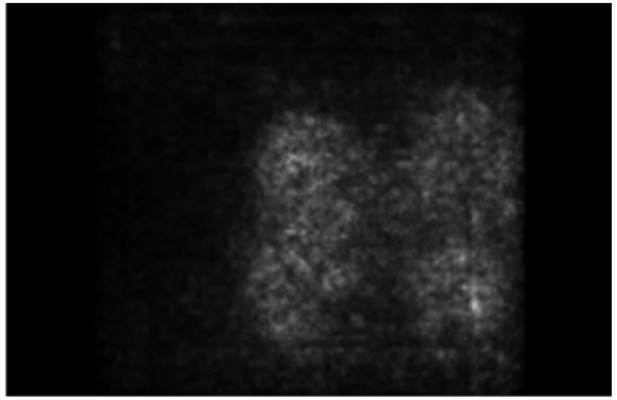
Stanford CS231n: http://cs231n.stanford.edu/

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



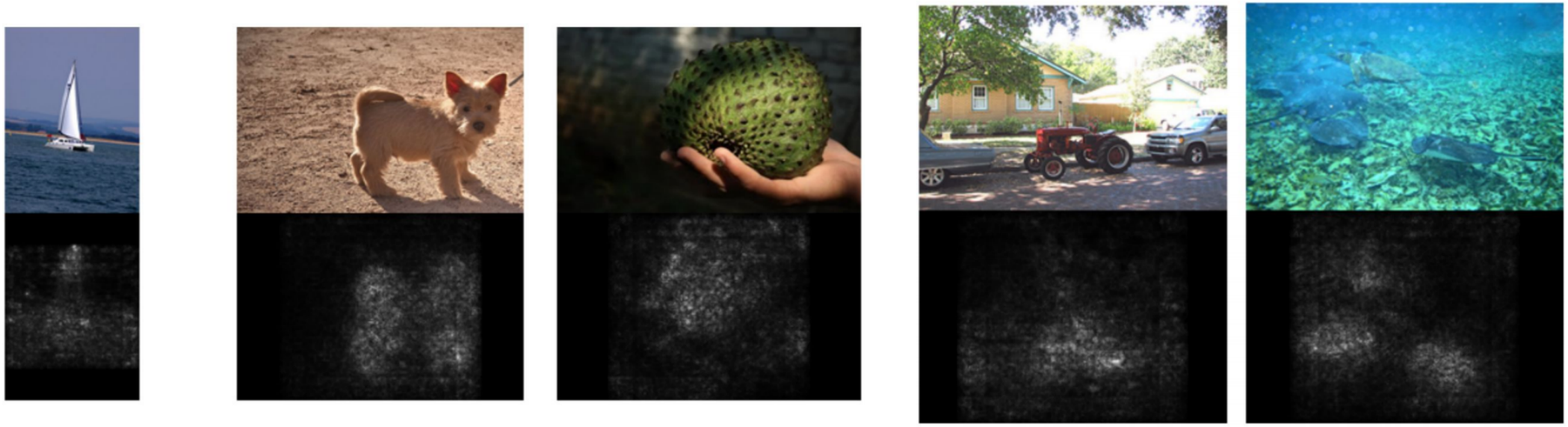
Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



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

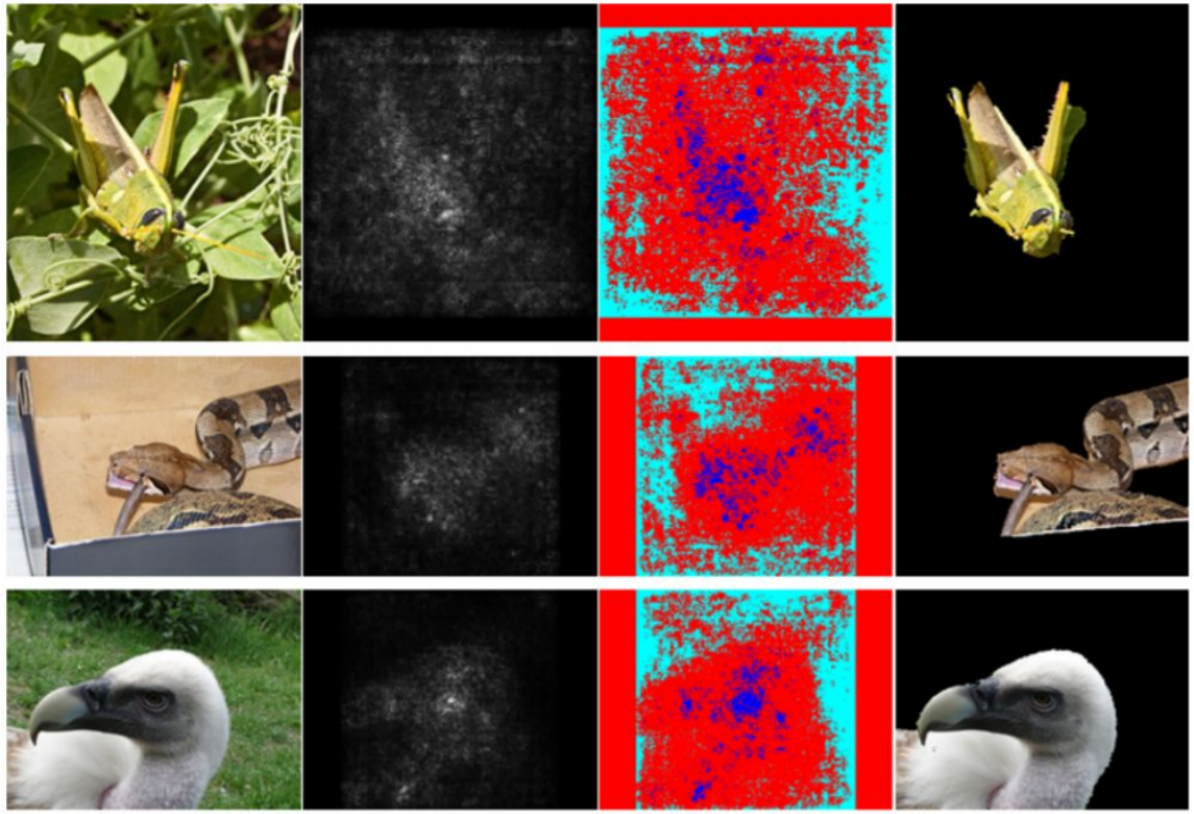


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

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

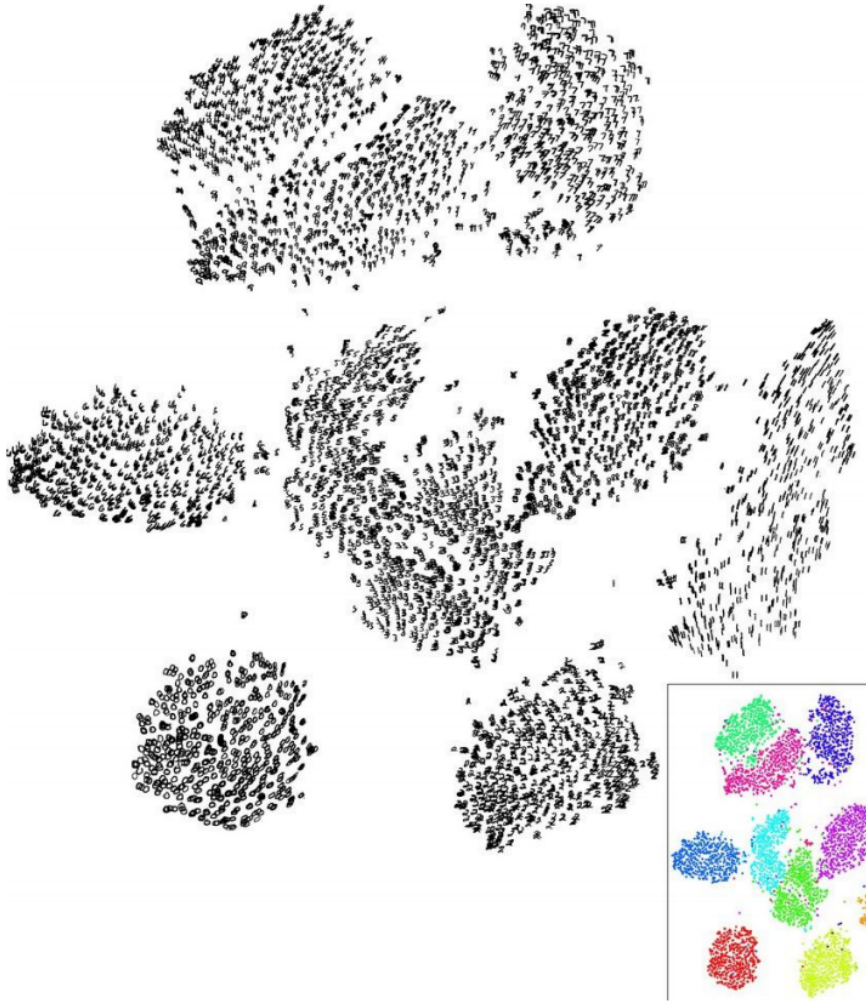
Stanford CS231n: http://cs231n.stanford.edu/

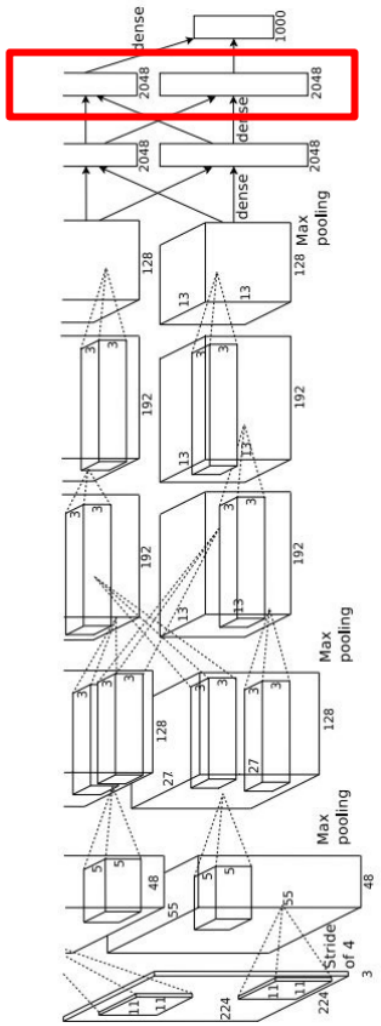
Last Layer: Dimensionality Reduction

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

Simple algorithm: Principal Component Analysis (PCA)

More complex: **t-SNE**





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

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

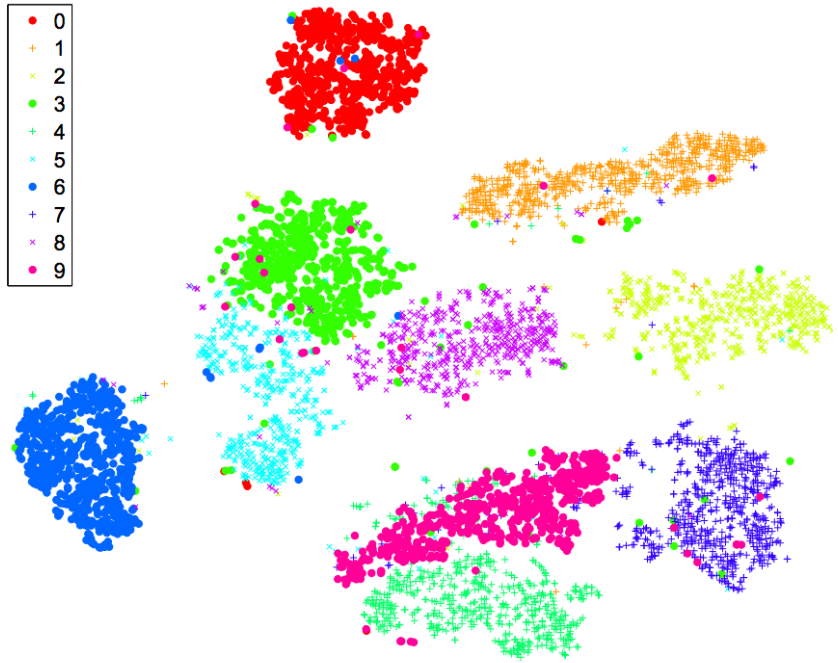
TSNE for data visualization

- 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

TSNE for data visualization

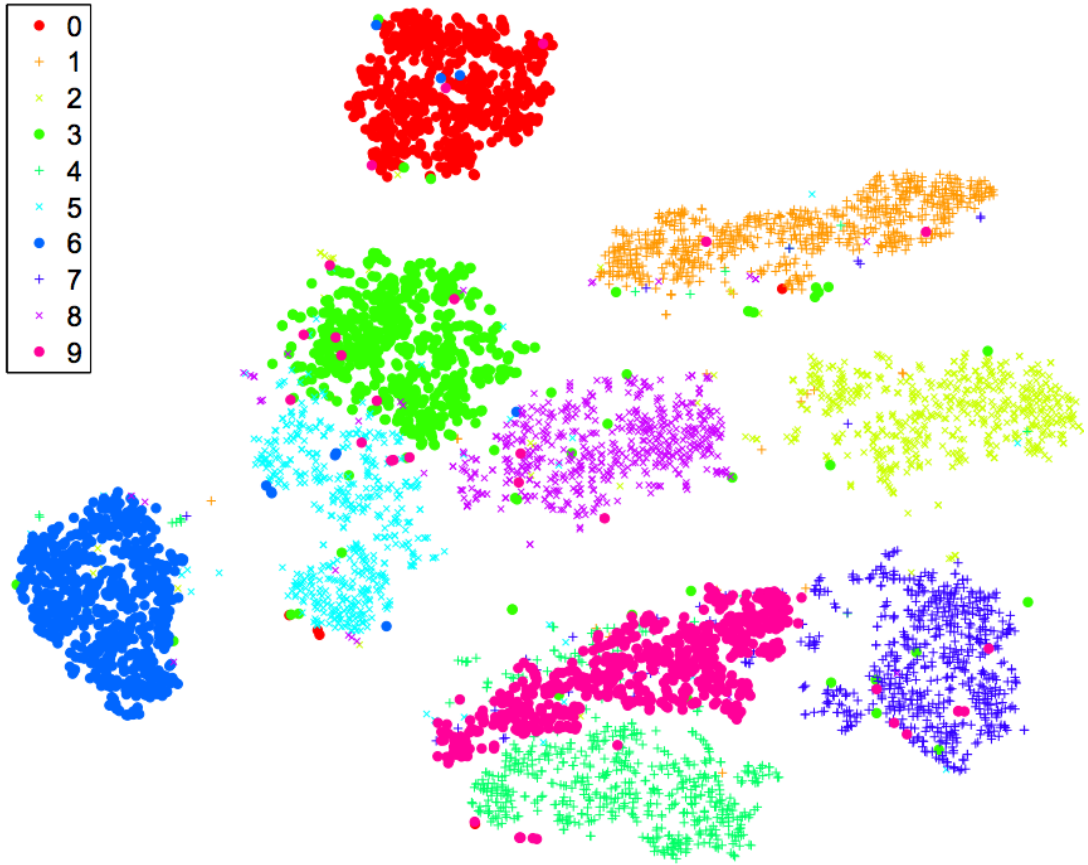
- Reduce data dimensions to enable visualization in 2D or 3D
  - nD -> 2D or 3D
  - Preserve local structure of data to highlight groups
  - Unsupervised – clusters unlabeled data

Applied to MNIST digits



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

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

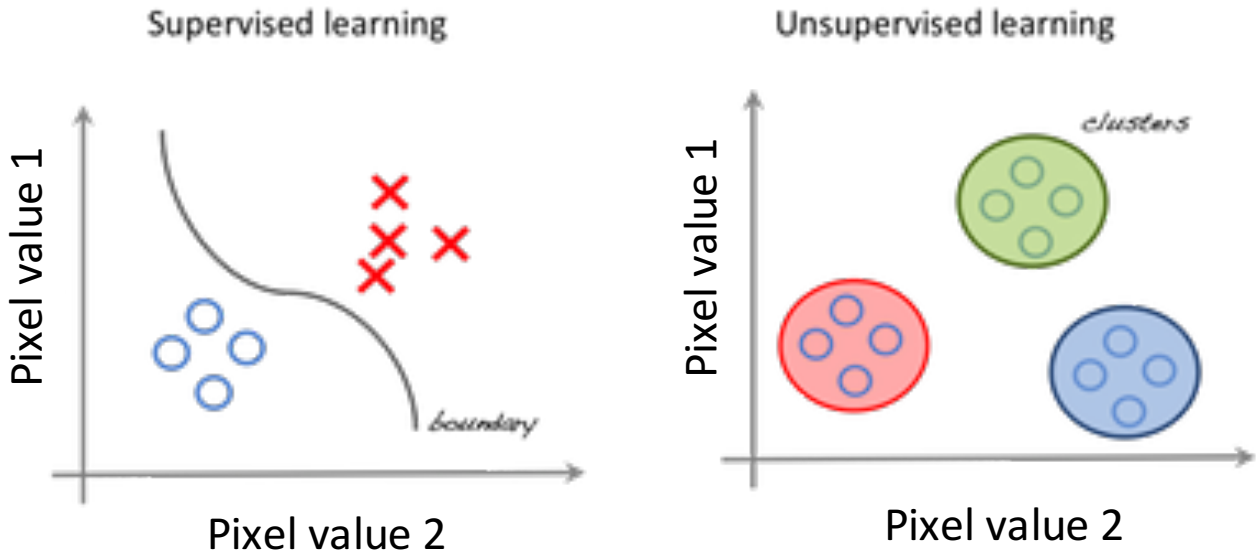


Isn't this good enough?

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

Unsupervised learning in a nutshell

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

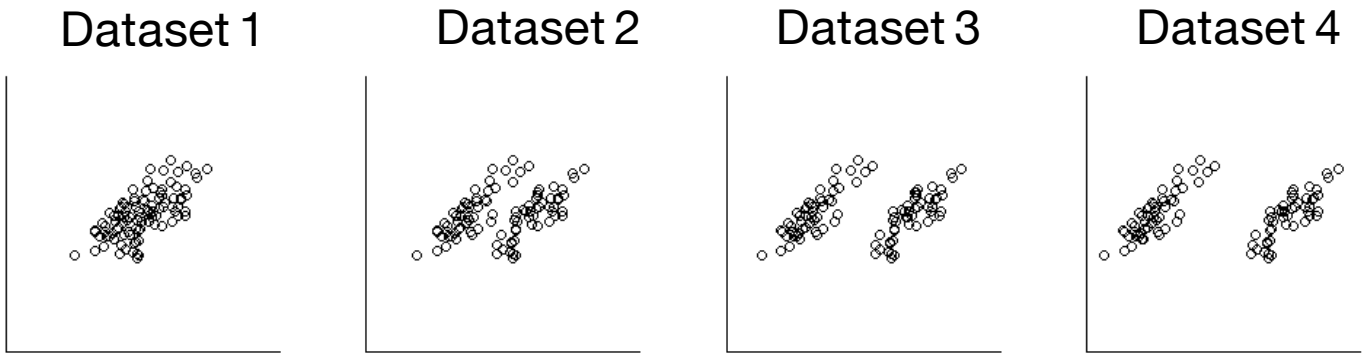


Unsupervised learning in a nutshell

deep imaging

Mathematical tools for finding patterns in data:

- Eigenvector decomposition
- Principal component analysis
- Singular value decomposition



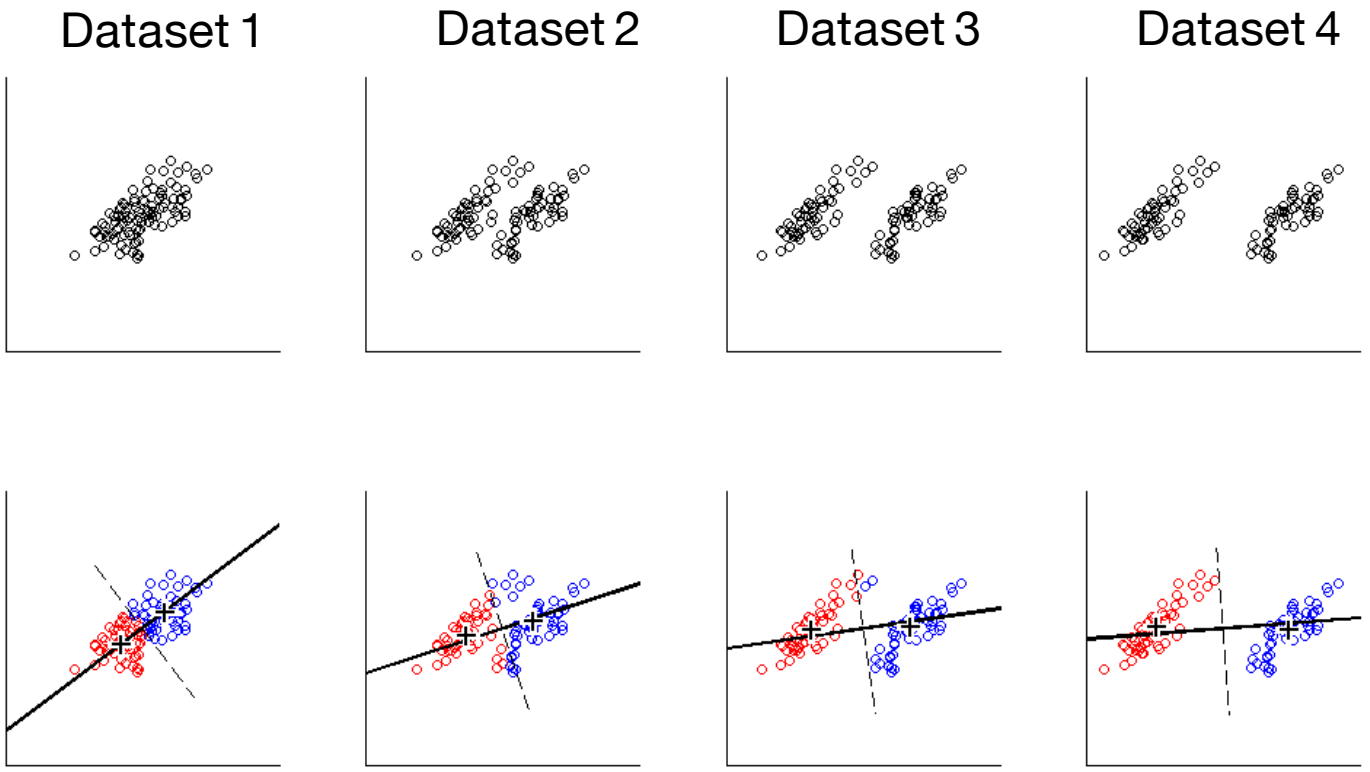
https://stats.stackexchange.com/questions/183236/what-is-the-relation-between-k-means-clustering-and-pca

Unsupervised learning in a nutshell

deep imaging

Mathematical tools for finding patterns in data:

- Eigenvector decomposition
- Principal component analysis
- Singular value decomposition

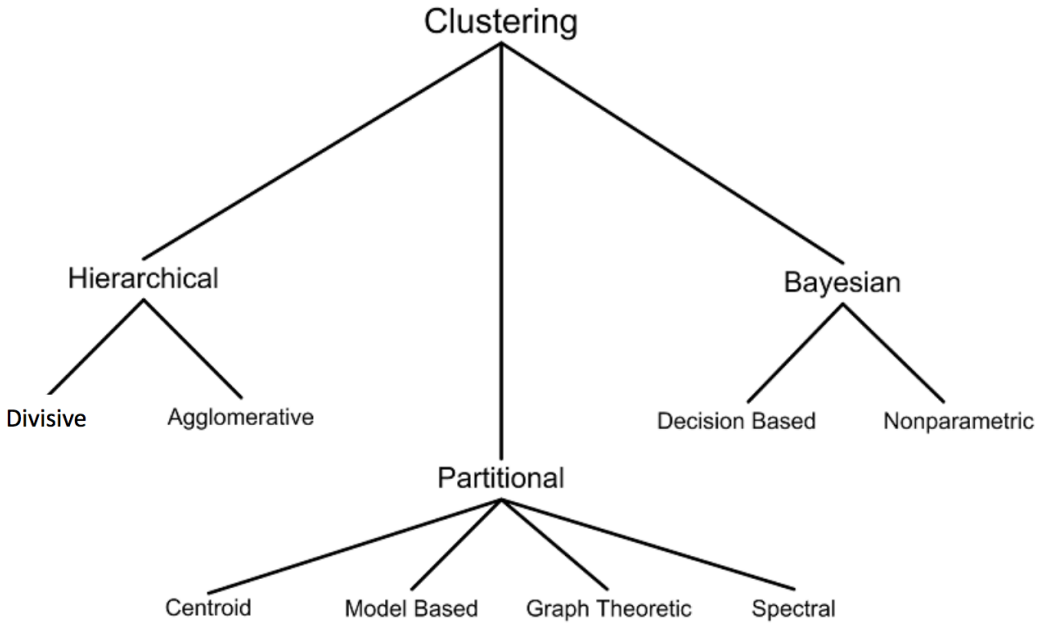


https://stats.stackexchange.com/questions/183236/what-is-the-relation-between-k-means-clustering-and-pca

Iterative methods for unsupervised learning - Clustering

deep imaging

Clustering techniques

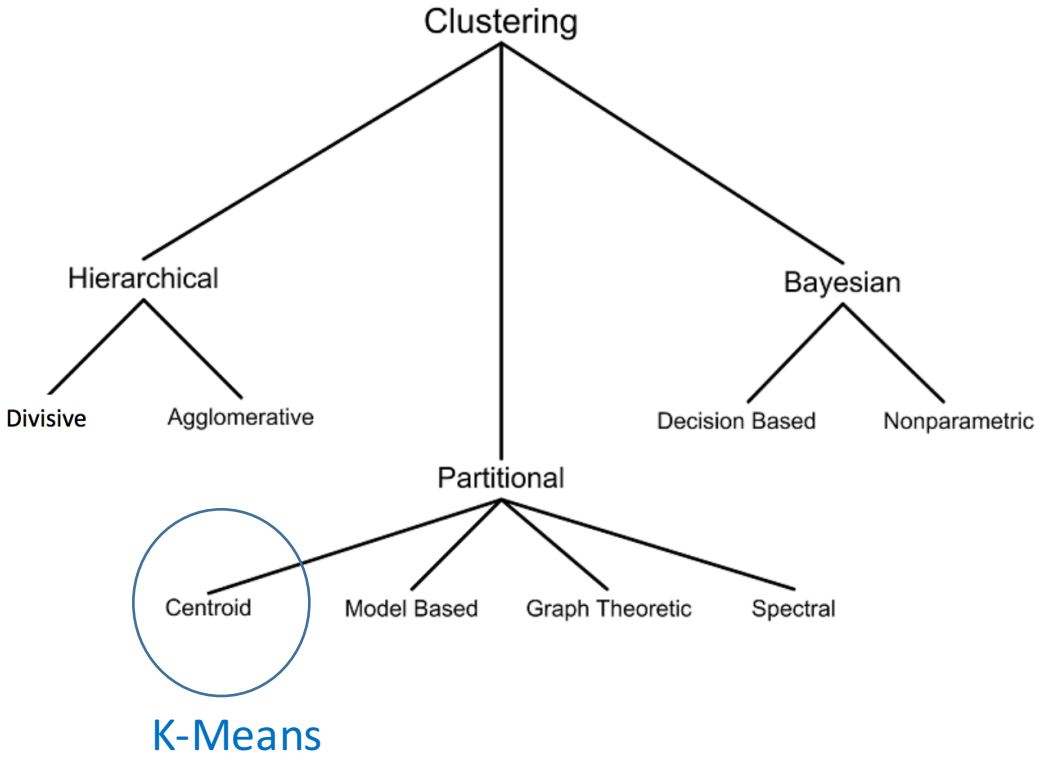


- **Hierarchical** algorithms find successive clusters using previously established clusters. These algorithms can be either agglomerative ("*bottom-up*") or divisive ("*top-down*"):
  1. Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters;
  2. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
- **Partitional** algorithms typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering.
- **Bayesian** algorithms try to generate a *posteriori distribution* over the collection of all partitions of the data.

Iterative methods for unsupervised learning - Clustering

deep imaging

Clustering techniques



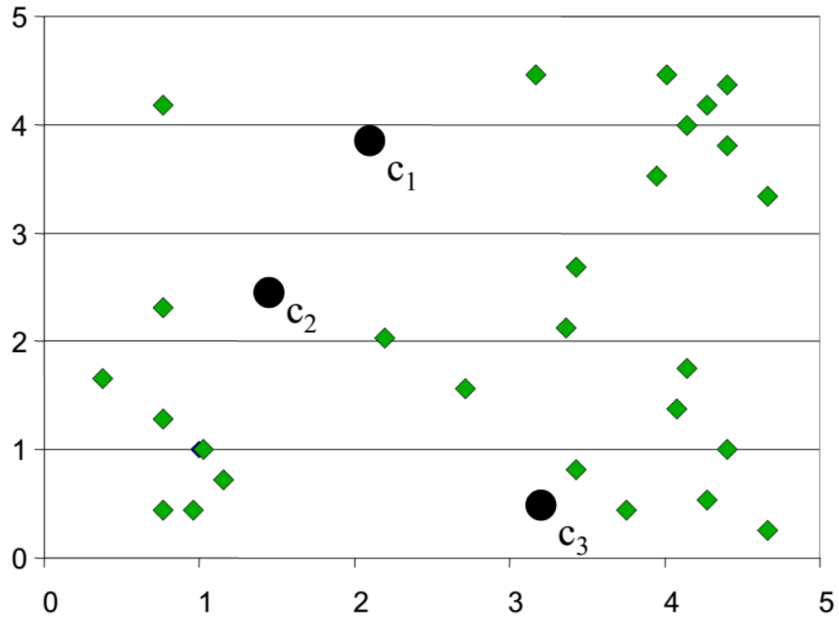
K-Means

- **Hierarchical** algorithms find successive clusters using previously established clusters. These algorithms can be either agglomerative ("*bottom-up*") or divisive ("*top-down*"):
  1. Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters;
  2. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
- **Partitional** algorithms typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering.
- **Bayesian** algorithms try to generate a *posteriori distribution* over the collection of all partitions of the data.

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

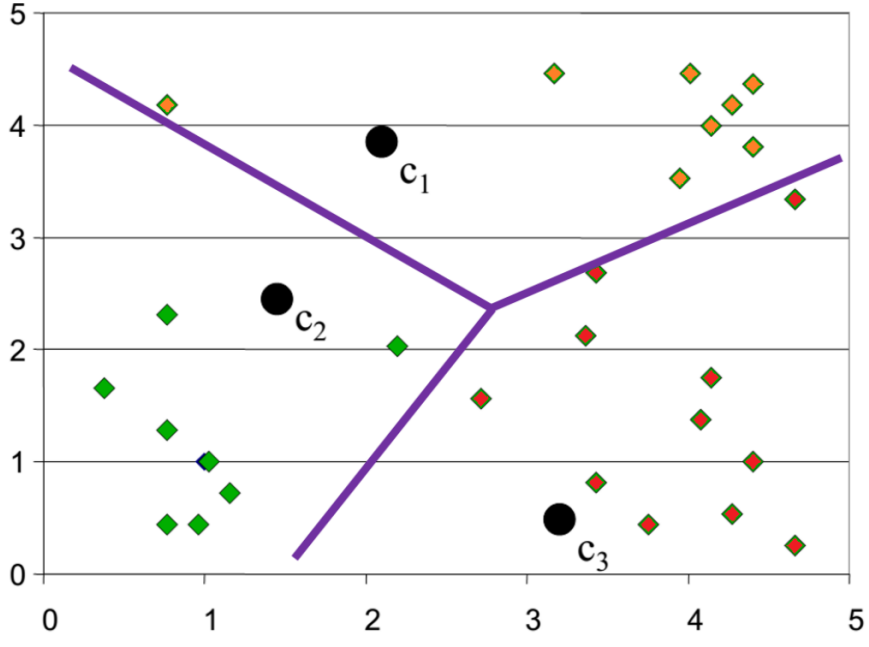
Randomly initialize seeds



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

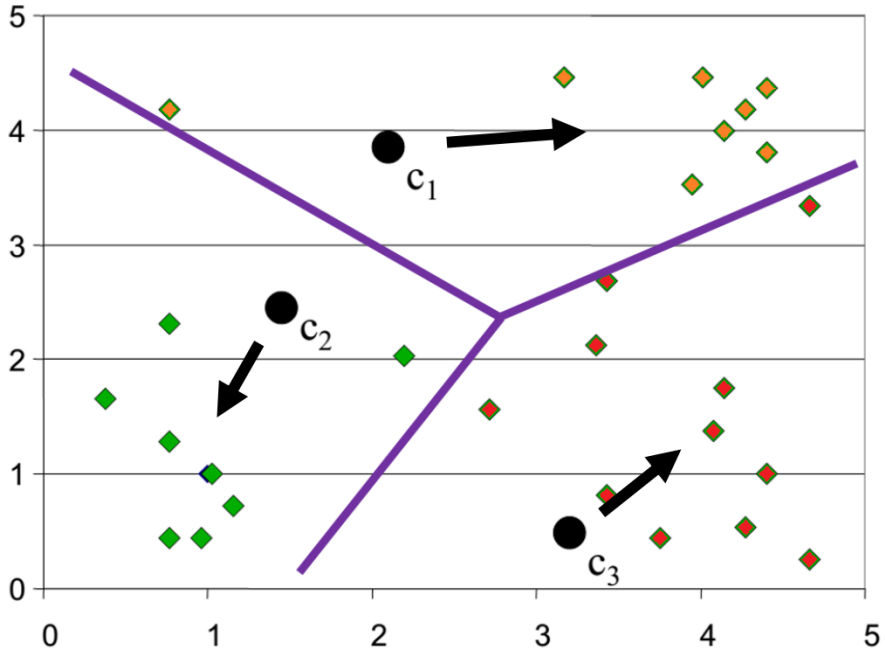
Determine cluster membership for each data point



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

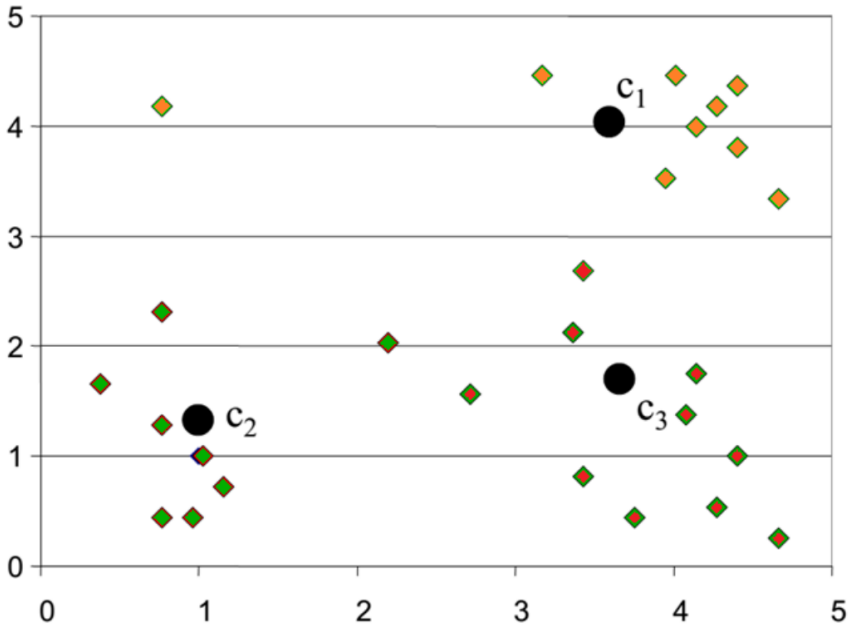
Compute and update new cluster center

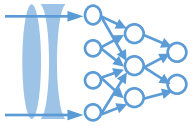


K-Means Clustering

- 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





A good time to talk about the class project!

- Select a “base” dataset
- Simulate parameters of a physical (imaging) system with base dataset
- Train deep neural net with simulated dataset
- Report results

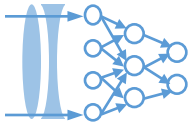
What you’ll need to submit:
1) The project’s source code

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) An 8-minute presentation that each student will deliver to the class

Projects from BME590: Machine Learning in Imaging



Finding Ultrasound Sub-apertures for Liver Vessel Segmentation
Single-Pixel, Single-Frequency Hand Gesture Recognition with a Dynamic Metasurfaces
Going Deeper: Depth Image Classification via simulated SPAD array images
Trained Blur Kernel for histology slide segmentation using a Deep Neural Network
Classification of Tuberculosis Bacilli With and Without Staining
A deep learning approach to improving ultrasonic plane wave imaging
Automated Image Focus Detecting Algorithm for Low-Cost Handheld Microscope
Optimal shift-variant point-spread function for improved classification
Deep Learning for Motion Tracking on the Micron Scale with Ultrasound
Sensor Multiplexing and Reconstruction for Color Images
Noise Reduction in Optical Coherence Tomography using a Deep Image Prior
Optimization of illumination for Unet-Base Cervix Segmentation
HDR image reconstruction with filters over pixels – What is the optimal design?
Detection of Lesions in Variably Noisy Ultrasound Images Using Machine Learning
Methods for Segmentation of Fine Structure in Rodent Histological Specimens
Direct reconstruction network for photoacoustic imaging with fewer measurements
Machine Learning for Ultrasound Lesion Mapping with Apodization Optimization
Resolution versus Precision in X-ray detection of Pneumonia
Optimizing illumination for overlapped image classification