deep imaging

Lecture 24: Looking Ahead + Review

Machine Learning and Imaging

BME 548L
Roarke Horstmeyer

Announcements and schedule

- Today is the last lecture! (no lectures next week)

- HW4 being graded, HW5 will be graded quickly once turned in

- Homework #5 Due Wed April 24 at 11:59pm

- Final code and presentation due via email: Thursday May 2, 5pm

- Final presentation time slots: Thursday May 2, 7pm - 10pm
- Sign up for slot at google sheets link will be shared via Ed Discussion

- Final presentation paper write-up, website template and permission form due: Saturday May 4 at 11:59pm

- Project help:
  - We will continue lab sessions this week/next week
    - Office hours Tuesday 11am – noon
    - Email me if you'd like to meet another time
    - Email TA's / reach out on Canvas with questions!

Project content details –

1. Start with code-base and annotated data
   - e.g., downloaded or from existing work

2. Make sure it works and gives reasonable results
   - work with TA's and others to make sure code runs successfully!

3. Main project component – experiment and explore with code and data

   A. do not just randomly alter neural network architecture

   B. instead, explore something meaningful about how image data was acquired, properties of the image data, different ML-related goals for data

   C. One useful direction is to incorporate "physical layers" – trainable weights that optimize some aspect of image capture. But this is an effective way to explore point B above!

4. Discuss insights grained from step 3

Components of final project

See https://deepimaging.github.io/proj-info/

38% of total grade

1. Presentation Slides – 10%
   - 7-minute presentation, 2 minute for questions
2. 4-6 page write up with at least 3 figures and 5 references – 20%
   - Introduction, related work, methods, results, discussion
3. Code used for final results in folder or .ipynb’s – 4%
4. brief website template & permission to share results – 4%
5. shared annotated datasets & permissions – no grade, but would be much appreciated if using an interesting dataset

Final project webpage

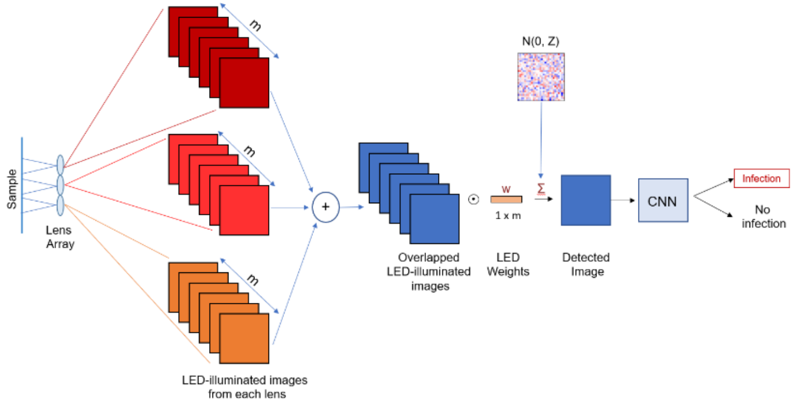
- Must be submitted
- Will share template
- Will post to deepimaging.io with permission
- Will also send permission form, which must be submitted with final project as well

Optimizing illumination for overlapped classification

Amey A Chaware

amey.chaware@duke.edu

Paper PDF



This project presents an imaging system that simultaneously captures multiple images and automatically classifies their contents to increase detection throughput. Our optical design consists of a set of multiple lenses that each image a unique field-of-view onto a single image sensor. The resulting "overlapped" image exhibits reduced contrast, but includes measurements from across a proportionally larger viewing area. We then post-process this overlapped image with a deep convolutional neural network to classify the presence or absence of certain features of interest. We examine the specific case of detecting the malaria parasite within overlapped microscope images of blood smears. We demonstrate that it is possible to overlap 7 unique images onto a common sensor while still offering accurate classification of the presence or absence of the parasite, thus offering a 7x potential speed-up for automated disease diagnosis with microscope image data. Additionally, we explore the use of supervised deep-learning network to jointly optimize the physical setup of an optical microscope to improve automatic image classification accuracy in overlapped imaging. We take advantage of the wide degree of flexibility available in choosing how a sample is illuminated in a microscope to design a specific pattern of light that leads to a better performance.

Paper:

- Paper PDF

Code and Data:

- You can provide a link to your code here: Code

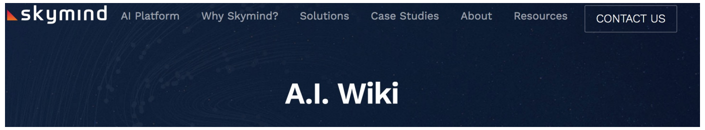
deep imaging

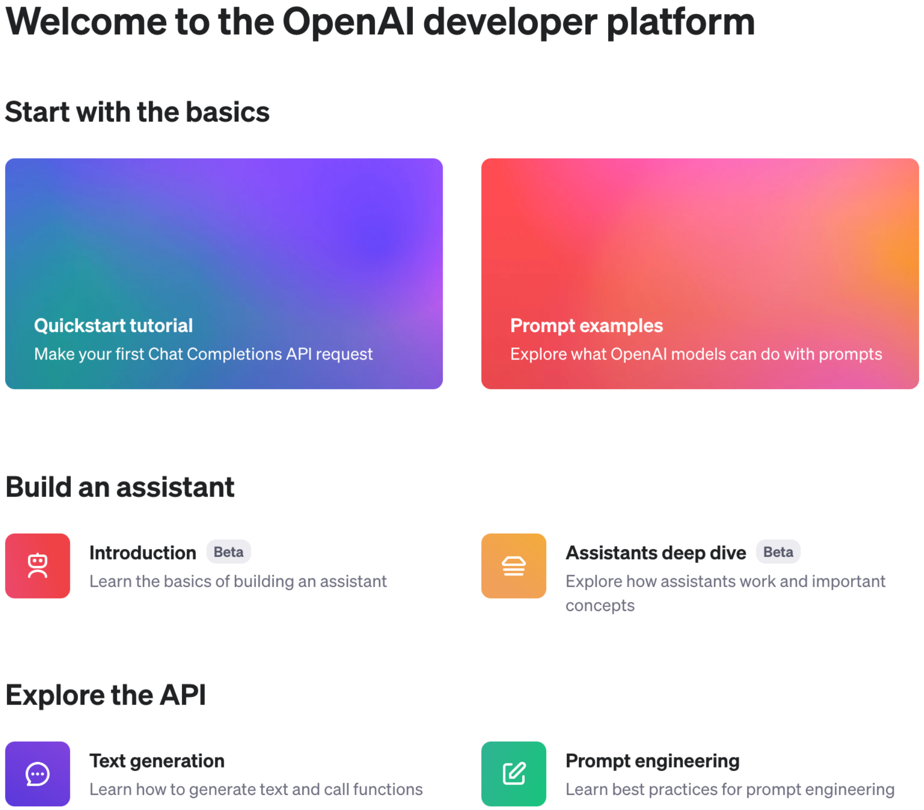
Please complete course evaluations!

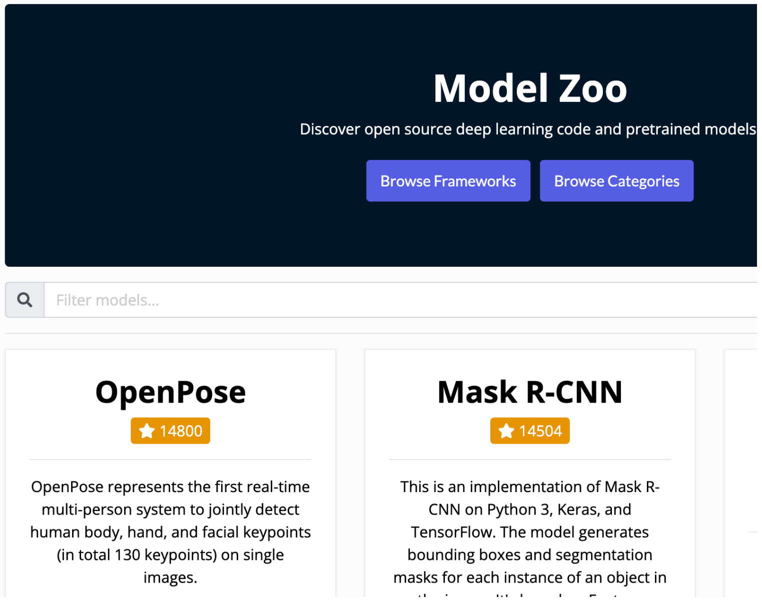
Where are things going with Machine Learning and Imaging in 10 years?

Where are things going with Machine Learning and Imaging in 10 years?

1. Proliferation of trained models, similar datasets, and novel programming languages







Where are things going with Machine Learning and Imaging in 10 years?

1. Proliferation of trained models, similar datasets, and novel programming languages

Environment developed by Facebook (Meta) – arguably more popular for research use

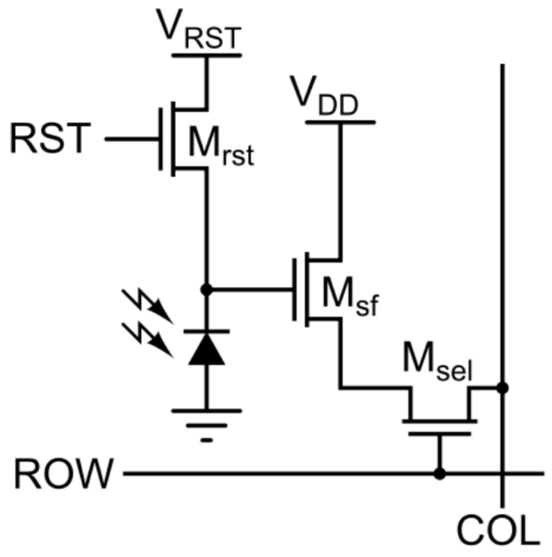
Jax: Rapidly deploy the same code on GPU's and TPU's

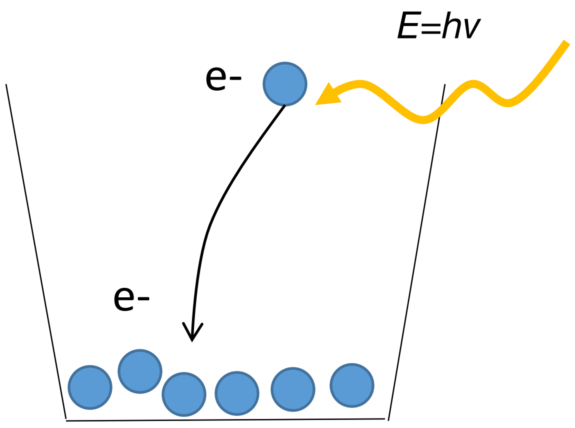
Where are things going with Machine Learning and Imaging in 10 years?

deep imaging

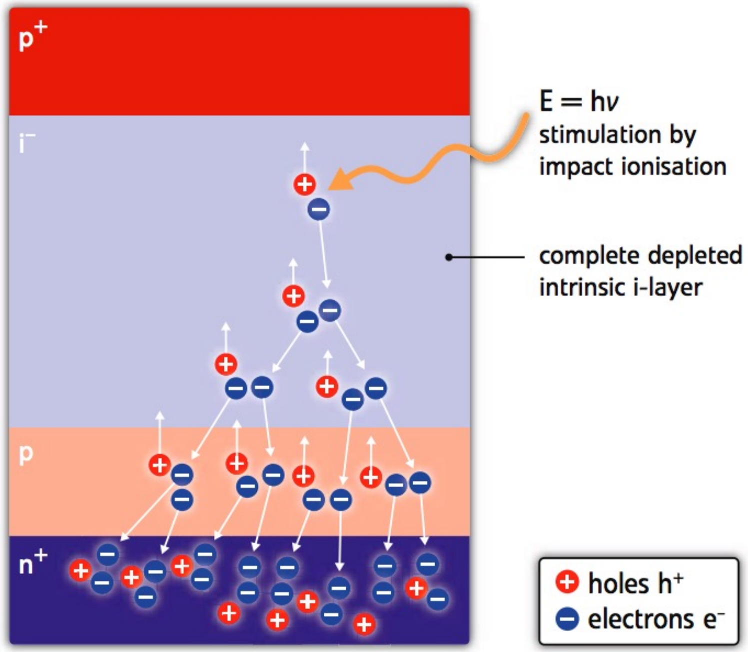
2. "Cameras" on many devices & new types of sensors

Standard CMOS pixel = bucket that collects electrons





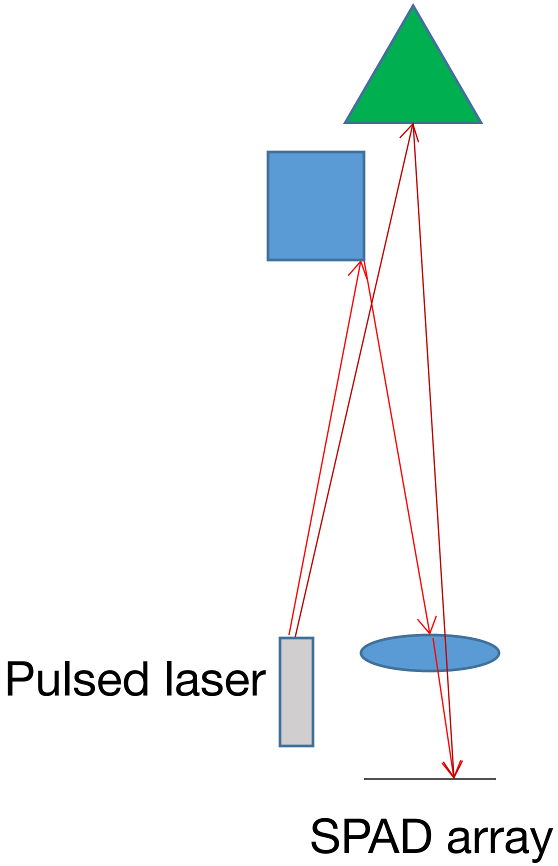
SPAD pixel: was there a photon or not?



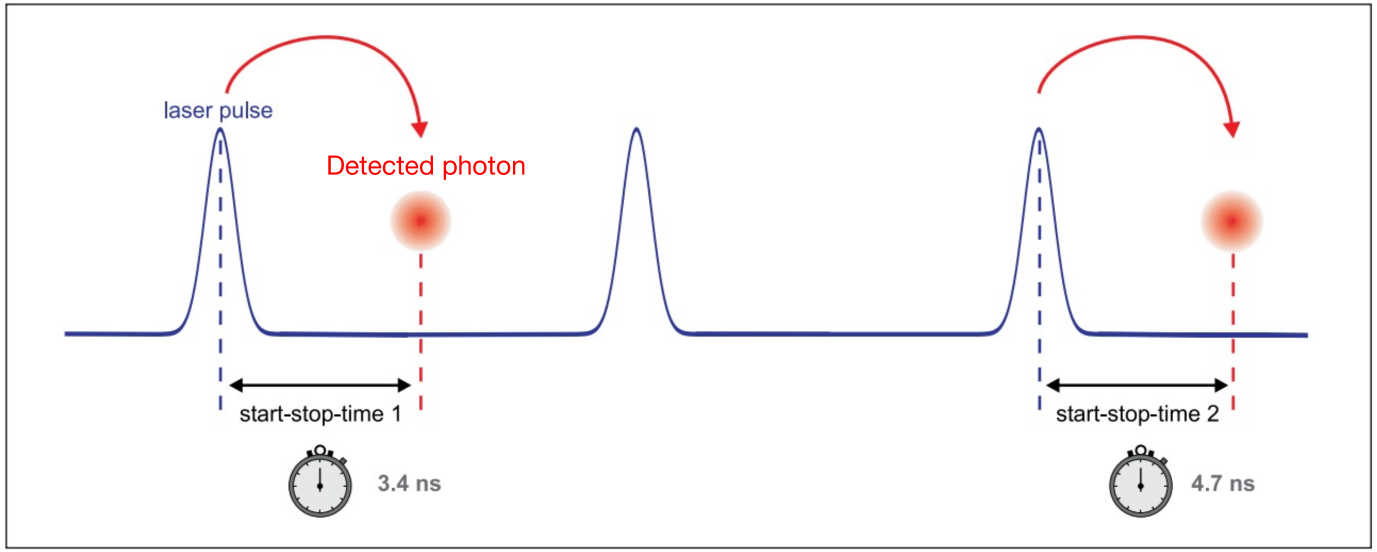
Where are things going with Machine Learning and Imaging in 10 years?

deep imaging

2. “Cameras” on many devices & new types of sensors



- Light travels 1 ft in 1 ns.
- SPADs can precisely photon arrival time to measure travel distance (TOF)

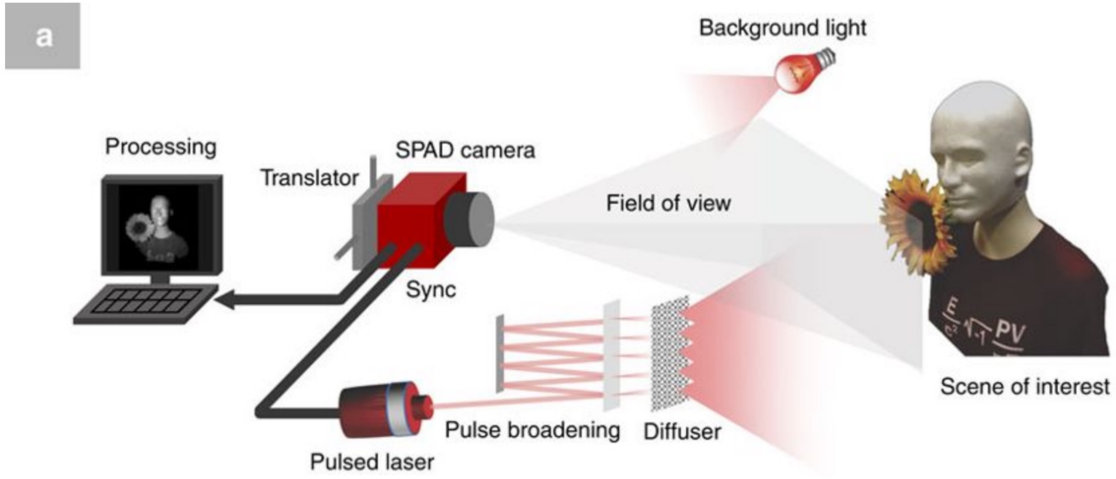


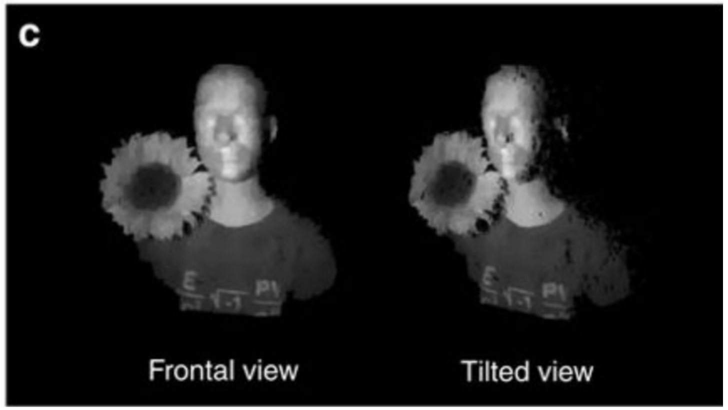
https://www.picoquant.com/images/uploads/page/files/7253/technote_tcspc.pdf

Where are things going with Machine Learning and Imaging in 10 years?

deep imaging

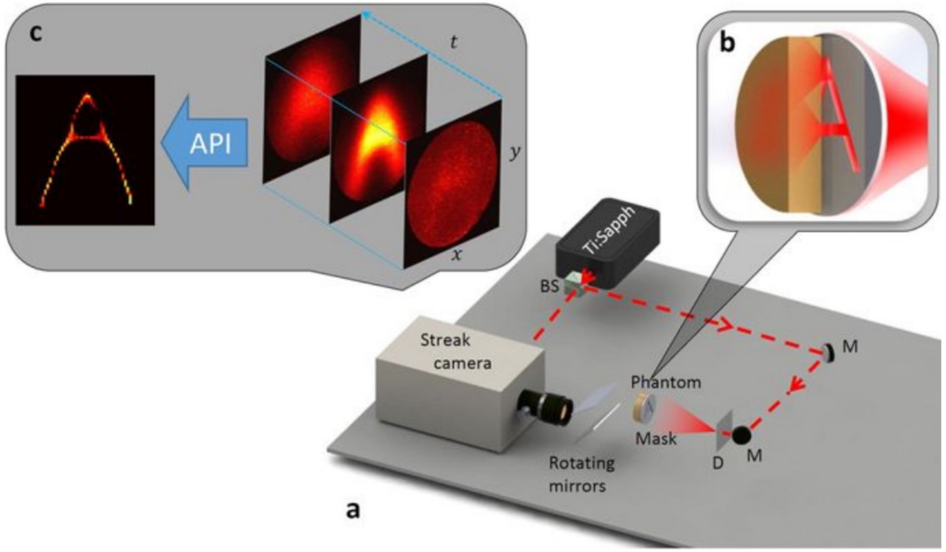
2. "Cameras" on many devices & new types of sensors





D. Shen et al, https://www.nature.com/articles/ncomms12046

Figure 1: Imaging Through Thick Scattering.



G. Satat et al, https://www.nature.com/articles/srep33946

LiDAR Scanner

LiDAR (Light Detection and Ranging) is used to determine distance by measuring how long it takes light to reach an object and reflect back. It is so advanced, it’s being used by NASA for the next Mars landing mission. And it’s now been engineered to fit in the thin and light iPad Pro.

The custom-designed LiDAR Scanner uses direct time of flight to measure reflected light from up to five meters away, both indoors and out. It works at the photon level, operates at nanosecond speeds, and opens up tremendous possibilities for augmented reality and beyond.

Where are things going with Machine Learning and Imaging in 10 years?

deep imaging

3. Beyond convolutions - new constructs for deep networks

Where are things going with Machine Learning and Imaging in 10 years?

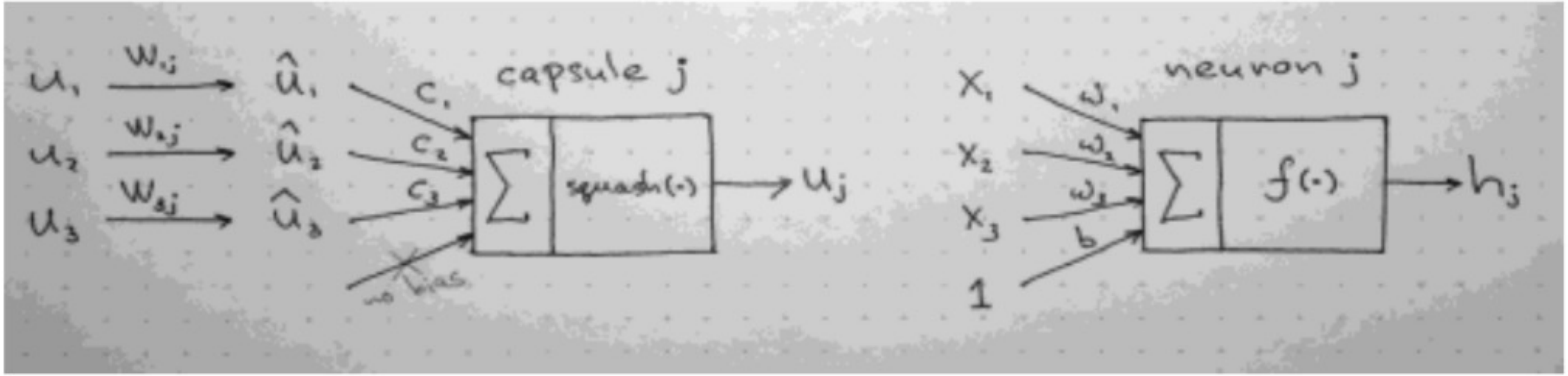
deep imaging

3. Beyond convolutions - new constructs for deep networks

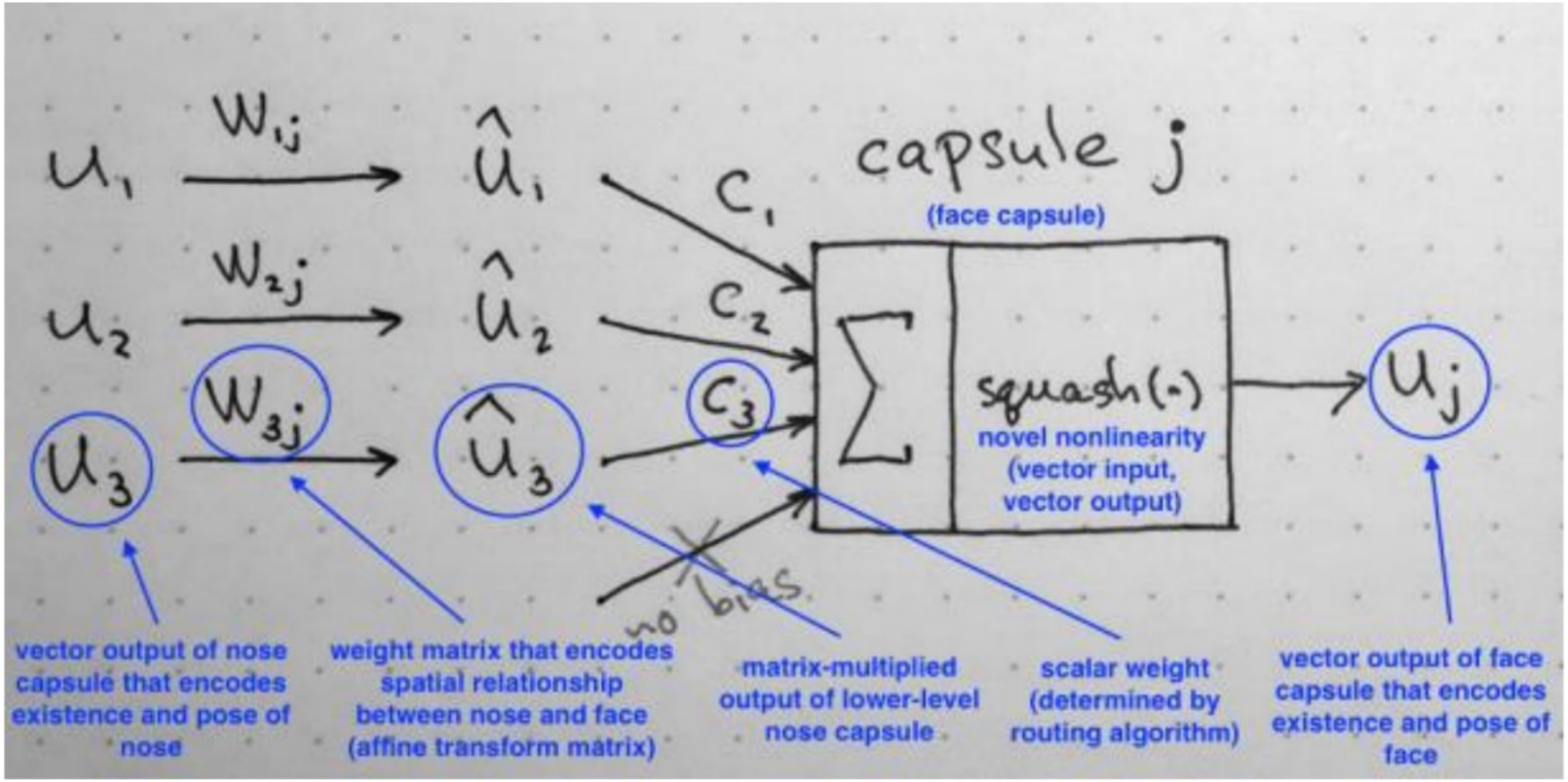
Dynamic Routing Between Capsules

Sara Sabour **Nicholas Frosst**

Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com



u_1 W_{1j} $\hat{u}_1$ c_1
u_2 W_{2j} $\hat{u}_2$ c_2
u_3 W_{3j} $\hat{u}_3$ c_3
capsule j
Σ squash(·)
u_j
no bias
neuron j
x_1 ω_1
x_2 ω_2
x_3 ω_3
1 b
Σ $f(\cdot)$
h_j



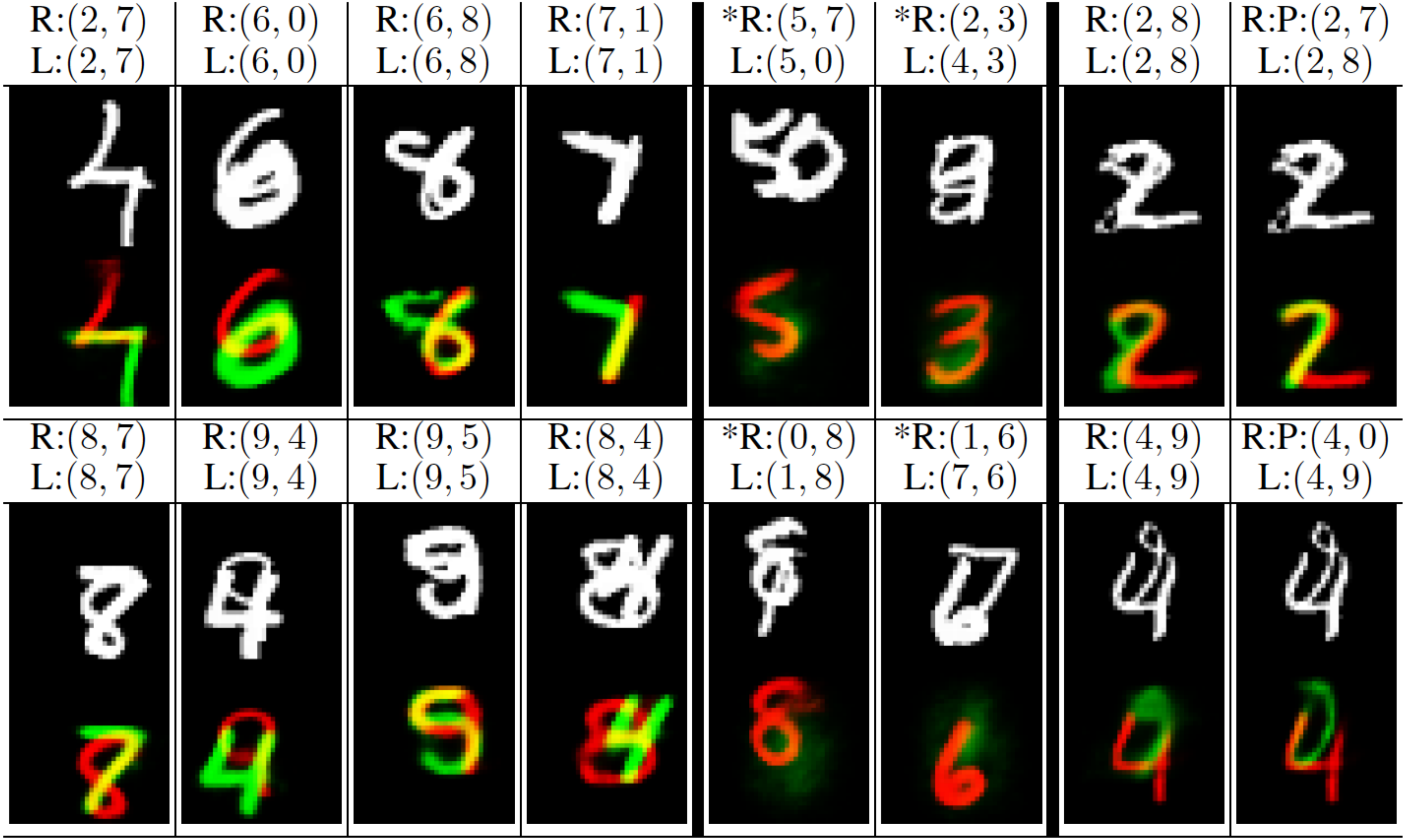
u_1 W_{1j} $\hat{u}_1$ c_1
u_2 W_{2j} $\hat{u}_2$ c_2
u_3 W_{3j} $\hat{u}_3$ c_3
capsule j
(face capsule)
Σ squash(·)
novel nonlinearity (vector input, vector output)
u_j
no bias
vector output of nose capsule that encodes existence and pose of nose
weight matrix that encodes spatial relationship between nose and face (affine transform matrix)
matrix-multiplied output of lower-level nose capsule
scalar weight (determined by routing algorithm)
vector output of face capsule that encodes existence and pose of face

Where are things going with Machine Learning and Imaging in 10 years?

3. Beyond convolutions - new constructs for deep networks

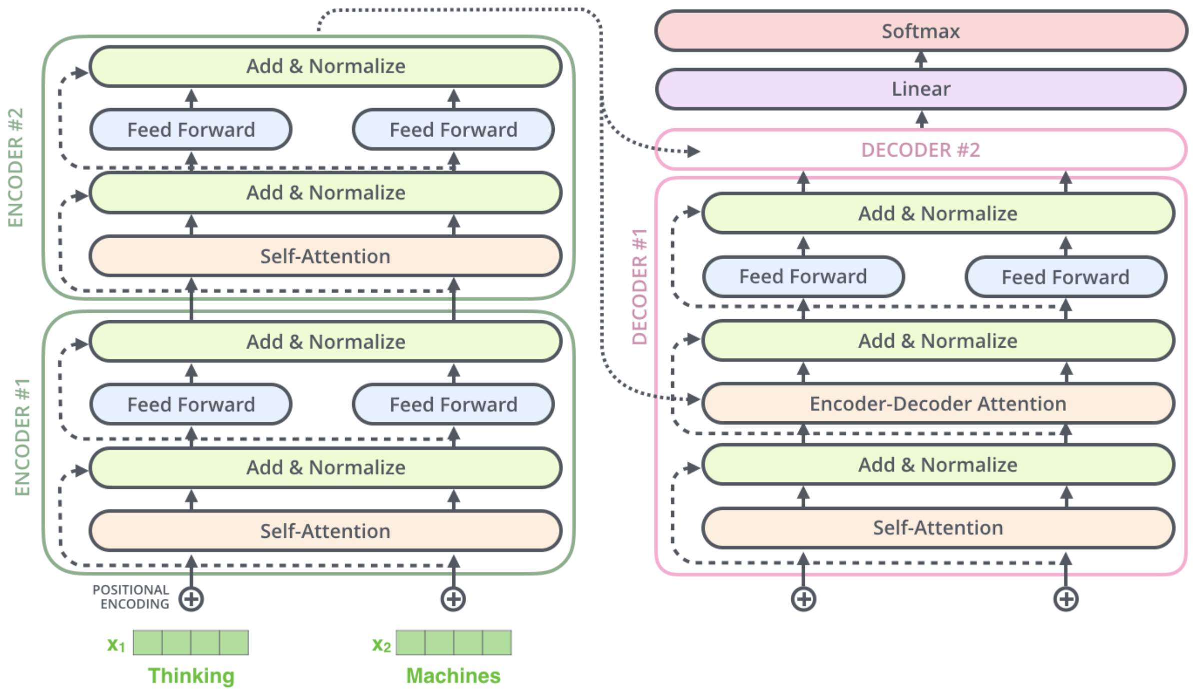
Capsule vs. Traditional Neuron			
Input from low-level capsule/neuron		vector($\mathbf{u}_i$)	scalar(x_i)
Operation	Affine Transform	$\hat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear Activation	$\mathbf{v}_j = \frac{\\|\mathbf{s}_j\\|^2}{1+\\|\mathbf{s}_j\\|^2}\frac{\mathbf{s}_j}{\\|\mathbf{s}_j\\|}$	$h_j = f(a_j)$
Output		vector($\mathbf{v}_j$)	scalar(h_j)

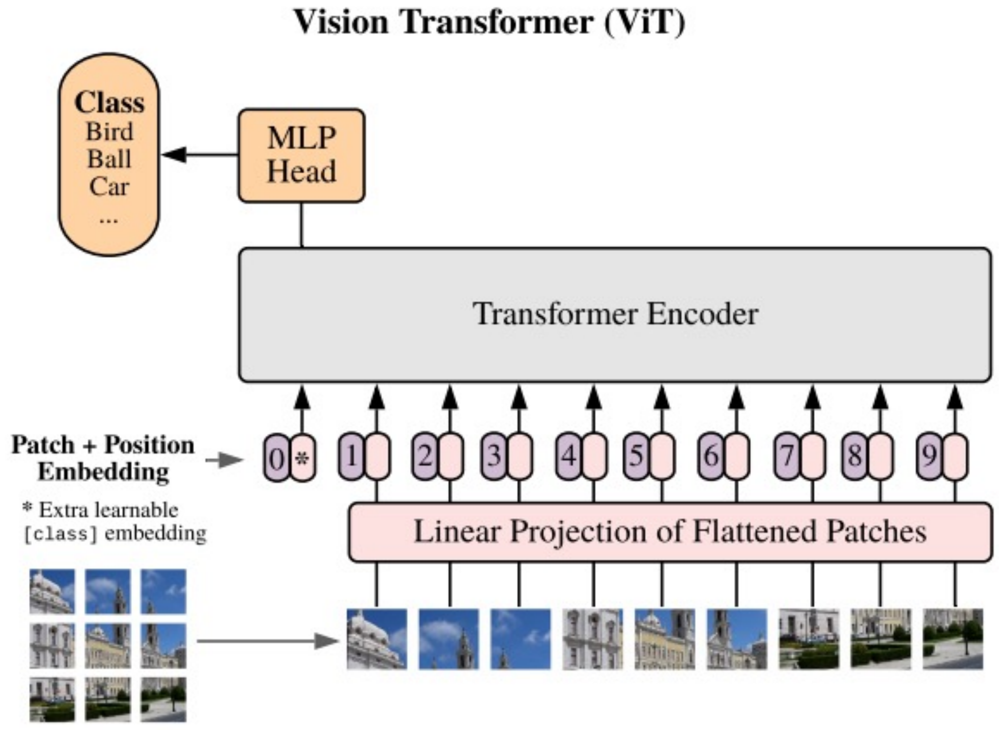
3. Therefore it will not assign one pixel to two digits if one of them does not have any other support.



Example 2: Transformers for image analysis

Transformers for text analysis/generation



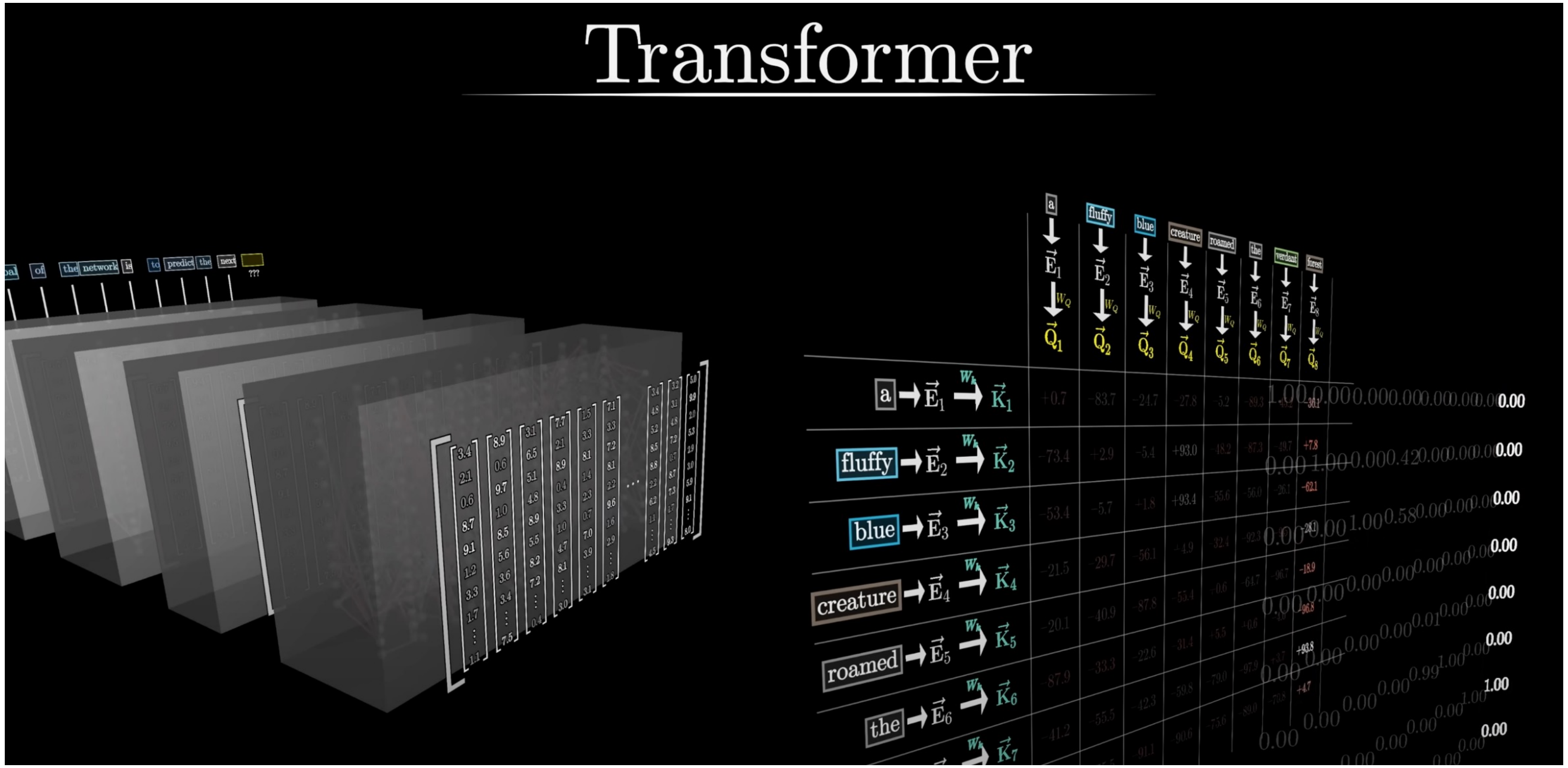


A. Dosovitsky et al, "AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE"

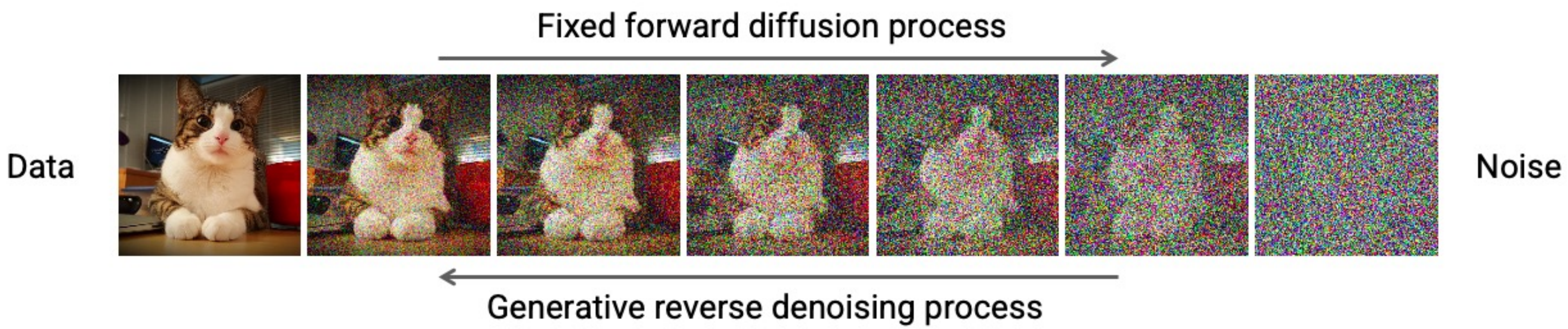
deep imaging

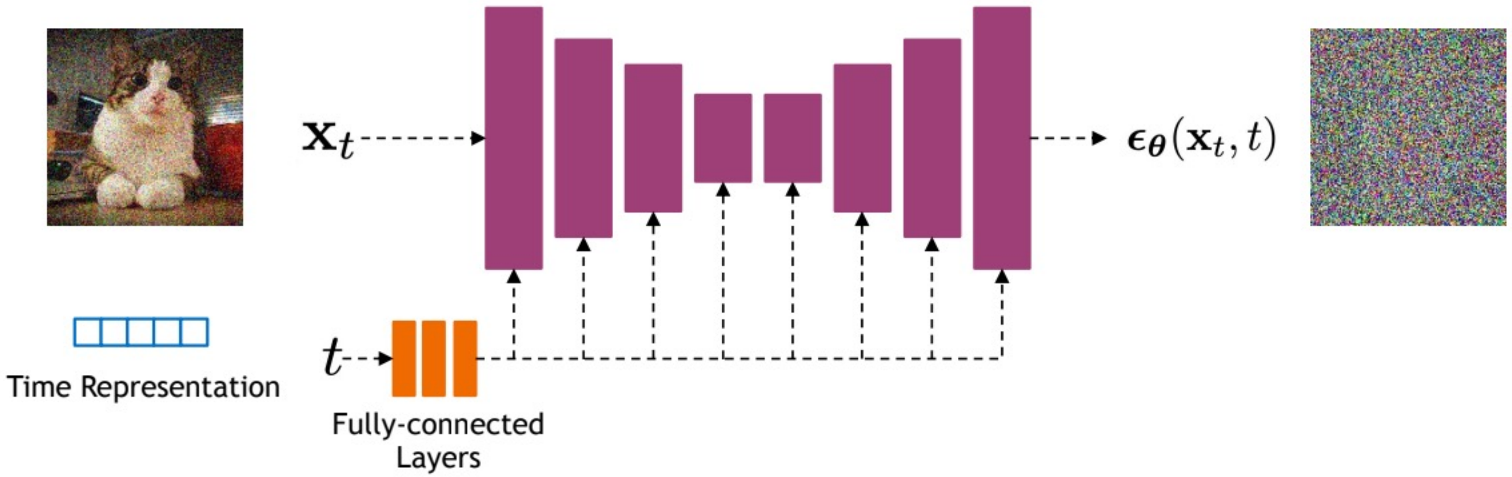
A visual introduction to transformers

- https://www.youtube.com/watch?v=wjZofJX0v4M



Example 2: Diffusion models for image generation and analysis





deep imaging

https://research.nvidia.com/labs/toronto-ai/VideoLDM/

Where are things going with Machine Learning and Imaging in 10 years?

deep imaging

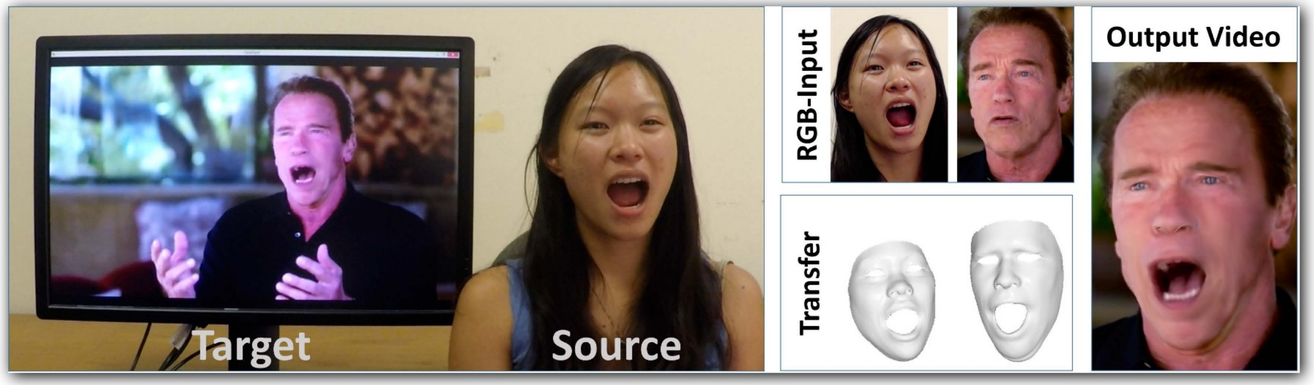
4. Generative data is getting pretty realistic...

Example of Realistic Synthetic Photographs Generated with BigGANTaken from Large Scale GAN Training for High Fidelity Natural Image Synthesis, 2018.

Face2Face: Real-time Face Capture and Reenactment of RGB Videos

Justus Thies [1] Michael Zollhöfer [2] Marc Stamminger [3] Christian Theobalt [2] Matthias Nießner [1]

[1] Technical University of Munich [2] Max Planck Institute for Informatics [3] University of Erlangen-Nuremberg



Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, June 2016

What are the implications of this for medical imaging?

5. Models will no longer be created in an ad-hoc manner

Model Search

Model search (MS) is a framework that implements AutoML algorithms for model architecture search at scale. It aims to help researchers speed up their exploration process for finding the right model architecture for their classification problems (i.e., DNNs with different types of layers).

6. Joint optimization of hardware and software is proliferating

technology feature

Check for updates

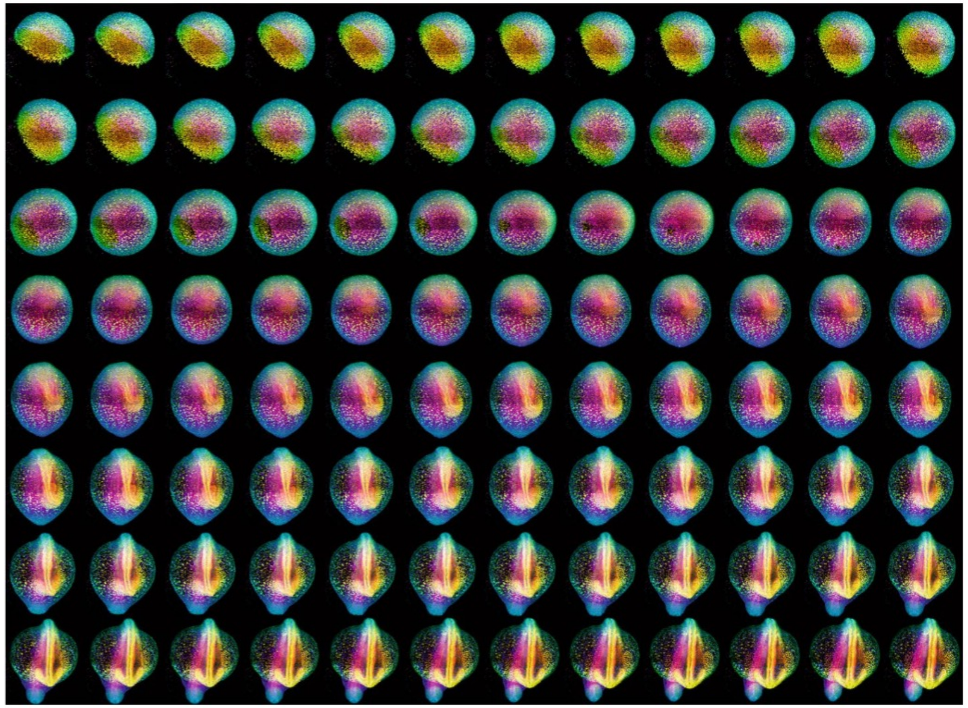
Smart solutions for automated imaging

Algorithms trained to interpret microscope data can greatly extend the information that can be derived from the resulting images, or even optimize how imaging experiments are conducted.

Michael Eisenstein

While buzzing about in search of food, a fruit fly encounters a deadly wasp. Fortunately, its brain reacts to the threat by initiating a cascade of responses across a network of neurons that help it to flee. Philipp Keller's group at the Howard Hughes Medical Institute's Janelia Research Campus has developed a variety of sophisticated strategies for deconvolving the circuitry underlying this and other complex functions of the *Drosophila* nervous system, using a combination of optogenetic manipulation and cutting-edge light-sheet microscopy to simulate various stimuli in living tissue and analyze the response. But perhaps the most remarkable aspect of this project is the extent to which the instruments themselves are running the show. "The microscope can basically do these experiments completely on its own," says Keller.

This work is a particularly advanced example of an emerging field of computer-assisted imaging known as 'smart microscopy'. In these configurations, the



deep imaging

Take-aways for the future of machine learning and imaging

1. It’s not going away....it works, there’s a big community, and lots of funding

2. Hardware *and* software are rapidly evolving

3. CNN’s work very well, but they are not the final solution...Transformer models and Diffusion networks are using CNN’s as building blocks

4. There is currently a lack of safeguards and not enough consideration for how to ensure processed results are accurate, secure and trustworthy

5. Merger of hardware and software for key applications is inevitable…

deep imaging

Ethical questions surrounding deep convolutional networks

1. What are your expectations for an image reconstruction algorithm used in a clinical setting?

2. What types of “guarantees” should we be able to make, if any, to a patient?

3. How should we guide future development of ML software to meet any guarantees?

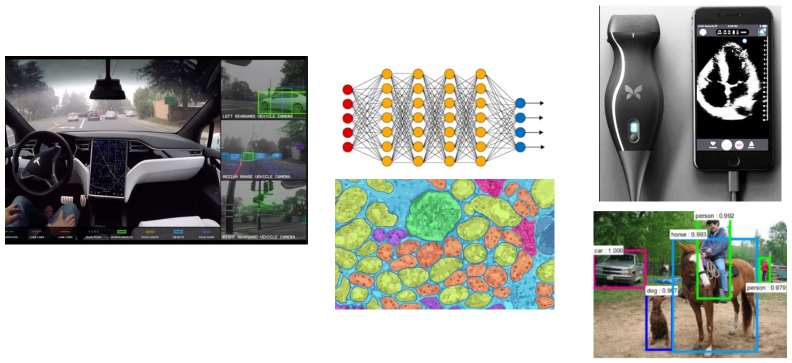
4. How should we guide future development of ML-designed hardware to meet any guarantees?

5. Thoughts towards a system of checks and balances?

What was this class about?

Where your final project is aimed

Computer-centered hardware + software



Human-centered hardware design

Computer-centered software design

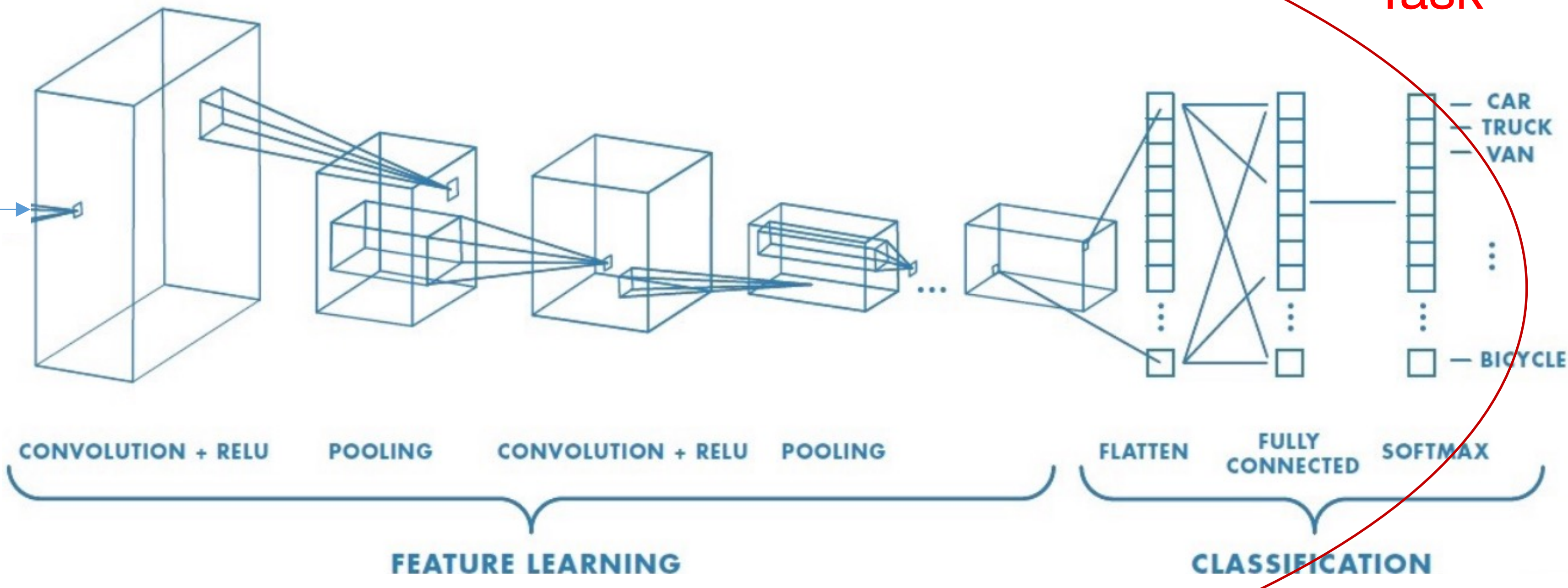
Bringing together physical and digital image representations

Physical world I_0

Hardware

Image I_1

Task

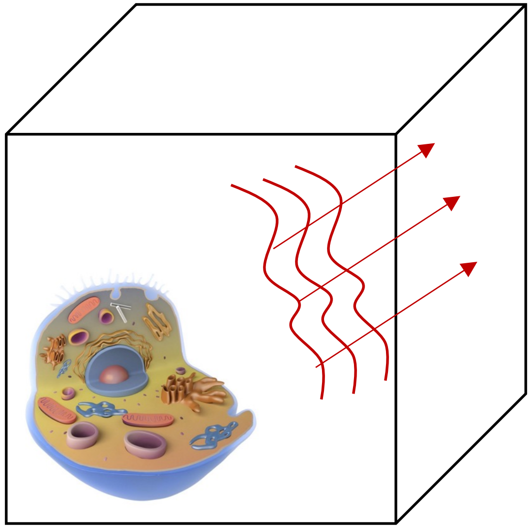


Final project: try to optimize all of this together!

ML+Imaging pipeline introduction

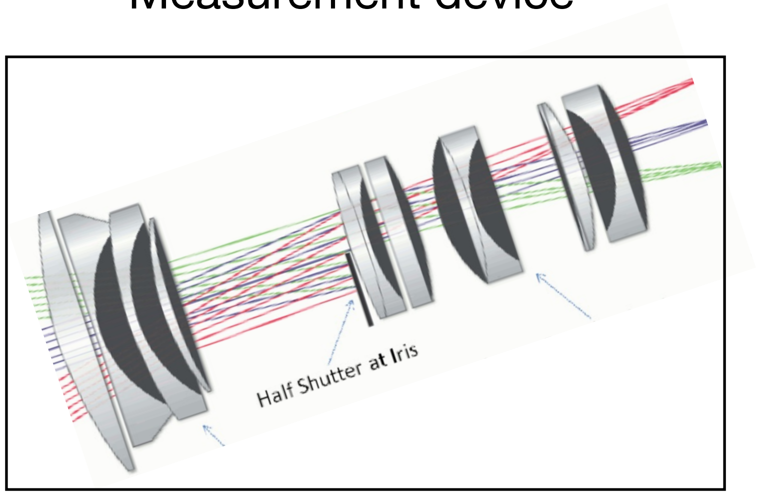
deep imaging

Real World



Continuous
complex fields

Measurement device

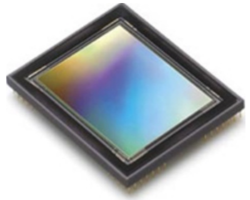


Black box transformations

- Convolution
- Fourier Transform

Digitization

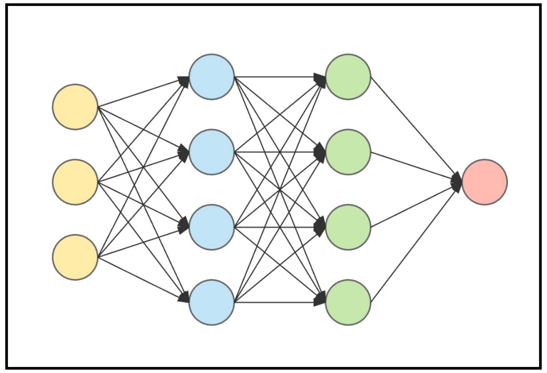
γ -> e^-



Sampling Theorem

Discrete math &
Linear algebra

Machine Learning



Optimization

Linear classification

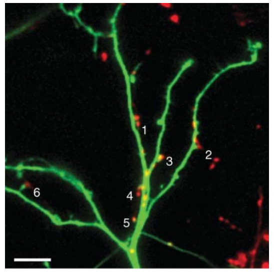
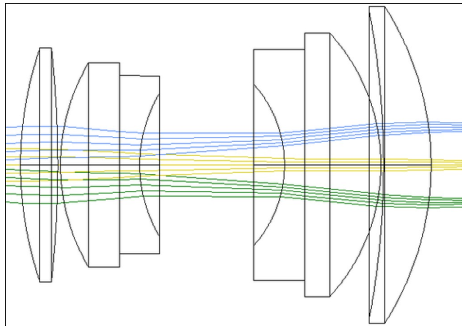
Logistic classifier

Neural networks

Convolutional NN's

Physical models for light propagation to sensor

- **Interpretation #1: Radiation (*Incoherent*)**
- Model: Rays

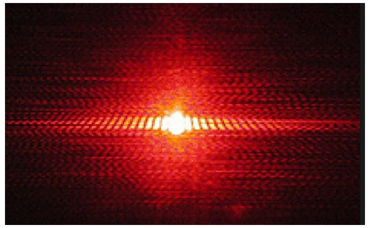




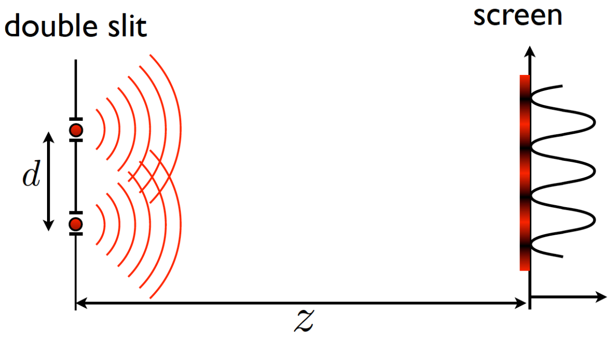
- Real, non-negative

$$\boxed{I_s = H\,B\,S_0}$$

- Sample absorption **S**
- Illumination brightness **B**
- Blur in **H**

- **Interpretation #2: Electromagnetic wave (*Coherent*)**
- Model: Waves





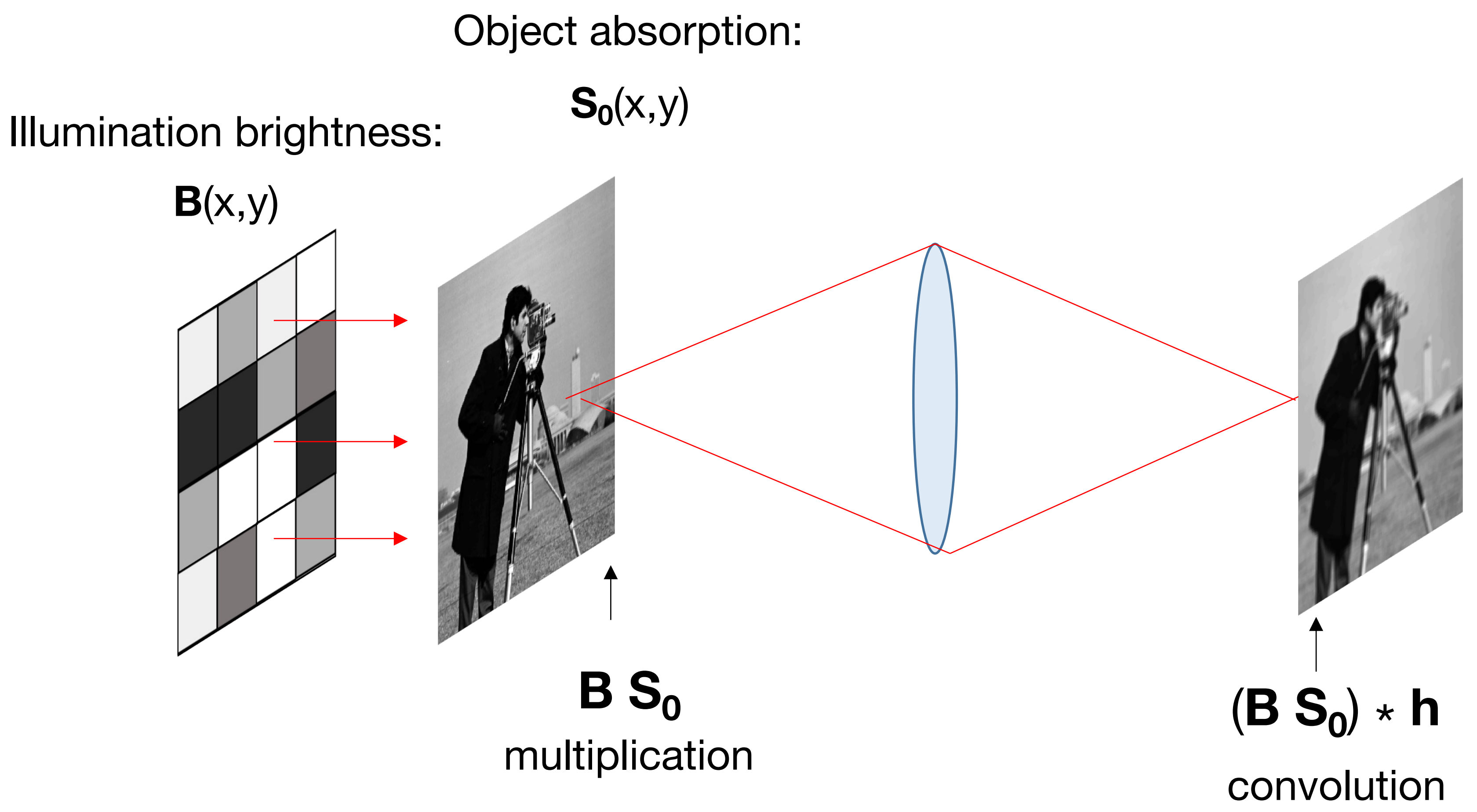
- Complex field

$$\boxed{I_C = |H\,C\,S_C|^2}$$

- Sample abs./phase **S**
- Illumination wave **B**
- Blur in **H**

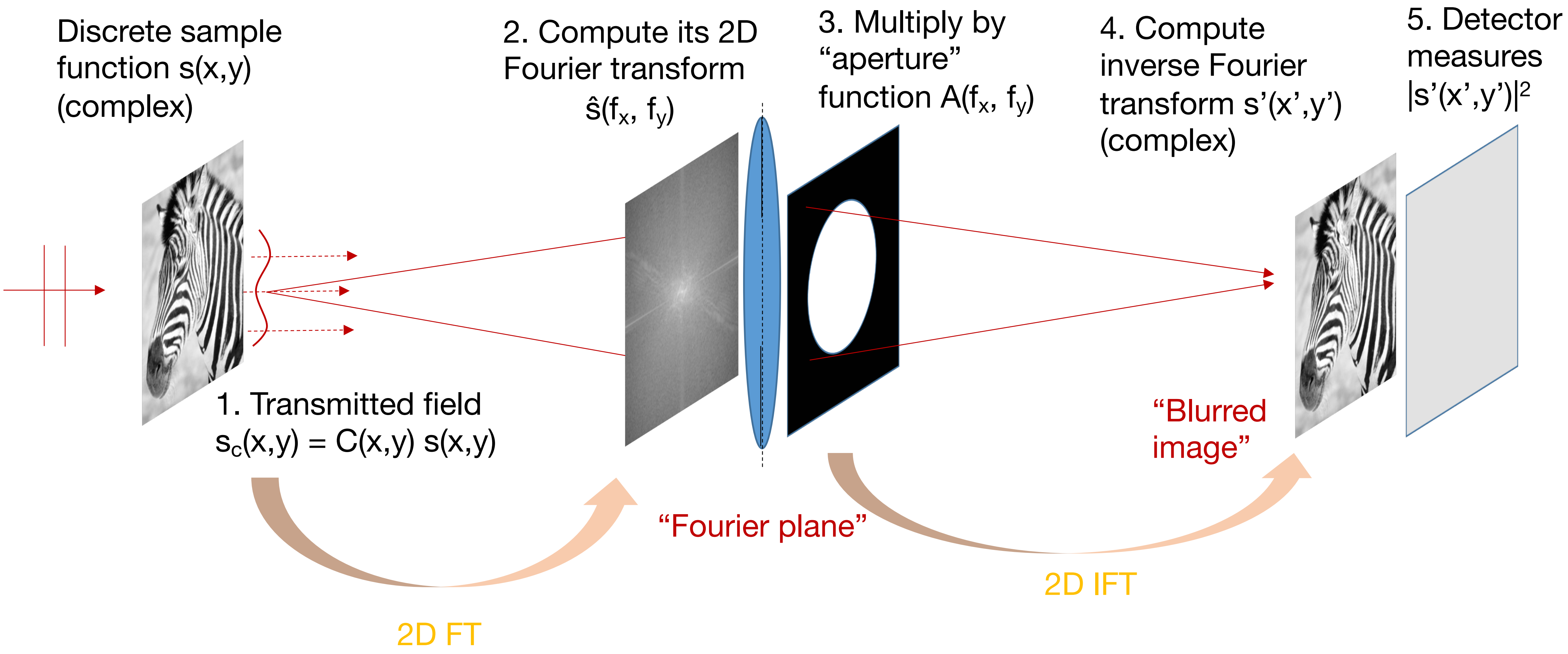
Mathematical model of for incoherent image formation

- All quantities are real, and non-negative



deep imaging

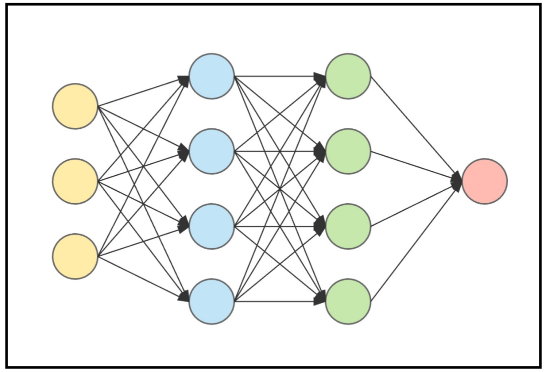
Model of image formation for wave optics (coherent light):

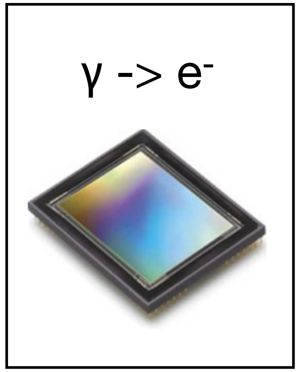


Machine Learning

deep imaging

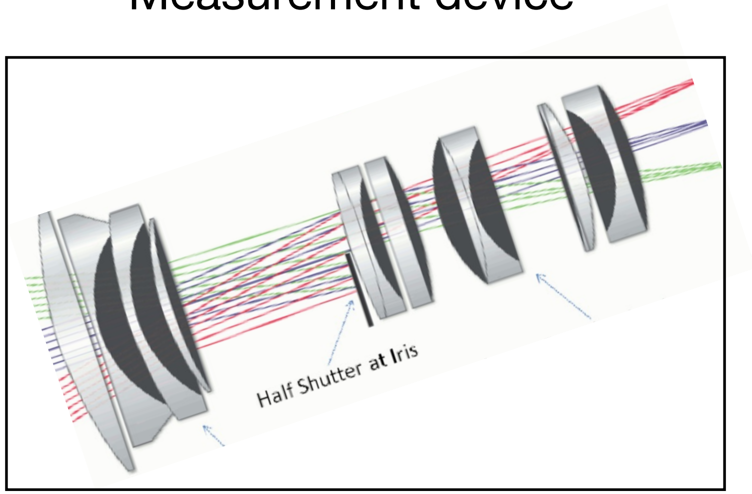
Digitization

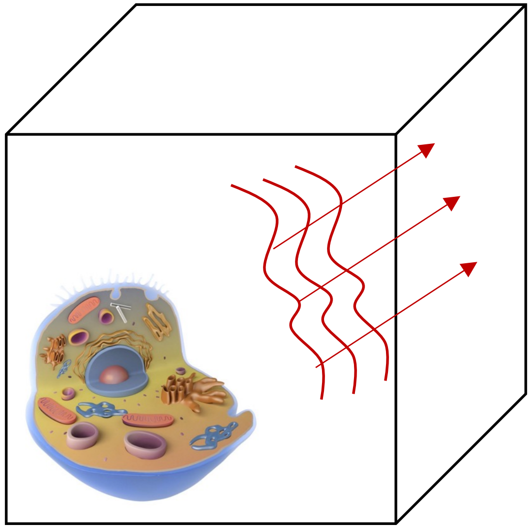




Measurement device

Real World





Optimization

Linear classification

Logistic classifier

Neural networks

Convolutional NN’s

Sampling Theorem

Discrete math & Linear algebra

Continuous complex fields

Black box transformations

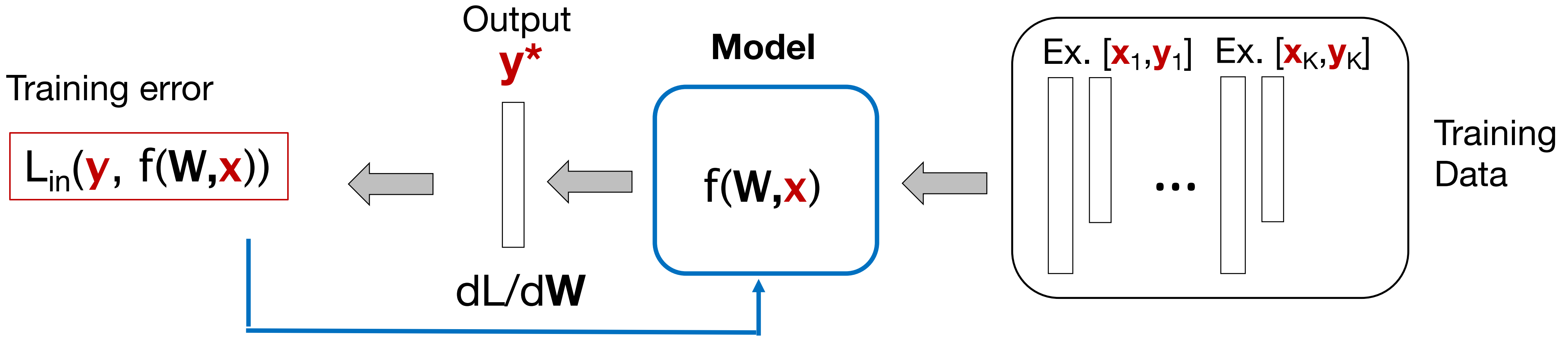
- Convolution
- Fourier Transform

$$\boxed{I_{Inc} = H\,B\,S_0}$$

$$\boxed{I_{Coh} = |H\,C\,S_C|^2}$$

Summary of machine learning pipeline:

1. Network Training

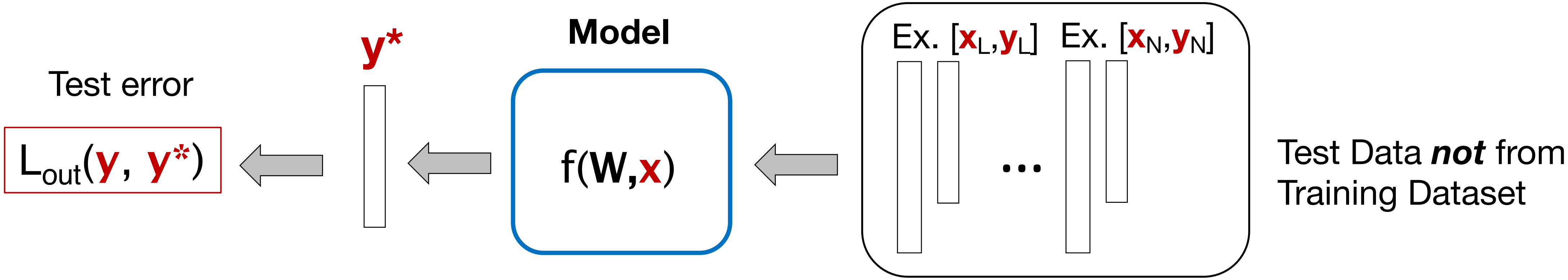


What we need for network training:

1. **Labeled examples**
2. **A model and loss function**
3. **A way to minimize the loss function L**

Summary of machine learning pipeline:

2. Network Testing



What we need for network testing:

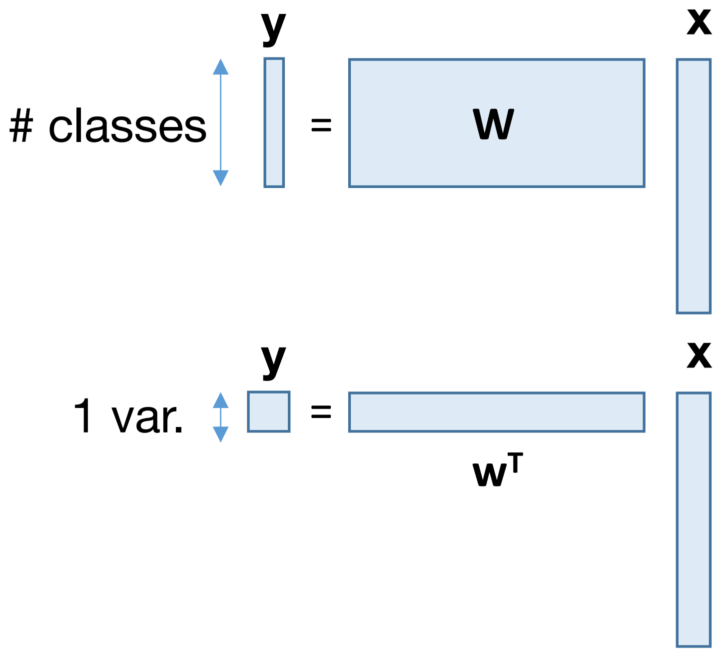
4. *Unique* labeled test data
5. Evaluation of model error

Let's start with a simpler approach: linear regression

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

General linear model:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(\mathbf{W}\mathbf{x}_i, y_i)$$



Assume 1 class =
1 linear fit

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(w^T x_i - y_i)$$

Use MSE error
model

$$L = \frac{1}{N}\sum_{i=1}^{N} (w^T x_i - y_i)^2$$

Where labels
determined by
thresholding

$$f(\mathbf{x}_i) = y_i^* = \operatorname{sgn}(\mathbf{w}^T\mathbf{x}_i)$$

$$\operatorname{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Let’s start with a simpler approach: linear regression

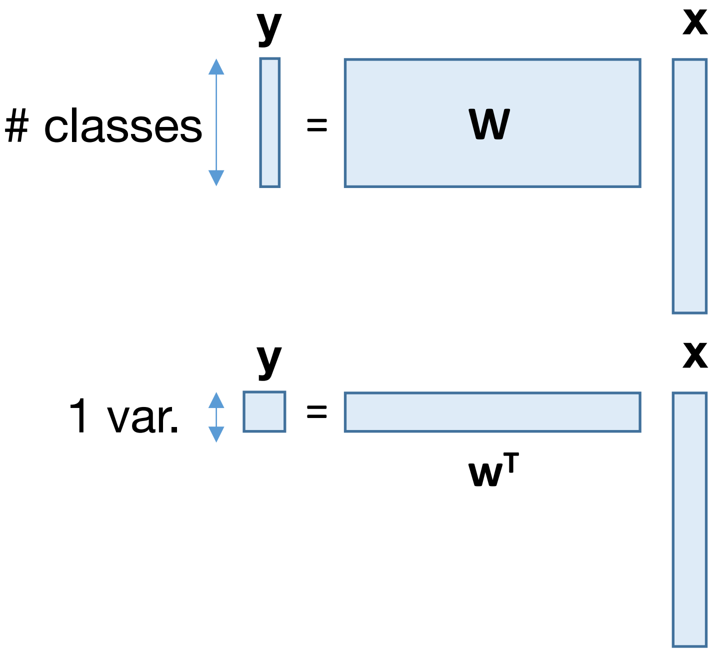
$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, \mathbf{W}), y_i)$$

General linear model:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(\mathbf{W}\mathbf{x}_i, y_i)$$

Assume 1 class = 1 linear fit

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(w^T x_i - y_i)$$



Use MSE error model

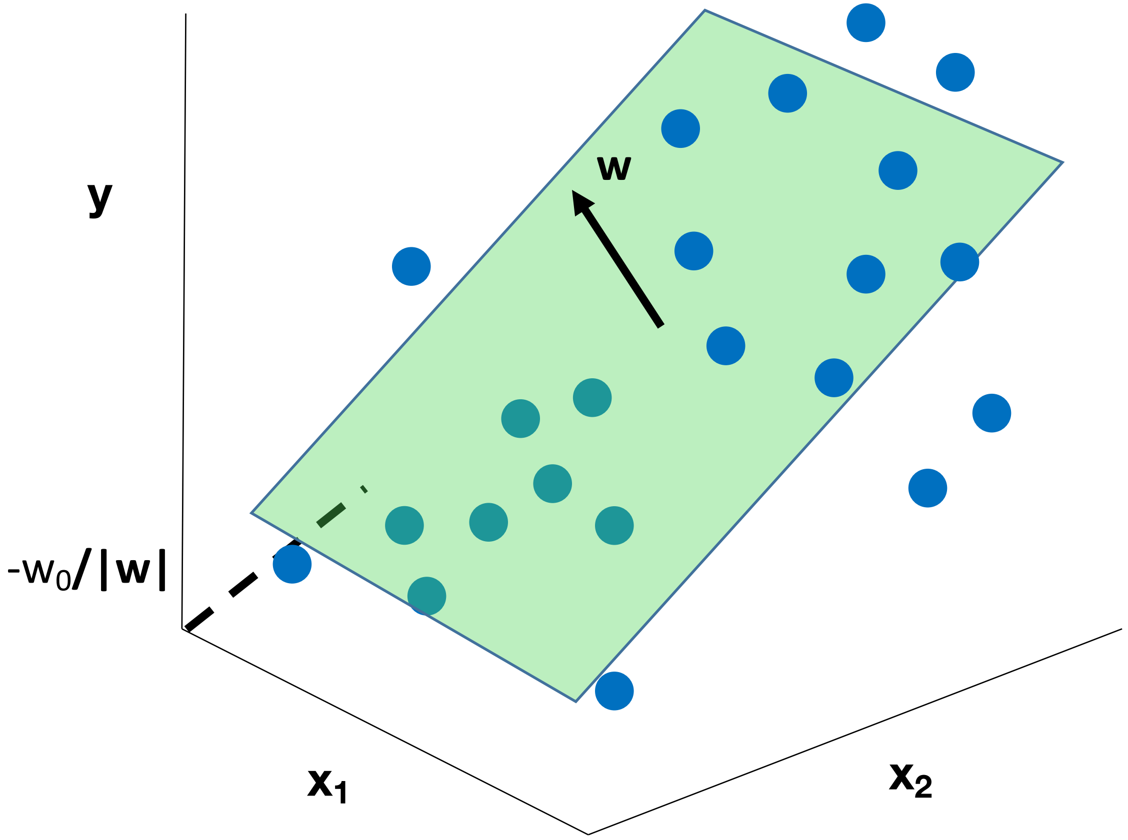
$$L = \frac{1}{N}\sum_{i=1}^{N} (w^T x_i - y_i)^2$$

Where labels determined by thresholding

$$f(\mathbf{x}_i) = y_i^* = \mathrm{sgn}(\mathbf{w}^T \mathbf{x}_i)$$

$$\mathrm{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Why does linear regression with sgn() achieve classification?



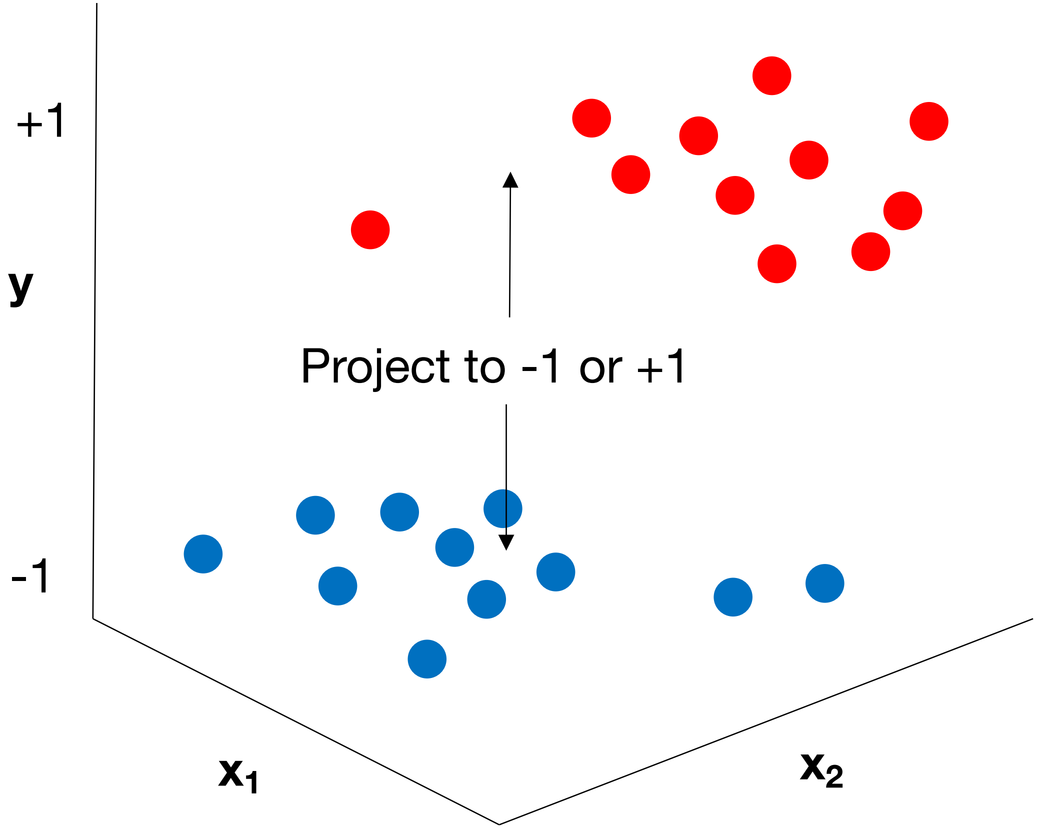
Without sgn(): regression for best fit

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i$$

$$L = \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2$$

- If y_i can be anything, minimizing L makes **w** the plane of best fit

Why does linear regression with sgn() achieve classification?



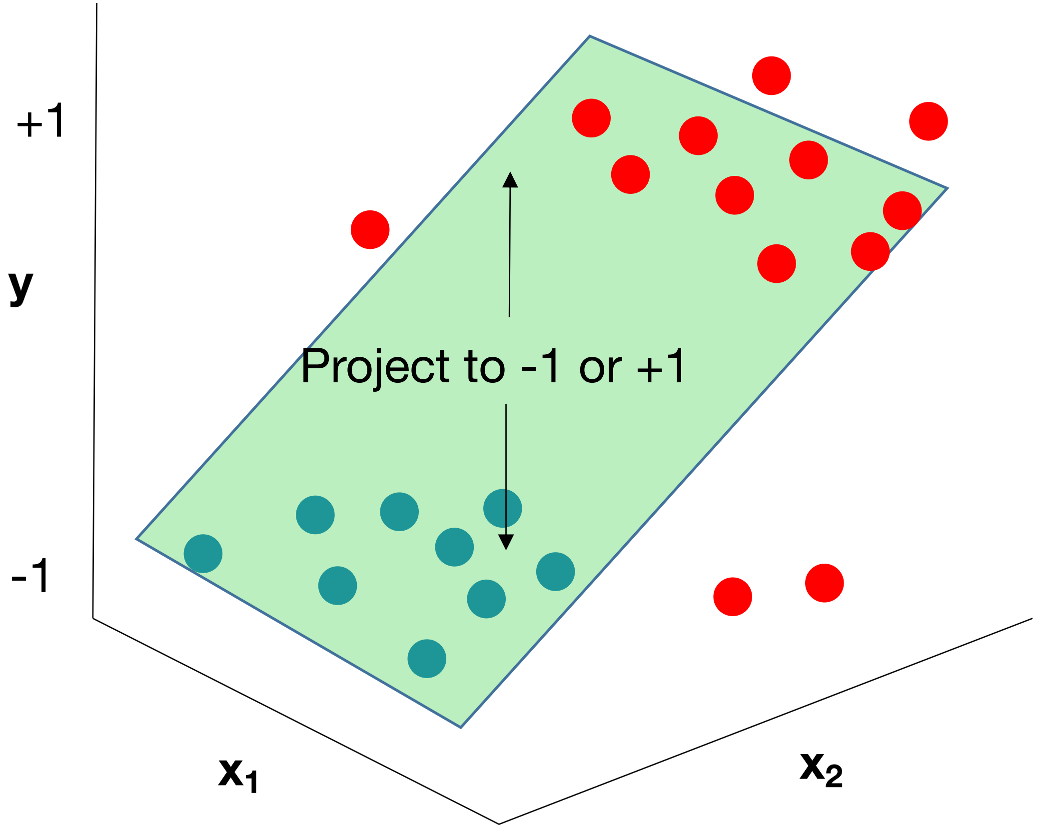
Without sgn(): regression for best fit

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i$$

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2$$

- y_i can only be -1 or +1, which defines its class

Why does linear regression with sgn() achieve classification?



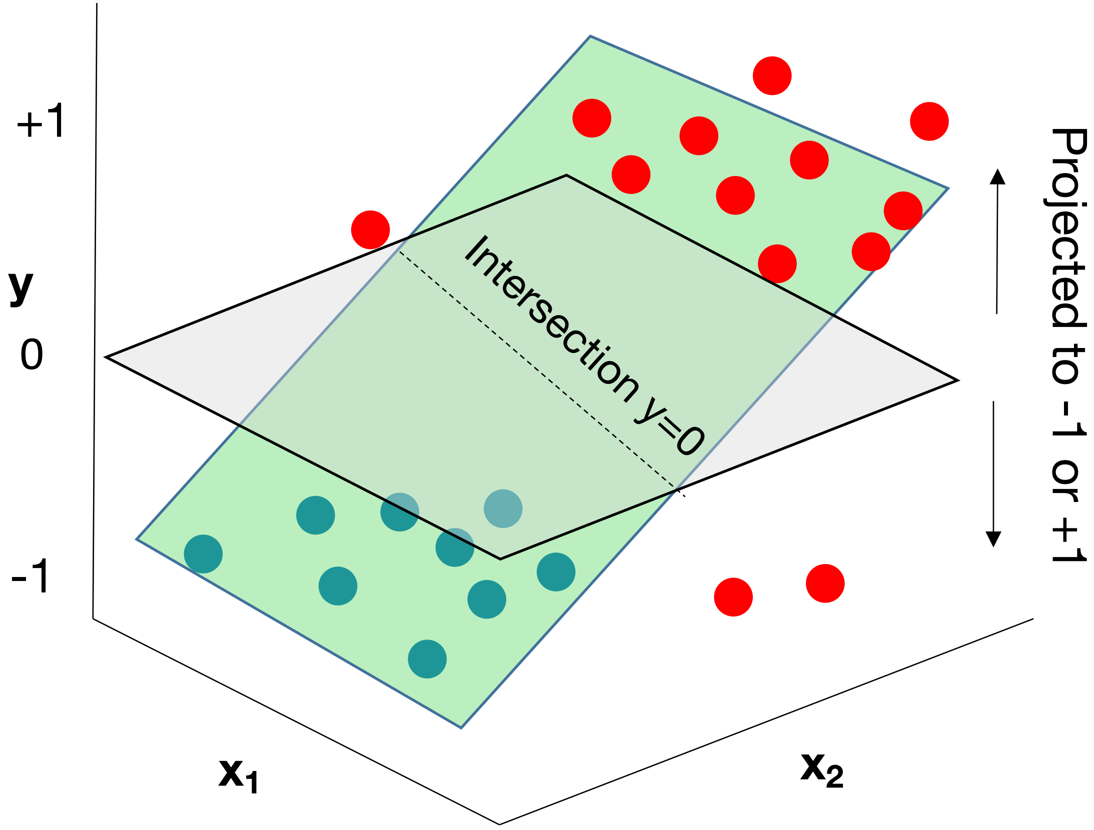
Without sgn(): regression for best fit

$$f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i$$

$$L = \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2$$

- y_i can only be -1 or +1, which defines its class
- Can still find plane of best fit

Why does linear regression with sgn() achieve classification?



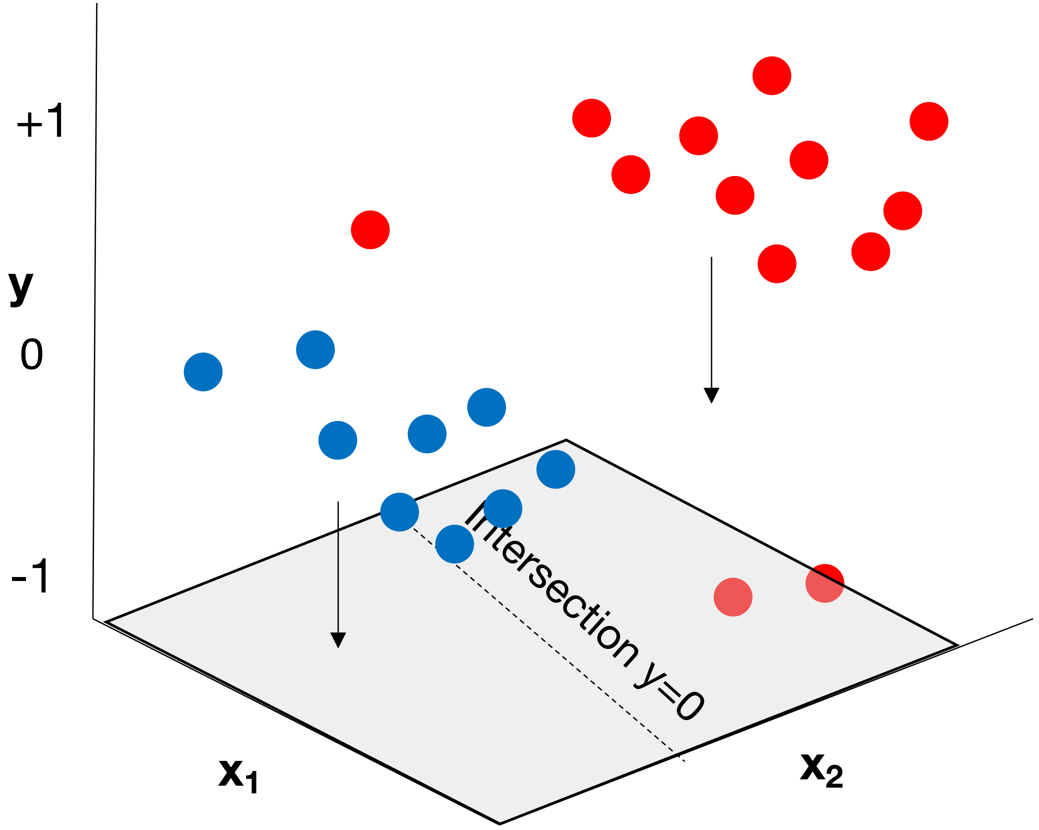
With sgn() operation:

$$f(\mathbf{x}_i) = y_i^* = \mathrm{sgn}(\mathbf{w}^T \mathbf{x}_i)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2$$

- Anything point to one side of y=0 intersection is class +1, anything on the other side of intersection is class -1

Why does linear regression with sgn() achieve classification?



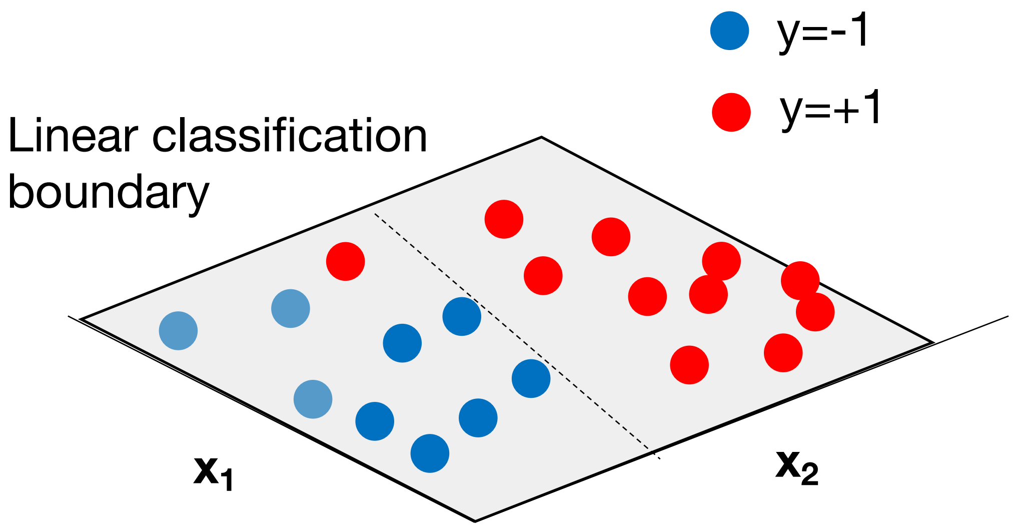
With sgn() operation:

$$f(\mathbf{x}_i) = y_i^* = \mathrm{sgn}(\mathbf{w}^T \mathbf{x}_i)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2$$

- y axis isn't really needed now & can view this decision boundary in 2D

Why does linear regression with sgn() achieve classification?



With sgn() operation:

$$f(\mathbf{x}_i) = y_i^* = \mathrm{sgn}(\mathbf{w}^T \mathbf{x}_i)$$

$$L = \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2$$

Sign operation takes linear regression and makes it a classification operation!

How to minimize L_{in}

For logistic regression,

$$L_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^{\mathsf{T}} \mathbf{x}_n}\right) \quad \longleftarrow \text{ **iterative** solution}$$

Compare to linear regression:

$$L_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{w}^{\mathsf{T}} \mathbf{x}_n - y_n)^2 \quad \longleftarrow \text{ closed-form solution}$$

In the rest of this class: solve via gradient descent

With a matrix, compute this for each entry:

$$\frac{dL(W_i)}{dW_i} = \lim_{h \to 0} \frac{L(W_i + h) - L(W_i)}{h}$$

Example:

W = [1,2;3,4]
L(W, x, y) = 12.79

W_1+h = [1.001,2;3,4]
L(W_1+h, x, y) = 12.8

d$L(W_1)/dW_1$ = 12.8-12.79/.001

$\boxed{dL(W_1)/dW_1 = 10}$

- Repeat for all entries of **W**, dL/d**W** will have NxM entries for NxM matrix

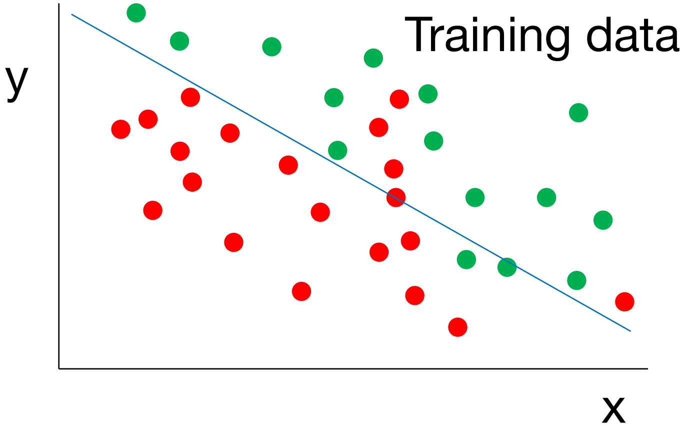
Steepest descent and the best step size ε

1. Evaluate function f($\mathbf{x}^{(0)}$) at an initial guess point, $\mathbf{x}^{(0)}$
2. Compute gradient $\mathbf{g}^{(0)} = \nabla_x f(\mathbf{x}^{(0)})$
3. Next point $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \varepsilon^{(0)}\mathbf{g}^{(0)}$
4. Repeat – $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \varepsilon^{(n)}\mathbf{g}^{(n)}$, until $|\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}| <$ threshold t

$$L = \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2$$

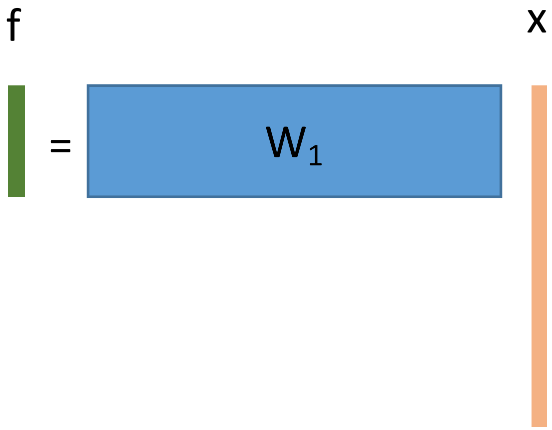
$$\nabla L(w) = \frac{2}{N}X^T(Xw - y) = 0$$

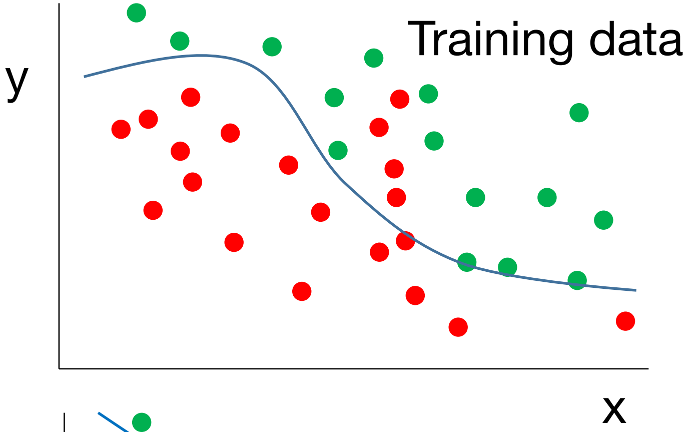
```
while previous_step_size > precision and iters < max_iters:
    prev_x = cur_x
    cur_x -= gamma * df(prev_x)
    previous_step_size = abs(cur_x - prev_x)
    iters+=1
```



$$f = W_1 x$$

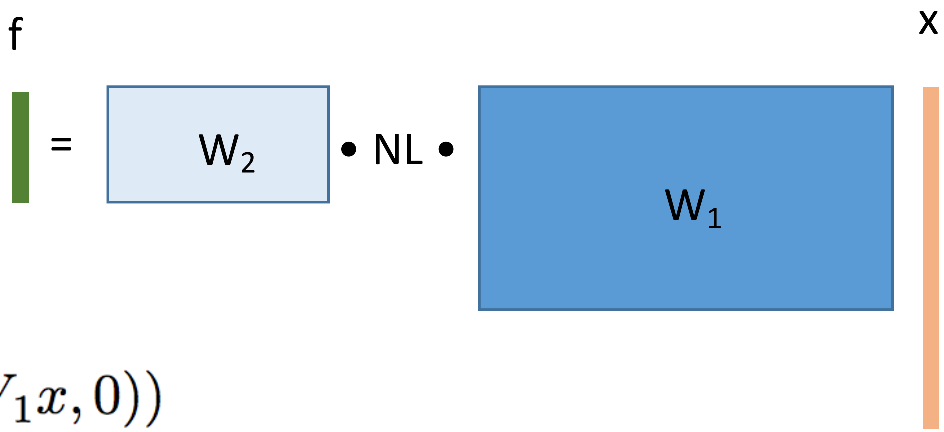
Learned *f:* not flexible

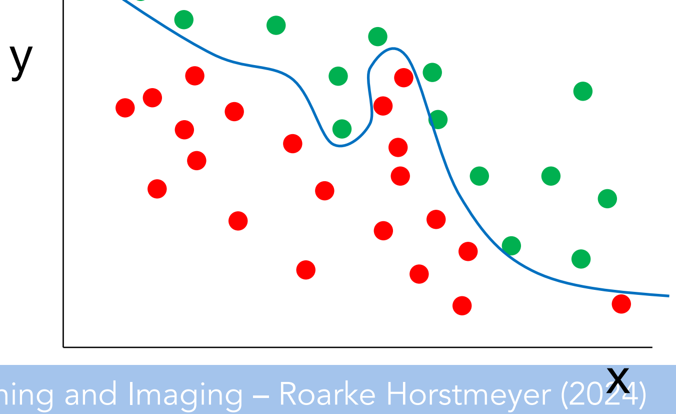




$$f = W_2 \max(W_1 x, 0)$$

Learned *f:* a bit flexible





$$f = W_3 \max(0, W_2 \max(W_1 x, 0))$$

Learned *f:* more flexible

Does it generalize???

We can keep adding these “layers”…

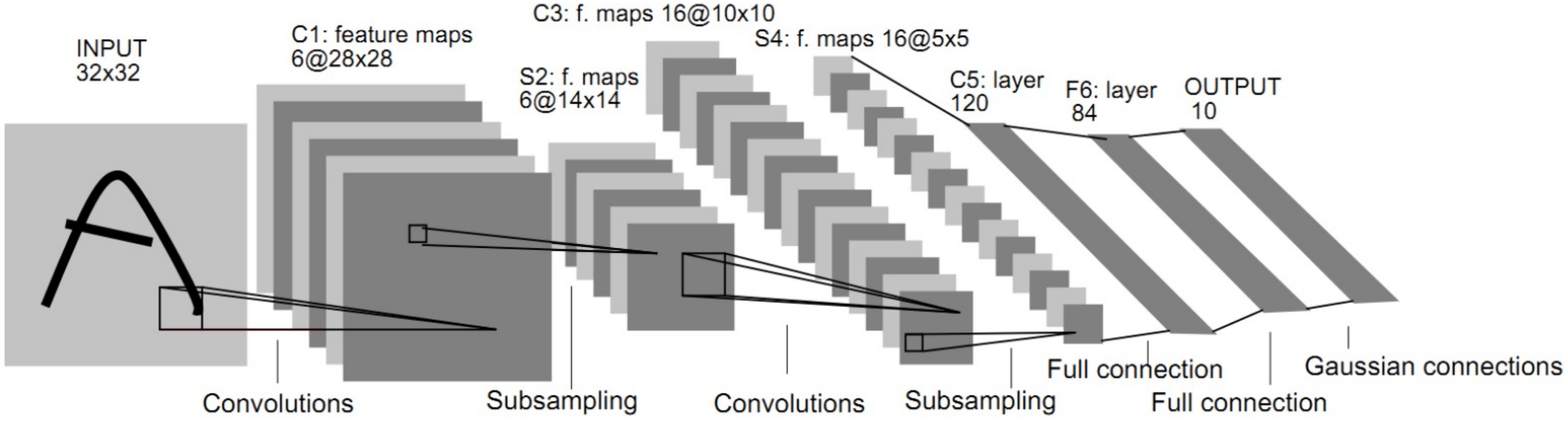
Before CNN's – understand two competing goals in machine learning

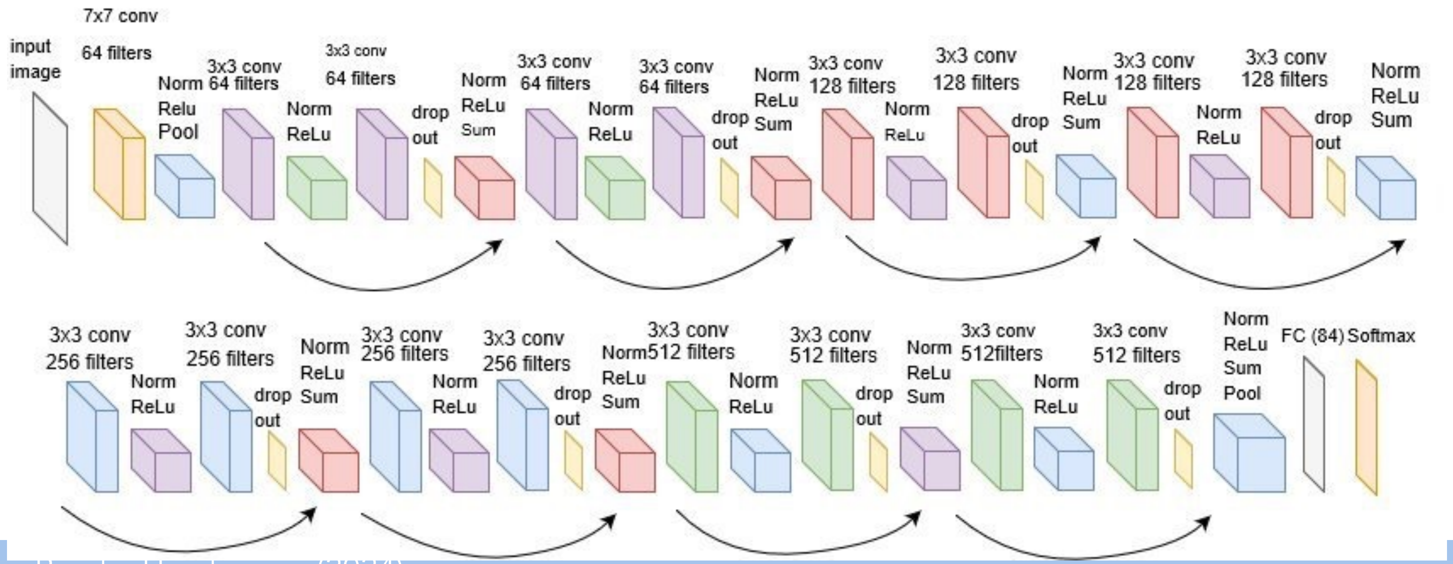
deep imaging

1. Can we make sure the in-sample error $L_{in}(y, f(x,W))$ is small enough?
   - Appropriate cost function
   - "complex enough" model

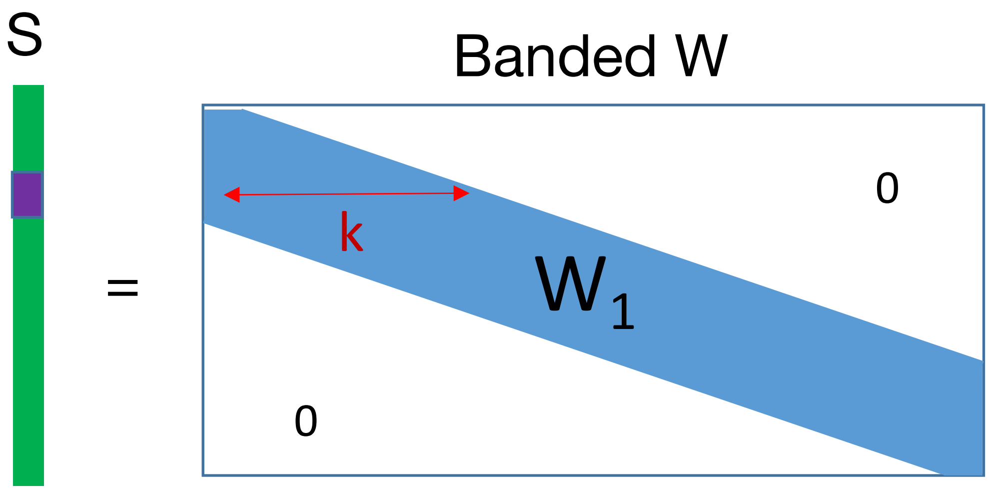
2. Can we make sure that $L_{out}(y, f(x,W))$ is close enough to $L_{in}(y, f(x,W))$?
   - Probabilistic analysis says yes!
   - $|L_{in} - L_{out}|$ bounded from above
   - Bound grows with model capacity (bad)
   - Bound shrinks with # of training examples (good)

Gets us to Convolutional Neural Networks

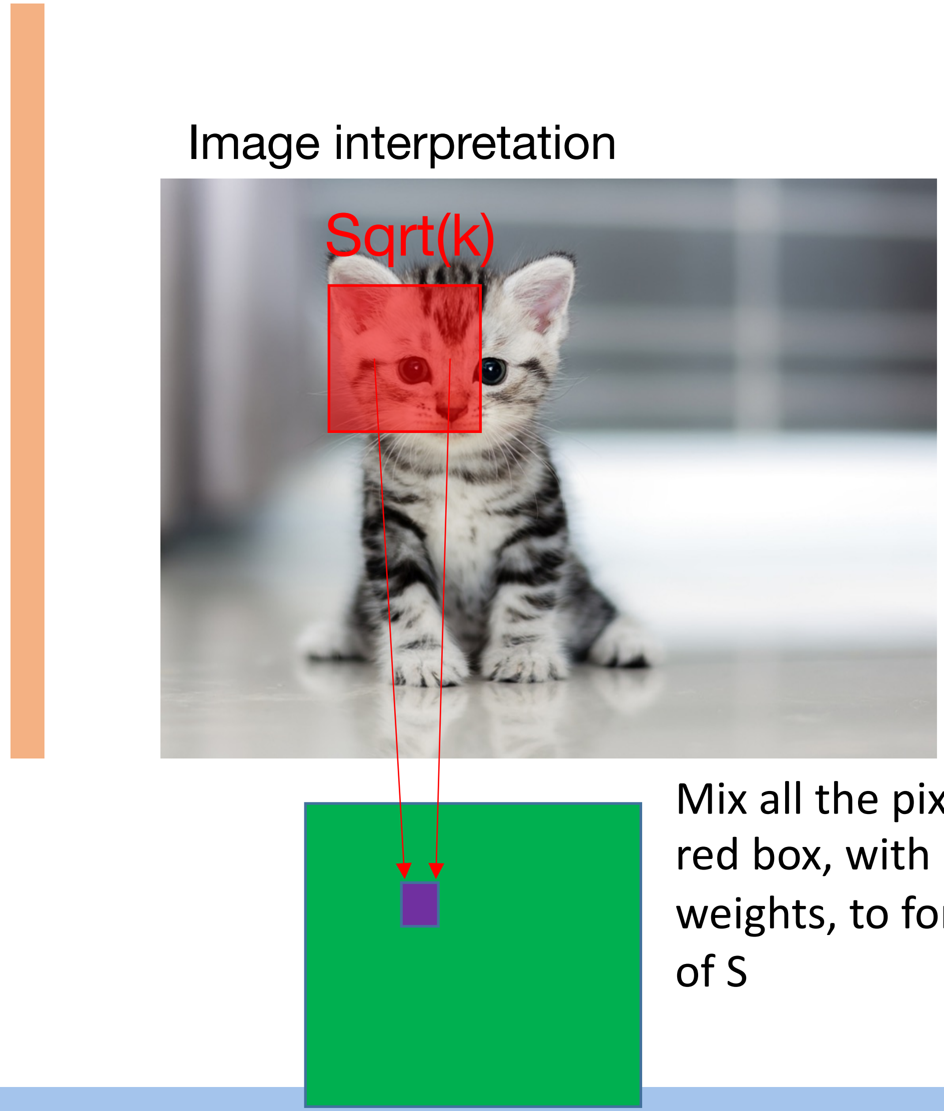




Why should we use these convolutions?



x = cat image



This type of matrix can dramatically reduce the number of weights that are used while still allowing *local* regions to mix:

Full matrix: $O(n^2)$

Banded matrix: k•O(n)

Mix all the pixels in the red box, with associated weights, to form this entry of S

Important components of a CNN

CNN Architecture

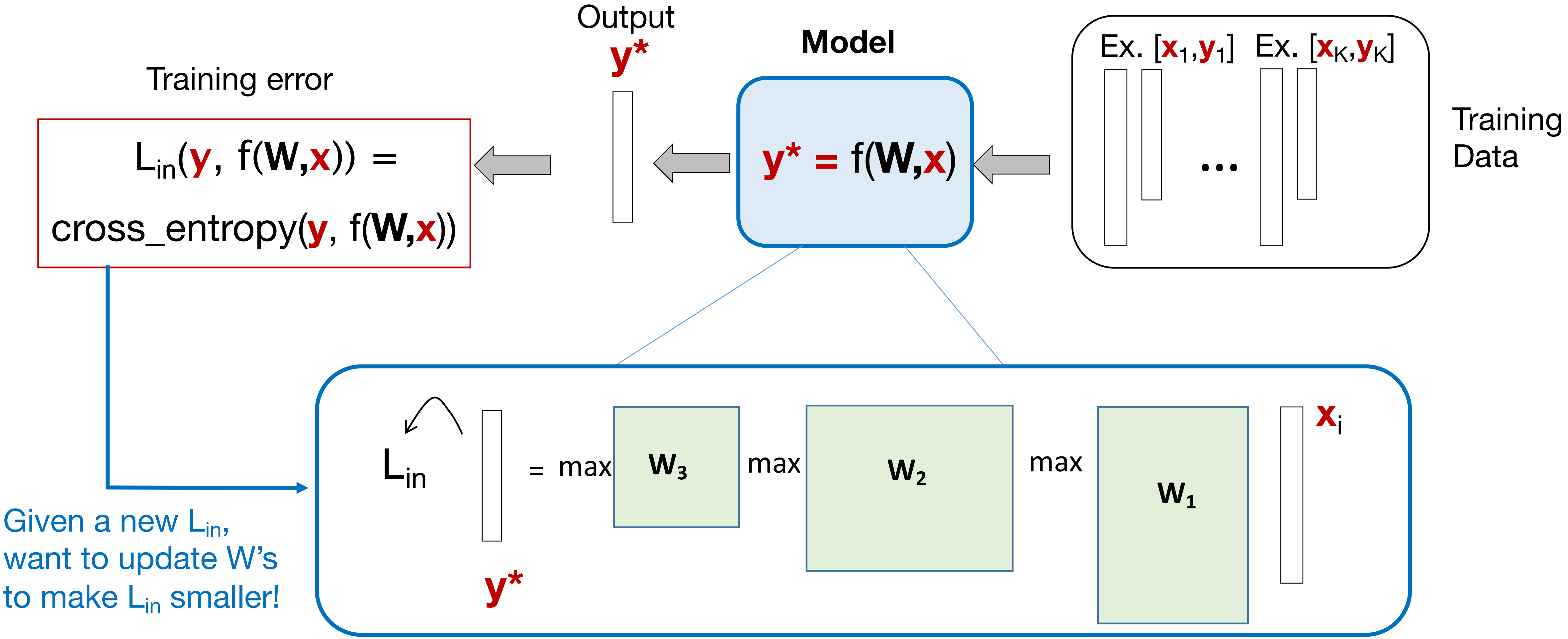
- CONV size, stride, pad, depth
- ReLU & other nonlinearities
- POOL methods
- # of layers, dimensions per layer
- Fully connected layers

Loss function & optimization

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

Other specifics: Pre-processing**,** initialization**,** dropout, batch normalization, augmentation

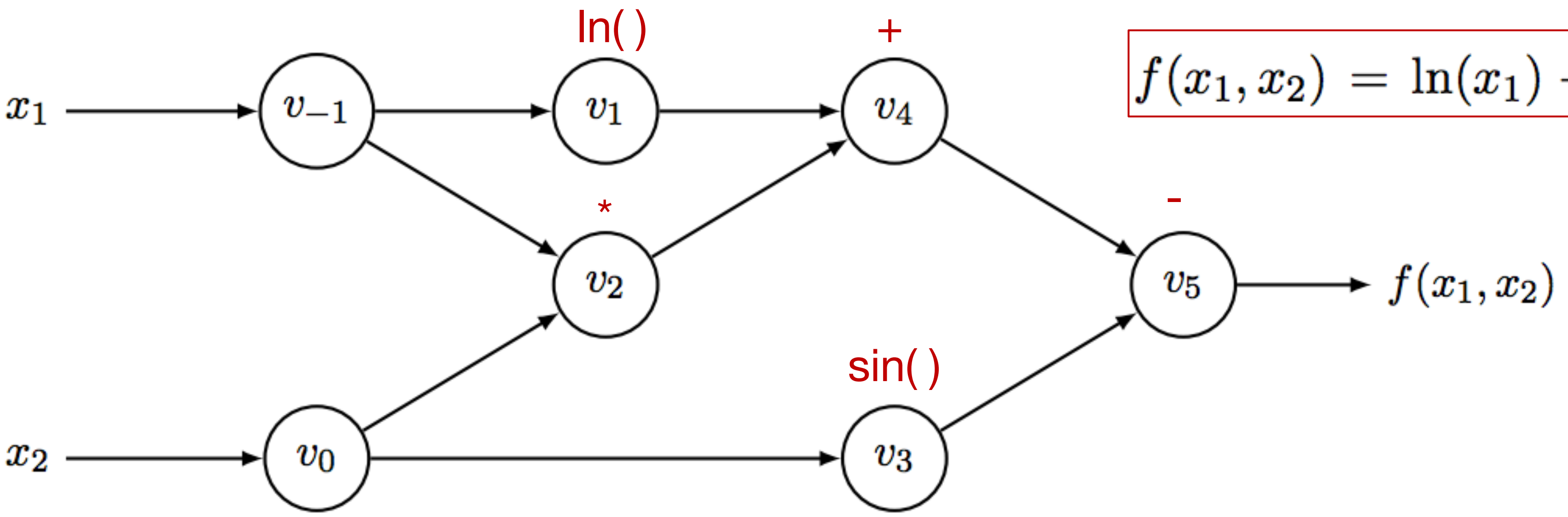
Our very basic convolutional neural network



Backwards pass uses new L_{in} to update **W**'s – backpropagation!

Automatic differentiation on computational graphs

A. Baydin et al., Automatic Differentiation in Machine Learning: a Survey



$$\boxed{f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)}$$

To both determine f and find df/dx_i :

- Create graph of local operations
- Compute analytic (symbolic) gradient at each node (unit) in graph
- Use inter-relationships to establish final desired gradient, df/dx_1
  - Forward differentiation
  - Backwards differentiation = Backpropagation

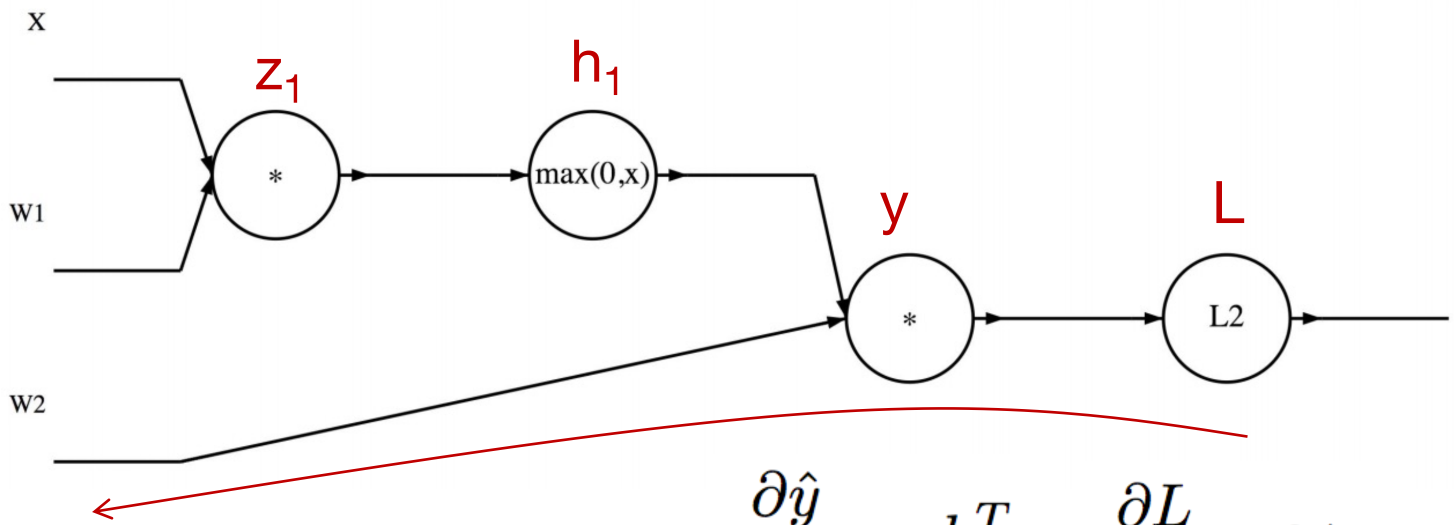
Let's go through an example:

$$L = || \mathbf{W_2}\ \mathrm{ReLU}(\mathbf{W_1}\ \mathbf{X})\ ||_2^2$$

(2-layer network with MSE where we neglect labels **y** for now)

$dL/d\mathbf{W}_1$ = ? $\boxed{dL/d\mathbf{W}_2 = ?}$

$$z_1 = XW_1$$
$$h_1 = \mathrm{ReLU}(z_1)$$
$$\hat{y} = h_1 W_2$$
$$L = ||\hat{y}||_2^2$$



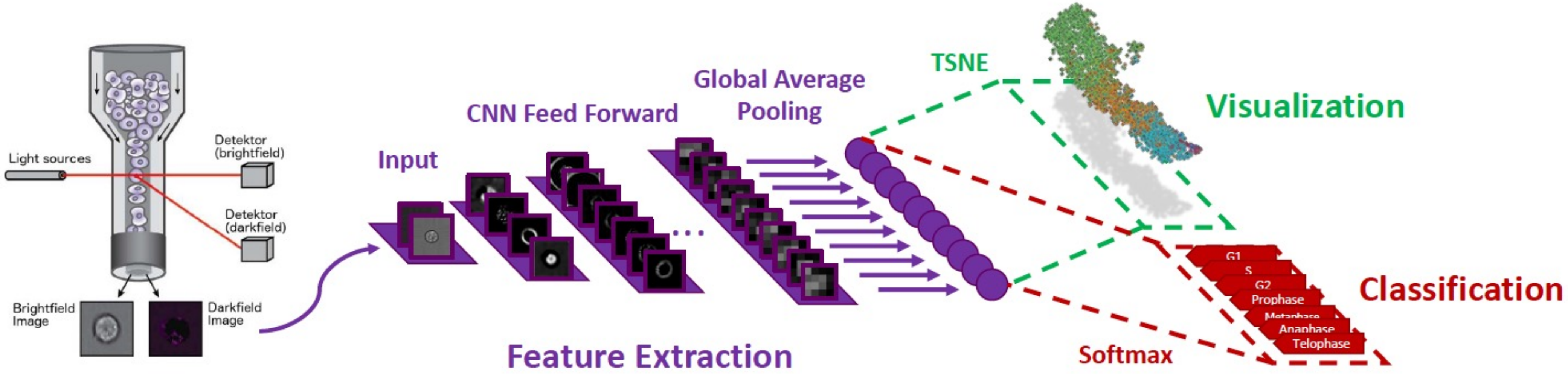
$$\frac{\partial \hat{y}}{\partial W_2} = h_1^T \qquad \frac{\partial L}{\partial \hat{y}} = 2\hat{y}$$

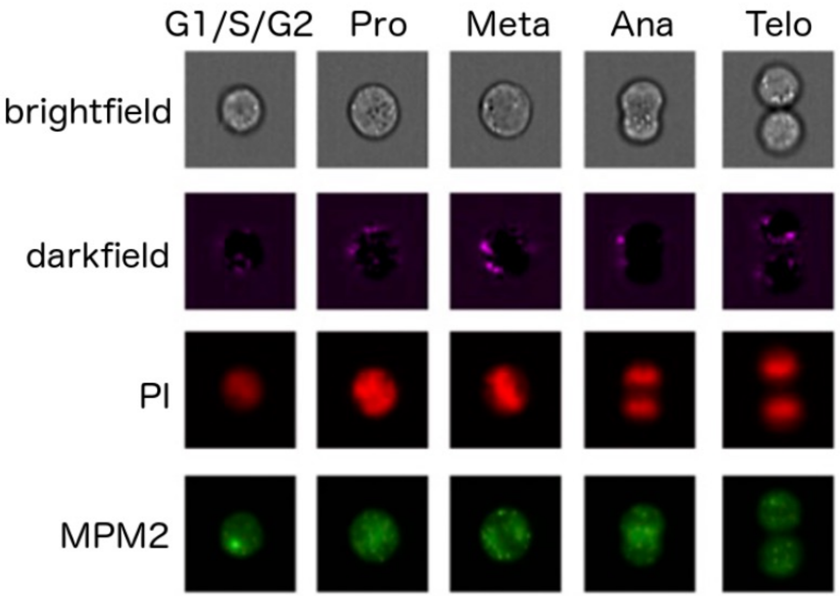
$$\boxed{\frac{\partial L}{\partial W_2} = \frac{\partial \hat{y}}{\partial W_2}\frac{\partial L}{\partial \hat{y}} = 2h_1^T\hat{y}}$$

CNNs for classification

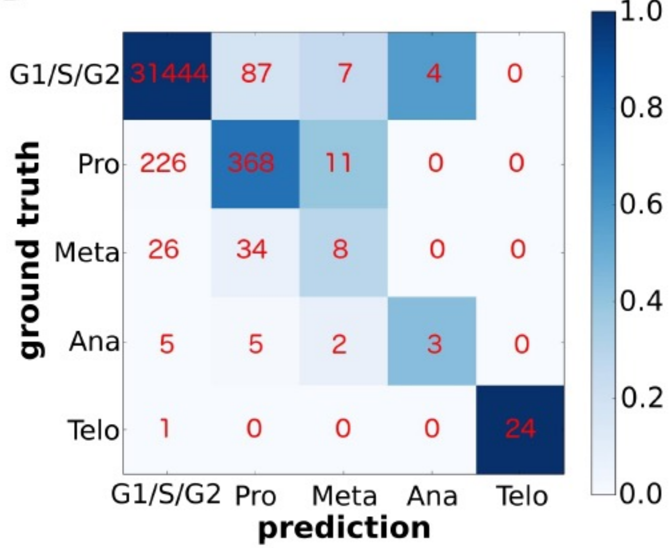
P. Eulenberg et al., “Reconstructing cell cycle and disease progression using deep learning”

deep imaging



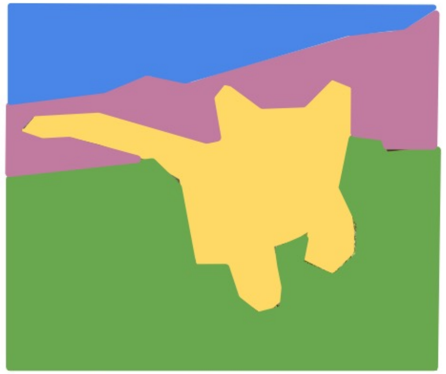


b



Other Computer Vision Tasks

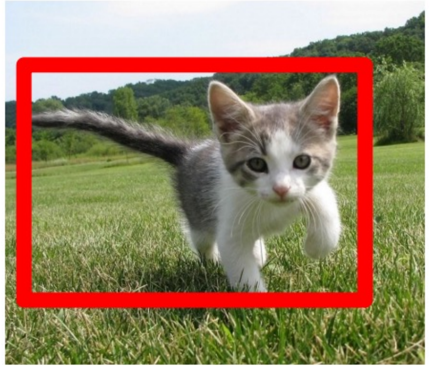
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

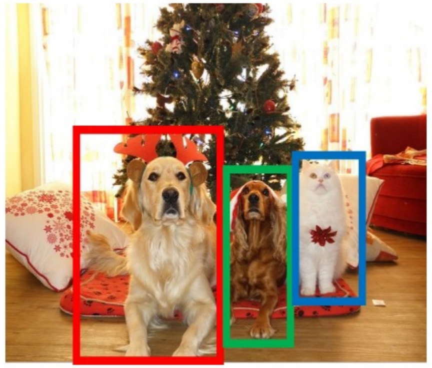
Classification + Localization



CAT

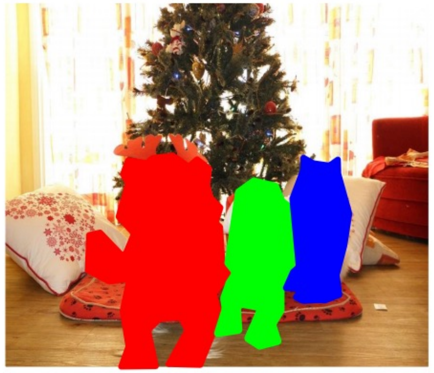
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Super-resolution

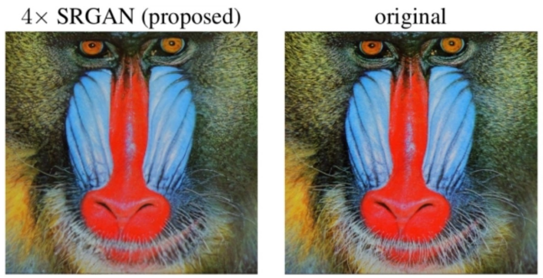


Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

This image is CC0 public domain

Post-processing of your results: a few options at different stages

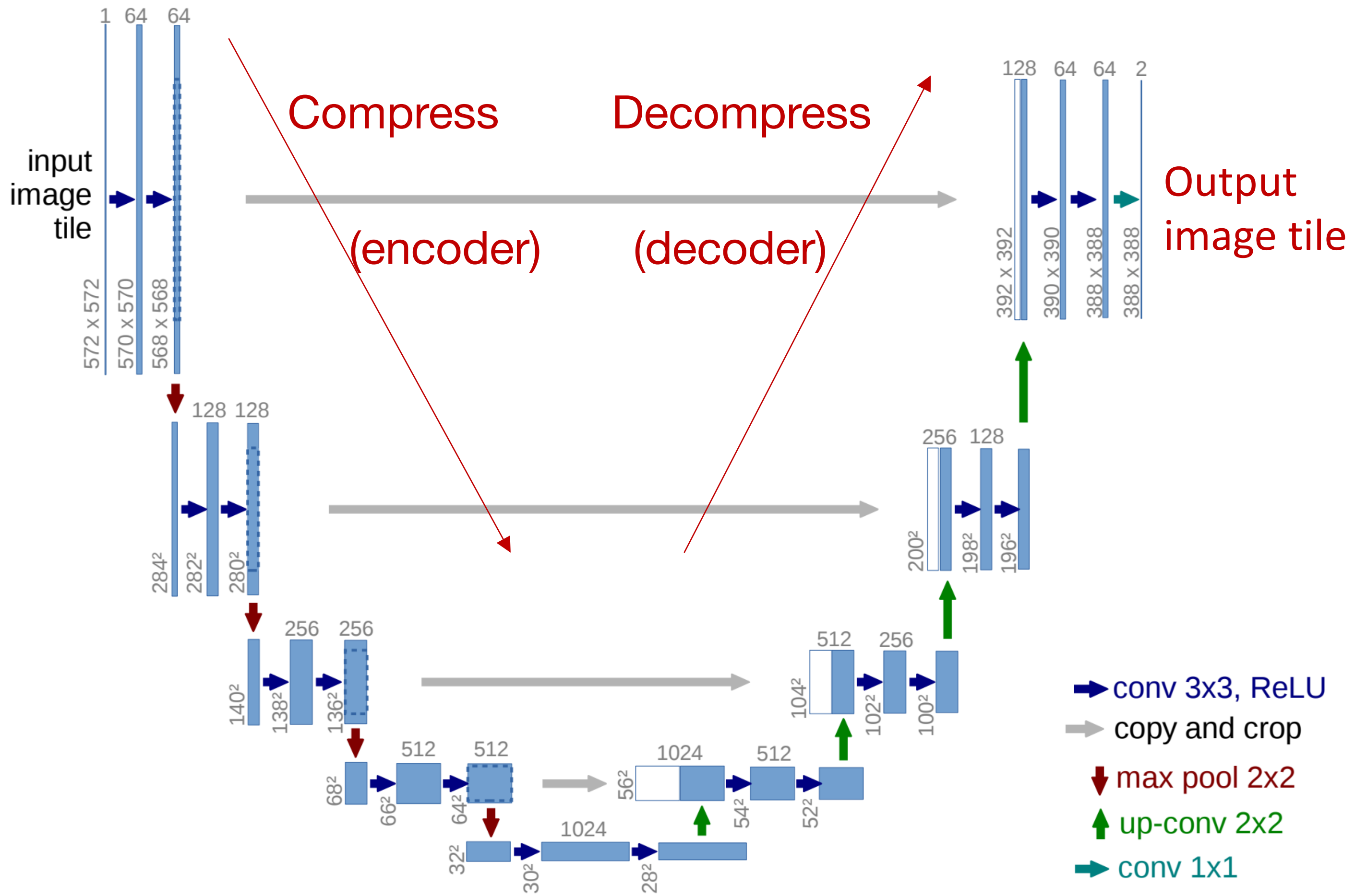
Options to examine your test data after processing:

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

Instead, *compress x-y dimensions of* input image

- Compress spatial features into learned filters
- Then, decompress learned filters back into same spatial dimensions
- **Can be an autoencoder**
- Analogous to image compression
- A very powerful idea…

U-Net Architecture



U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

Bringing together physical and digital image representations

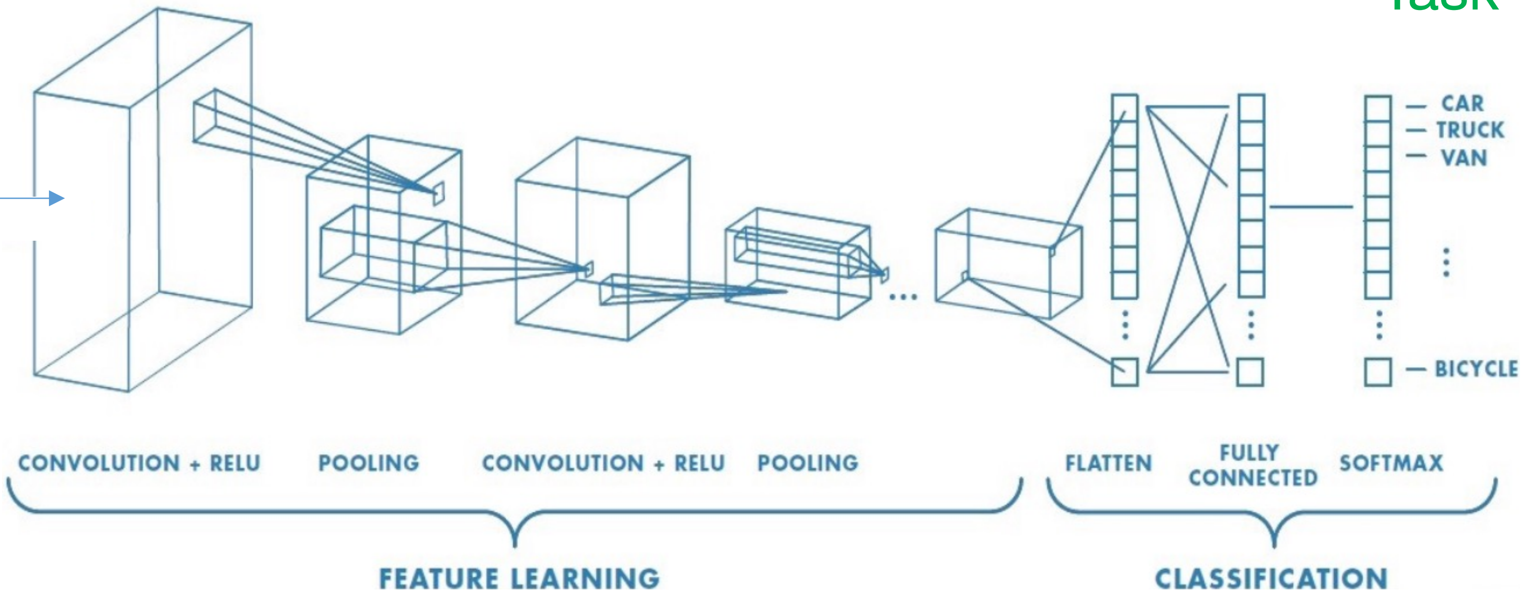
Physical Layers

Digital Layers

Task

Physical Function $\mathbf{I}_0$ → f[ ] → Digitized $\mathbf{I}_s$

$$\mathbf{I}_s = f[\mathbf{I}_0]$$



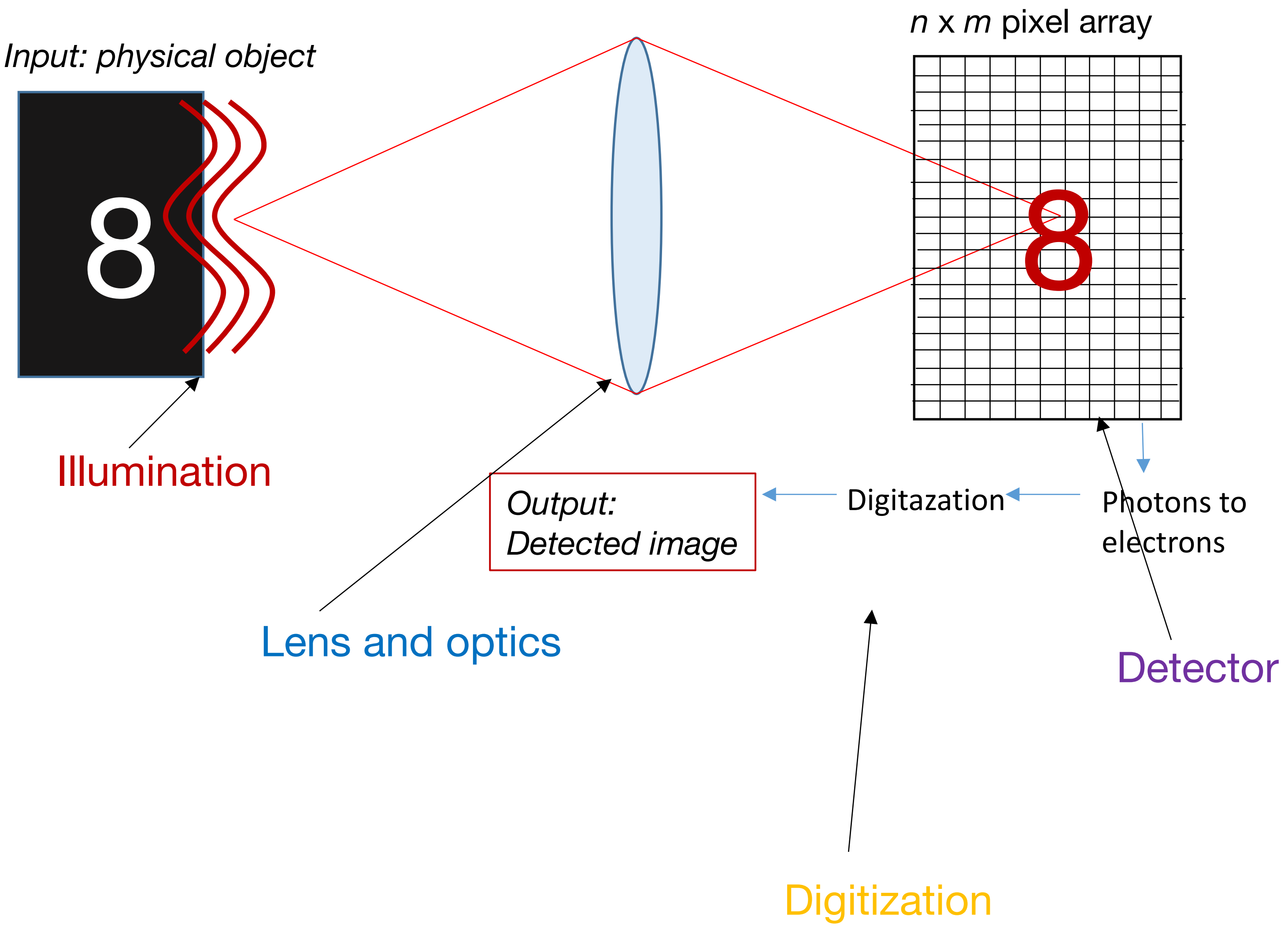
Digital layers

Physical layers

$$\text{Task} = \mathbf{W}_n \ldots \mathrm{ReLU}[\mathbf{W}_1\, \mathrm{ReLU}[\mathbf{W}_0\, f[\mathbf{I}_0]]\ldots]$$

What physical parameters effect image formation?

- Illumination
  - Spatial pattern
  - Angle of incidence
  - Color, polarization
- Lens and optics
  - Position/orientation
  - Shape
  - Focus
  - Transparency
- Detector
  - Pixel size
  - Pixel shape & fill factor
  - Color filters
  - Other filters
- Digitization
  - E to P curves
  - Digitization schemes/thresholds
  - Data transmission, multiplexing
- Physical object



Physical Layers

Digital Layers

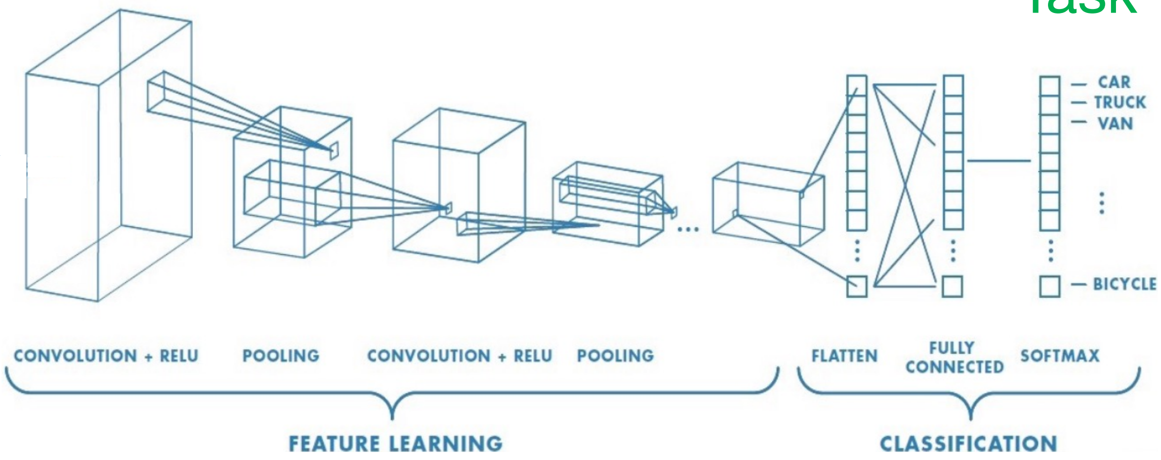
Input image data $\mathbf{I_0}$

f[ ]

$$\mathbf{I_s} = f[\mathbf{I_0}]$$

Digitized $\mathbf{I_s}$

Task



Q: Where and how should I implement my physical layer?

A: It depends on your data and implementation

- Situation #1: Fully simulated physical layers
- Situation #2: Experimentally-driven physical layers

Situation #2: Fully simulated physical layers

Generated Labels

Experimental measurements → "Expert" annotation

Physical Layers

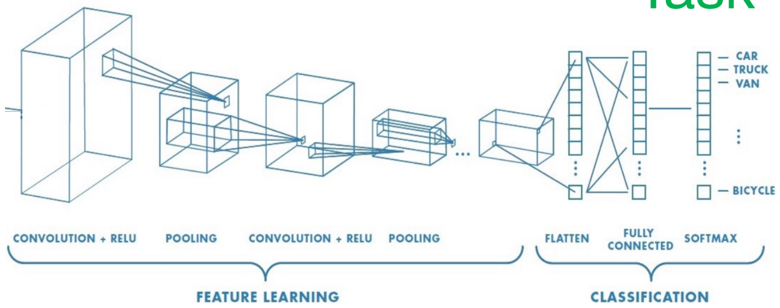
f[ ]

$$\mathbf{I_s} = f[\mathbf{I_0}]$$

Digitized $\mathbf{I_s}$

Digital Layers

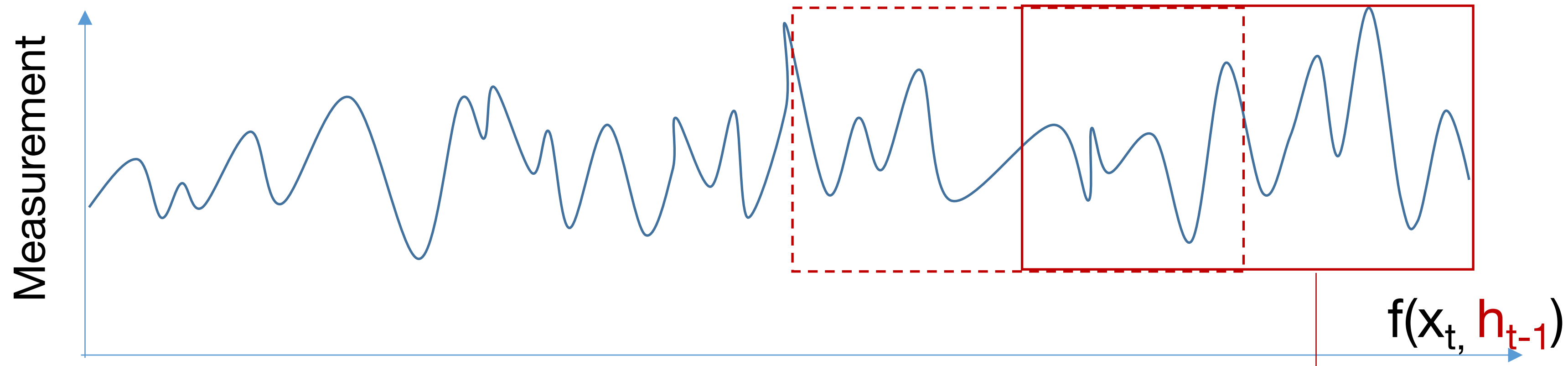
Task



<u>Pro's of experimental measurements</u>: Don't need to worry about making your simulations match the setup! (HUGE)

<u>Con's of experimental measurements:</u> You'll need to label them, limited access to desired sample information, often need to exploit some fundamental physical property

Many-to-one recurrent neural network



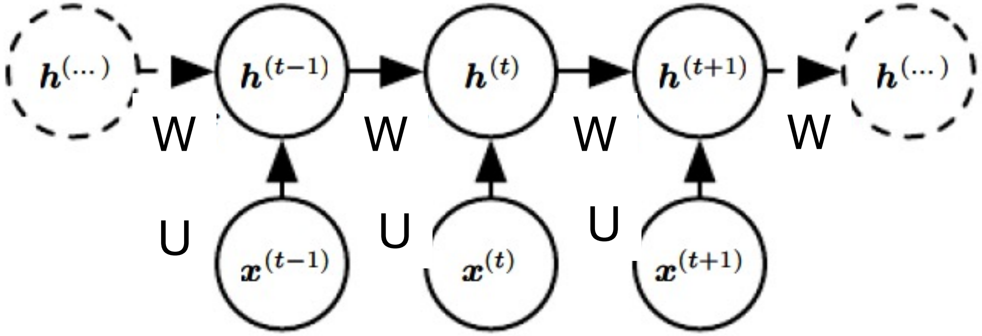
State h_t

$\boxed{V}$

Output o_t

Label y → Loss function $L(o_t, y)$

Backpropagate to
minimize dL/dW, dL/DU



Simple network structure:

$$\boxed{h_t = \mathrm{ReLU}[\mathbf{W}h_{t-1} + \mathbf{U}x_t + b]}$$

Taxonomy of Generative Models

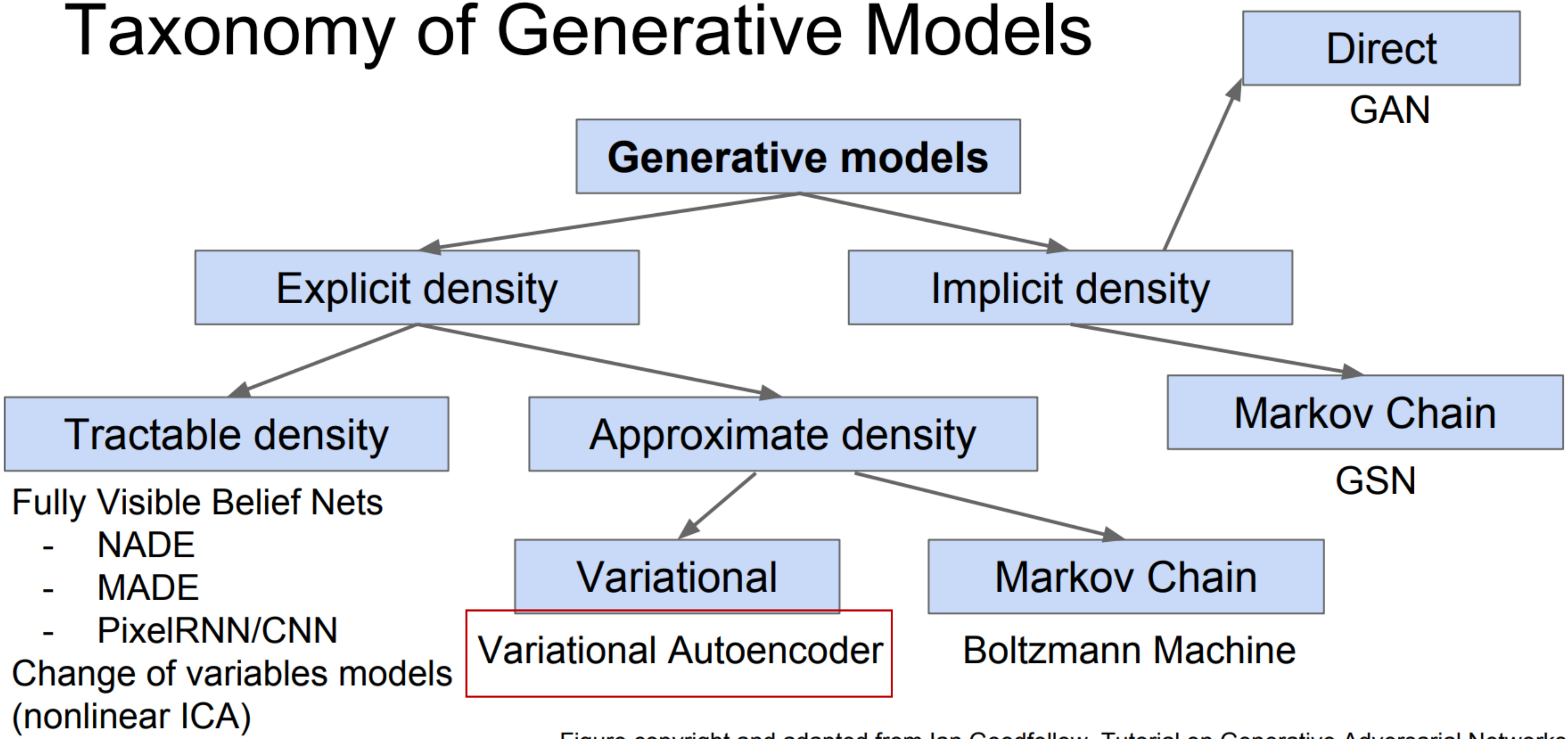
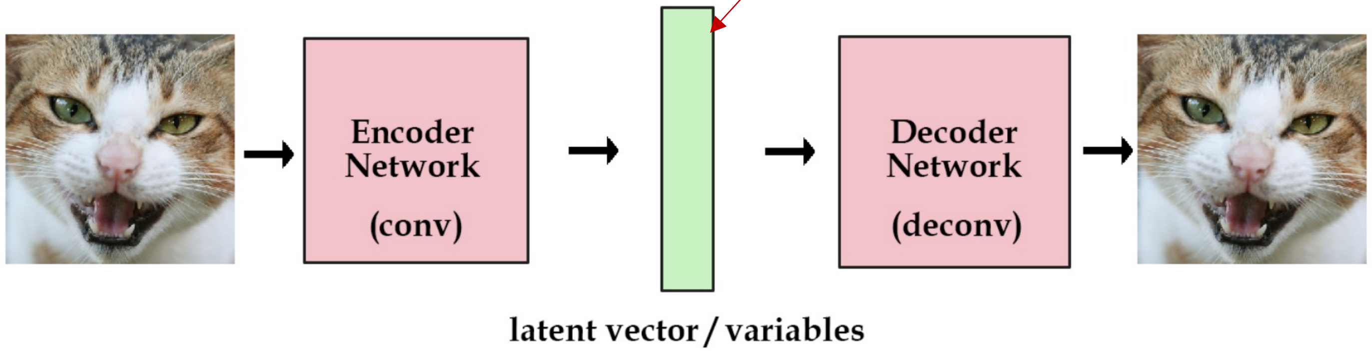


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Stanford CS231n, Lecture 12

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

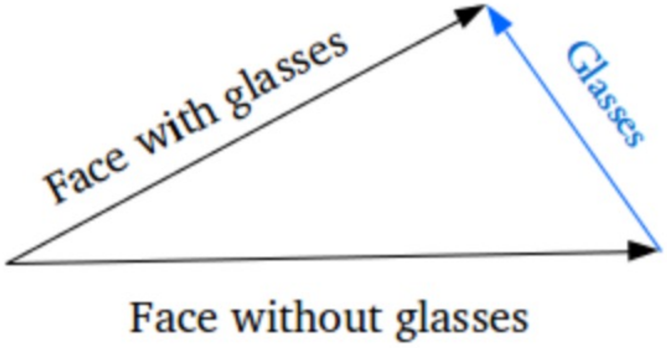


Minimize (KL) distance between latent vector and Gaussian normal

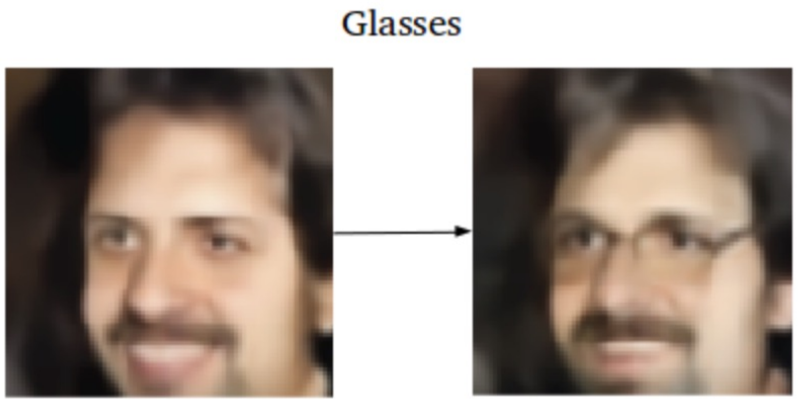
Generative Example (once trained):

- Encode image with glasses, obtain latent vector PDF $\mathbf{P_g}$
- Encode image without glasses, obtain PDF $\mathbf{P_{ng}}$
- Compute **diff** = $\mathbf{P_g - P_{ng}}$
- Encode new image to obtain $\mathbf{P_{new}}$, add in **diff**
- Decode $\mathbf{P_{new}}$ + **diff** to get guy with glasses!

- With Gaussian PDF, can start to add/subtract latent vector in a normalized vector space



Adding new features to samples



Exploring a specific variation of input data[1]

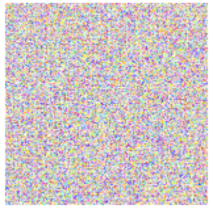
II.B - G/D Game

100-d random code

$$z = \begin{pmatrix} 0.47 \\ \vdots \\ 0.19 \end{pmatrix}$$

Generator "G" (Neural Network)

(64,64,3) generated image



Run Adam simultaneously on two minibatches (true data / generated data)

x

Discriminator "D" (Neural Network)

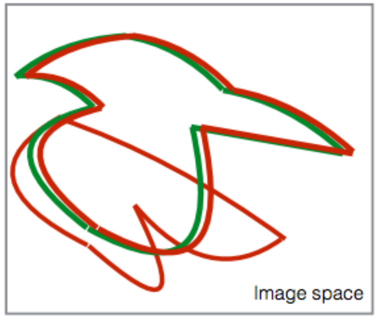
Gradients

Binary classification

$$\begin{cases} y = 0 & if \quad x = G(z) \\ y = 1 & otherwise \end{cases}$$

Real images (database)

Probability distributions



Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

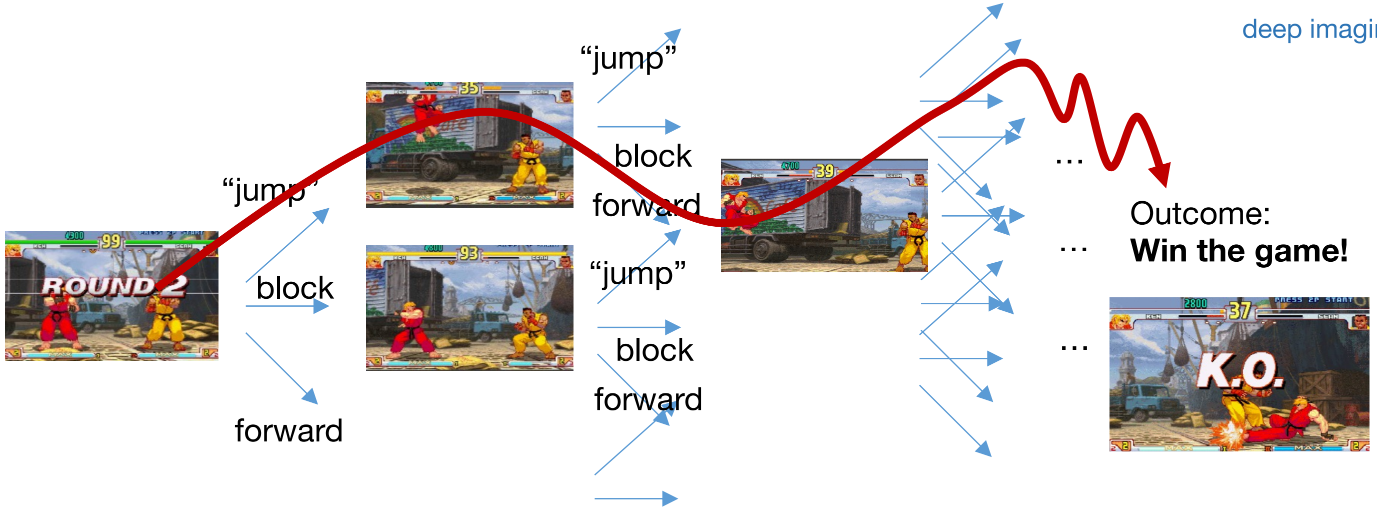
deep imaging

Supervised ML

Outcome:
Cell type B

- Fixed image sequence
- Goal: match to known label
  (large labeled dataset needed)
- Output: label
- Examines all data

Reinforcement learning



- Dynamic image sequence
- Goal: get to known desired outcome
  (no labels needed, really…)
- Output: sequence of actions
- Not possible to examine *all* data

deep imaging

Thanks for a great semester everyone!