RBE474X/595-B01-ST: Deep Learning For Perception

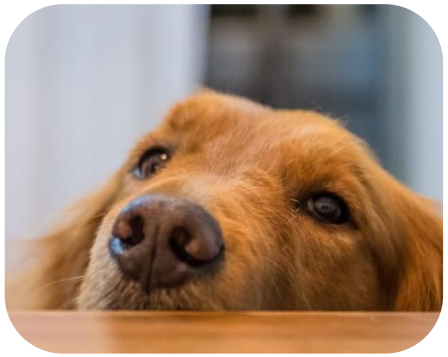
Class 3: NN Tuning, Image Filtering And Convolutional Neural Networks

Prof. Wei Xiao

Dog or Cat

Image Classification

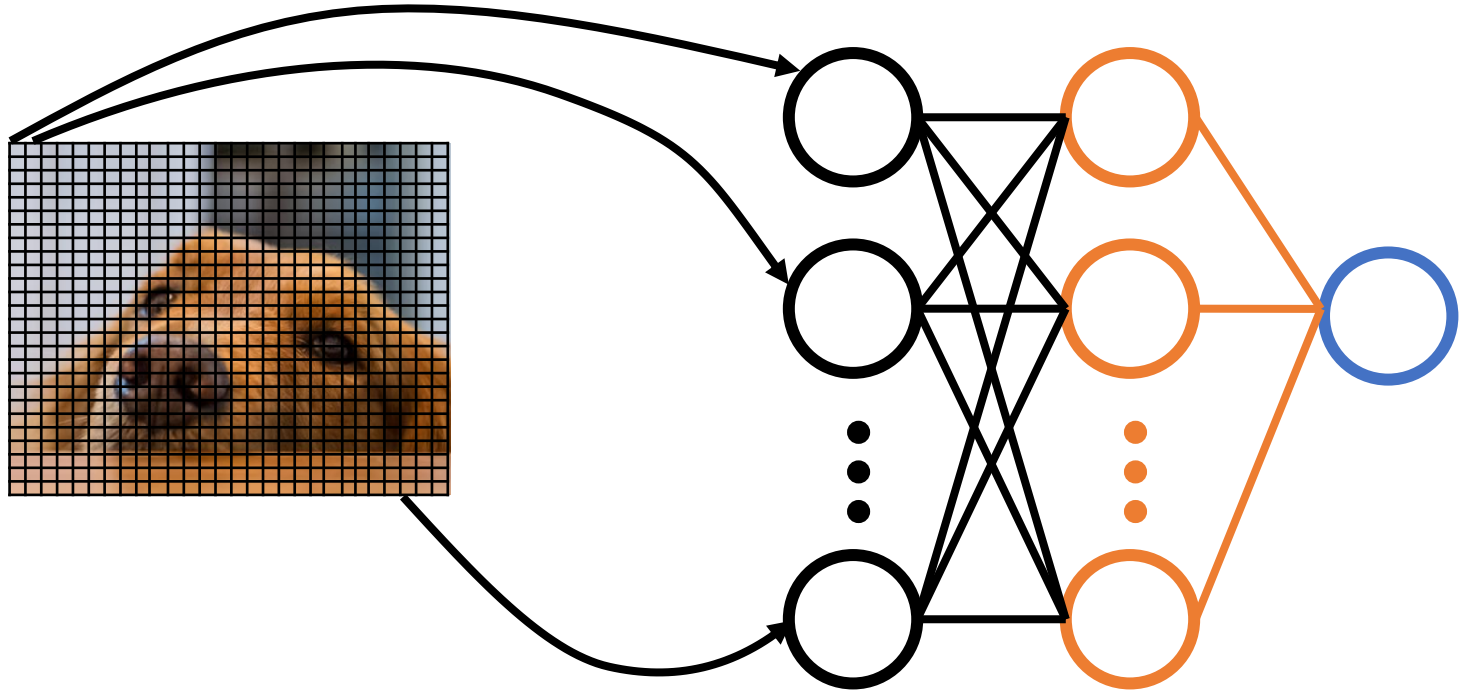
DIY Binary Classifier



Magical Classifier

$p(dog|I)$

Multi-Layer Perceptron



W_1 σ_1 W_2 σ_2 W_3 Loss

$x_i \rightarrow z_1 \rightarrow y_1 \rightarrow z_2 \rightarrow y_2 \rightarrow z_3 \rightarrow \mathcal{L}$

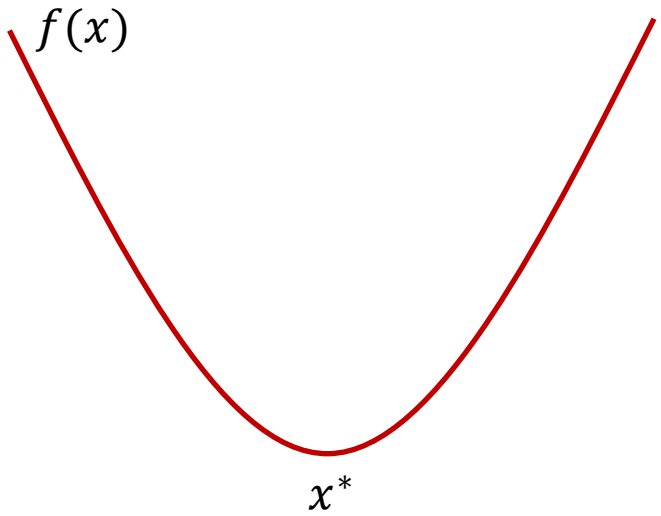
Forward Pass → Outputs

Error Gradients ← Backward Pass

You Got The Gradients!

Now What?

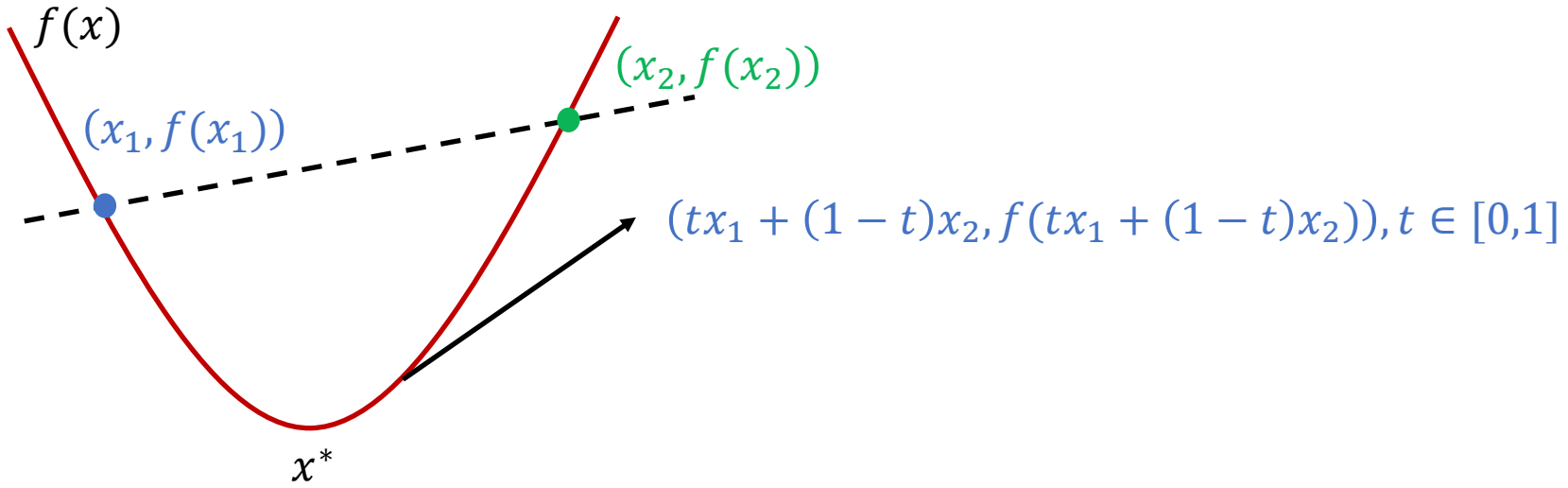
Numerical Optimization 101



Convex function

$$x^* = \underset{x}{\operatorname{argmin}}\, f(x)$$

Convex Function

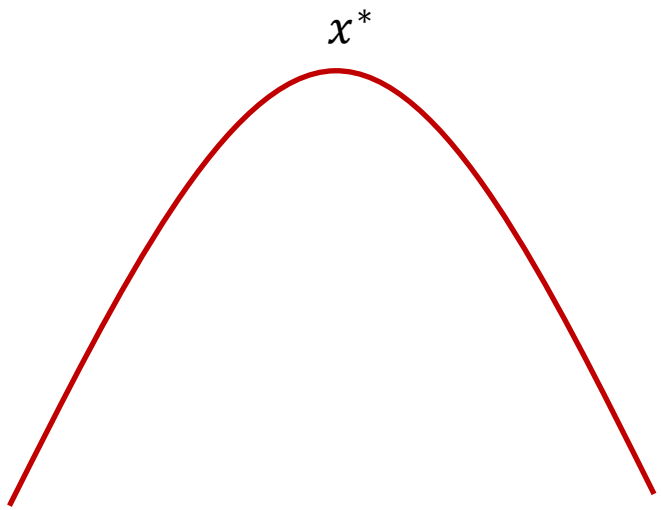


Convex function

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

Known as “Jensen’s Inequality”

Concave Function

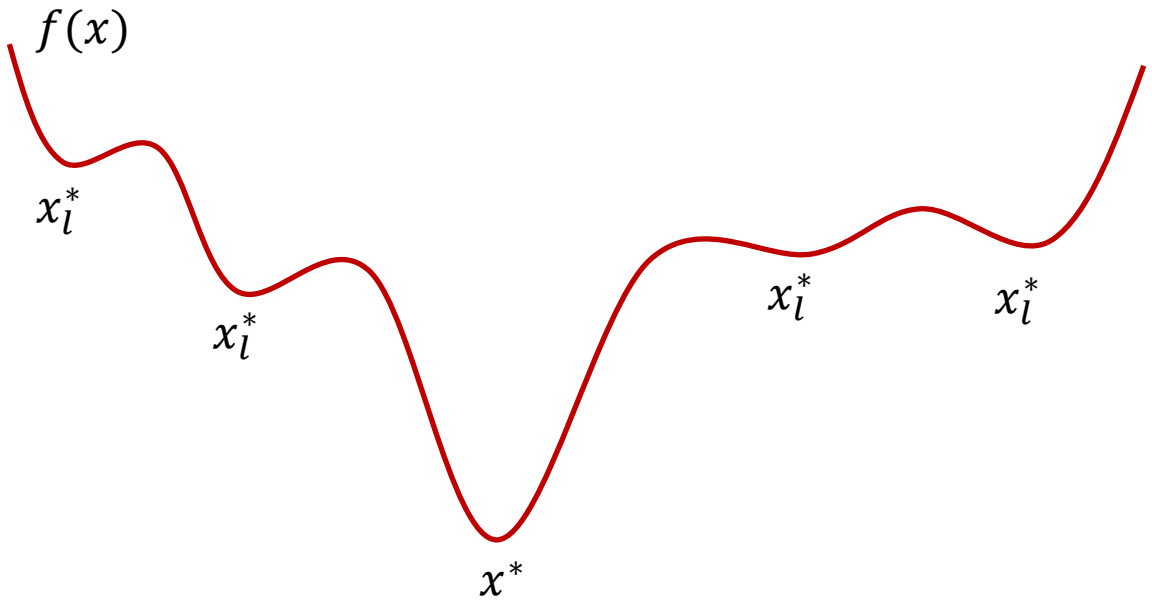


Concave function

$$x^* = \underset{x}{\operatorname{argmax}}\, f(x)$$

Intuitively:
Concave = −Convex or vice versa

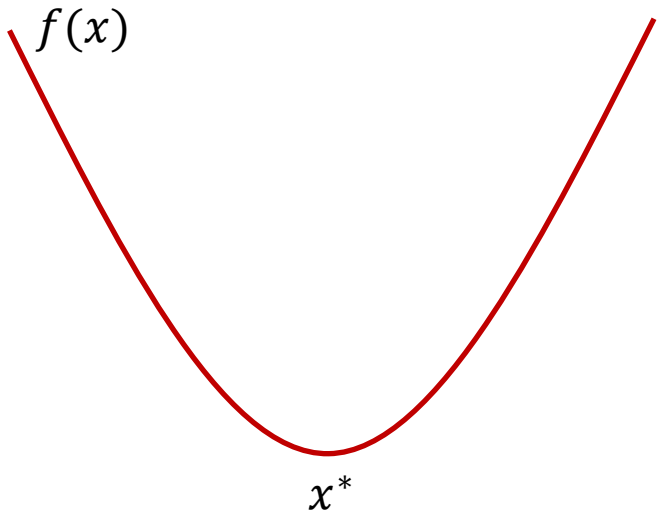
Non-Convex Function



Non-convex function

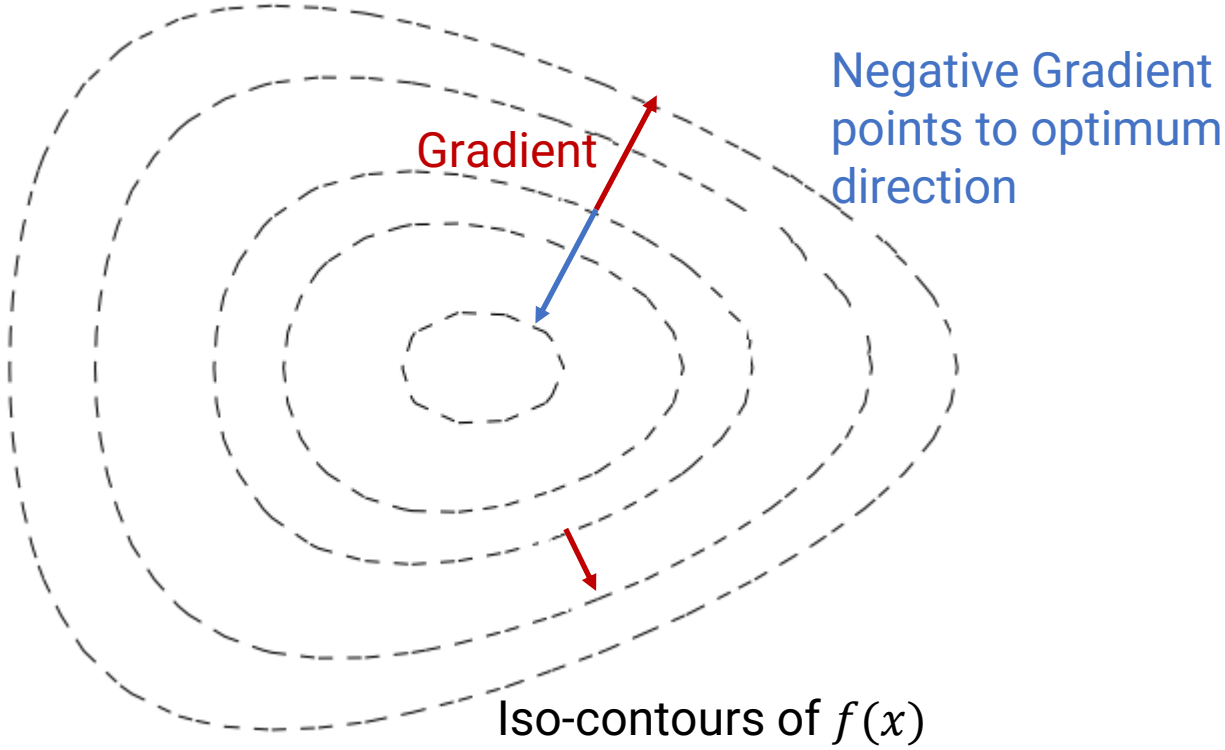
Neither convex nor concave
Intuitively has a lot of local optimum

Convex Optimization 101



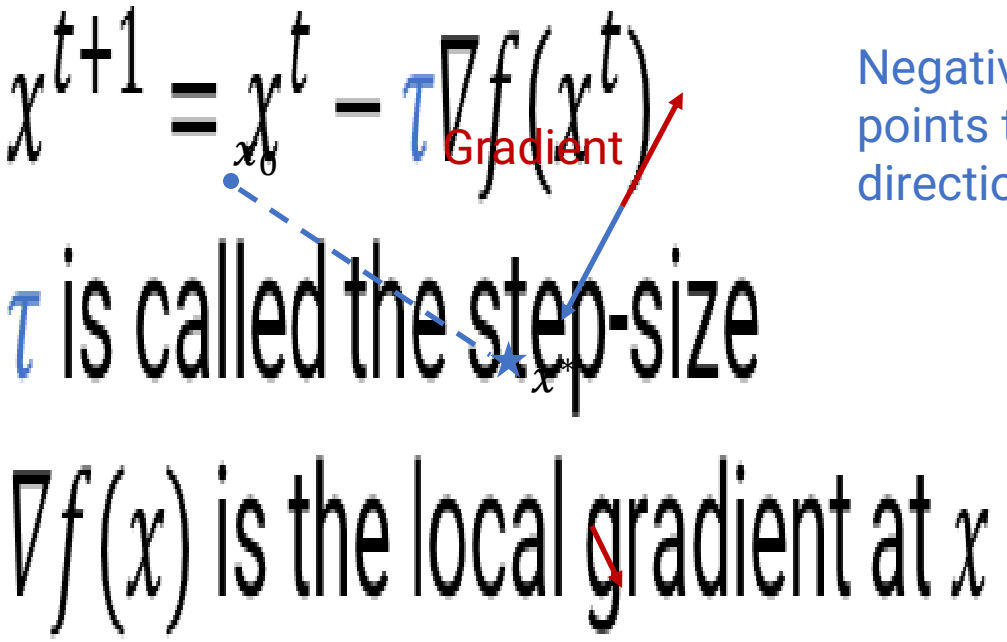
Convex function

Gradient Direction is the direction of steepest descent



Steepest Descent

$x^{t+1} = x^t - \tau \nabla f(x^t)$
τ is called the step-size
$\nabla f(x)$ is the local gradient at x



Negative Gradient points to optimum direction

Iso-contours of $f(x)$

Step-size Restrictions

$\tau < \frac{2}{\alpha}$ for $f(x) = \frac{\alpha}{2}x^2$

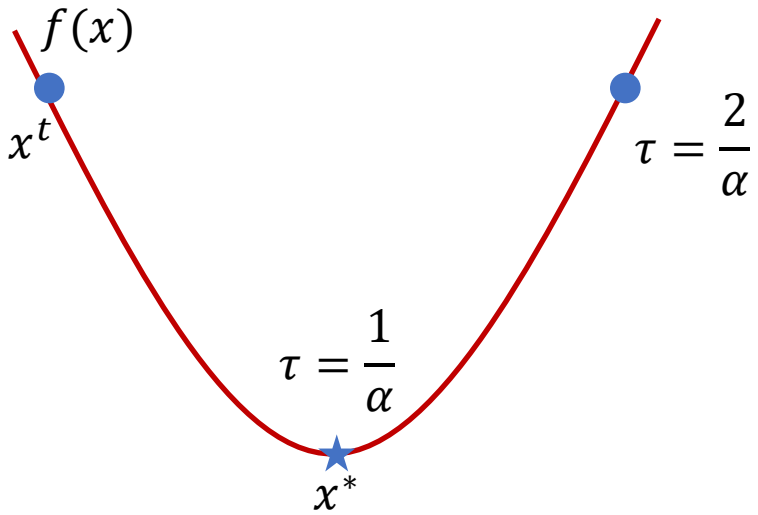
$\nabla f(x) = \alpha x$

$x^{t+1} = x^t - \tau\alpha x^t$

Sort of like PD controller

Too high τ will cause you to diverge

Too low τ will take forever to converge



Convex function

Lipschitz Constant

$\|\nabla f(x) - \nabla f(y)\| \leq M\|x - y\|$

Here M is the **Lipschitz constant** or intuitively M represents a function of maximum curvature

If a Hessian exists: $M \geq \left\|\nabla^2 f(x)\right\|$

The step-size restriction becomes

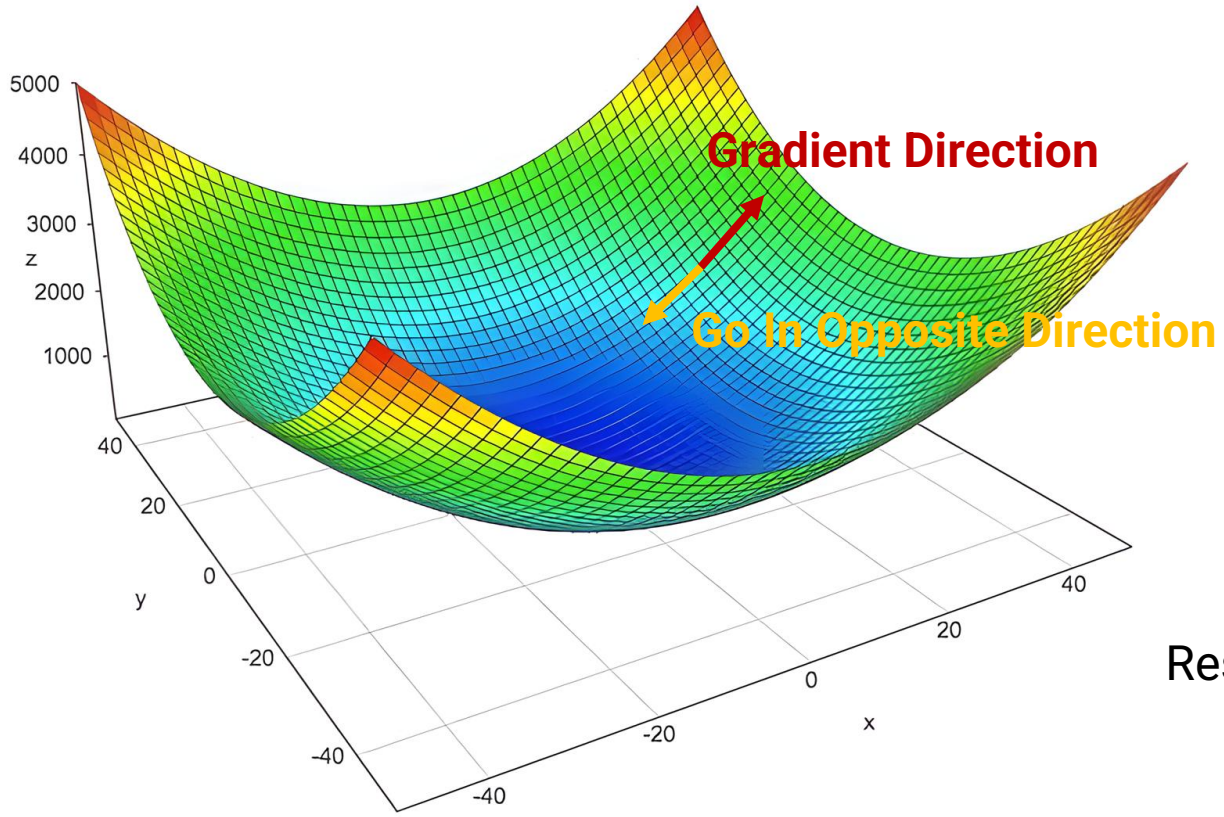
$$\tau < \frac{2}{M}$$

It is generally hard to obtain a value of M

There are methods to find "best" τ for each step and are called **Line Search Methods**

But How Do You Actually Train?

Gradient Descent!

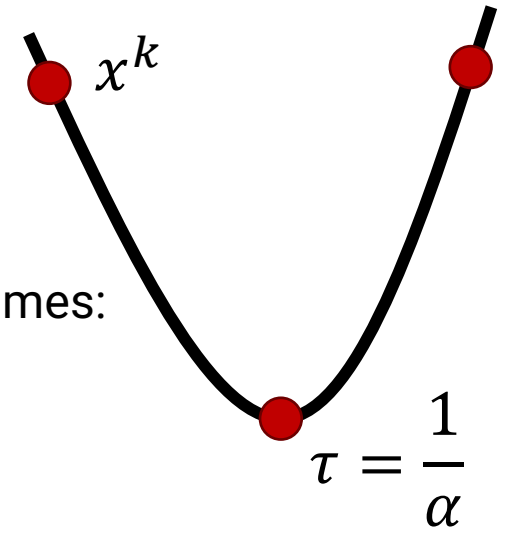


$$x^{k+1} = x^k - \tau \nabla f\left(x^k\right)$$

Learning Rate!

$$f(x) = \frac{\alpha}{2}x^2 \Rightarrow \nabla f(x) = \alpha x$$

$$x^{k+1} = x^k - \tau\left(\alpha x^k\right)$$



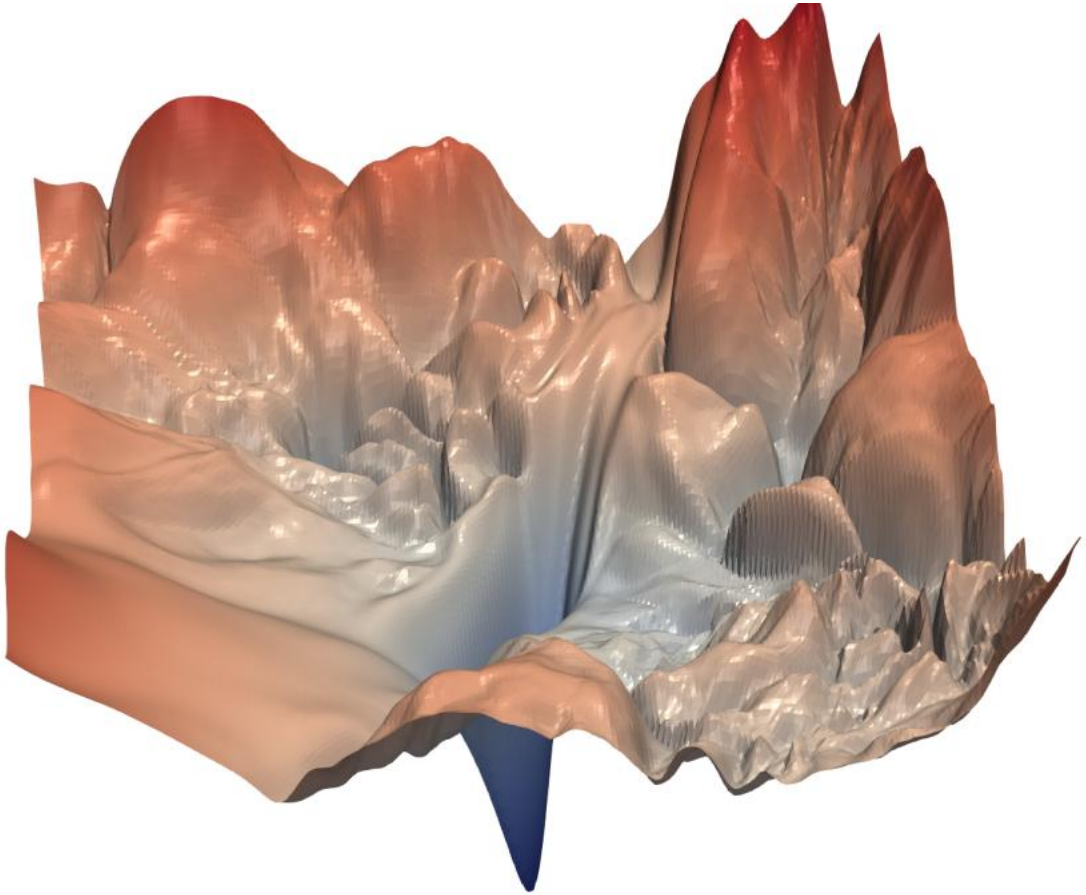
$$\tau = \frac{2}{\alpha}$$

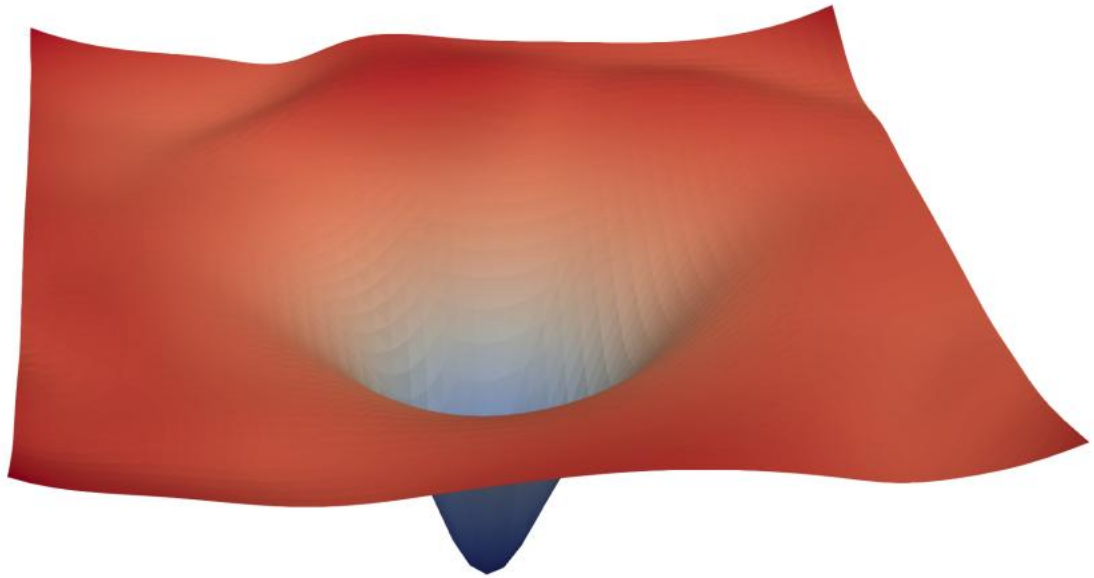
Restriction becomes:

$$\tau < \frac{2}{\alpha}$$

$$\tau = \frac{1}{\alpha}$$

NNs Are Not Necessarily Convex!





So how do they work?!

A better architecture!

Li, Hao, et al. "Visualizing the loss landscape of neural nets." *Advances in neural information processing systems* 31 (2018).

But How Do You Actually Train?

Stochastic Gradient Descent!

You compute ∇ on what set, ideally?
Entire set! Not always possible!

Solution?
Do it in mini-batches!
This is called Stochastic Gradient Descent (SGD)

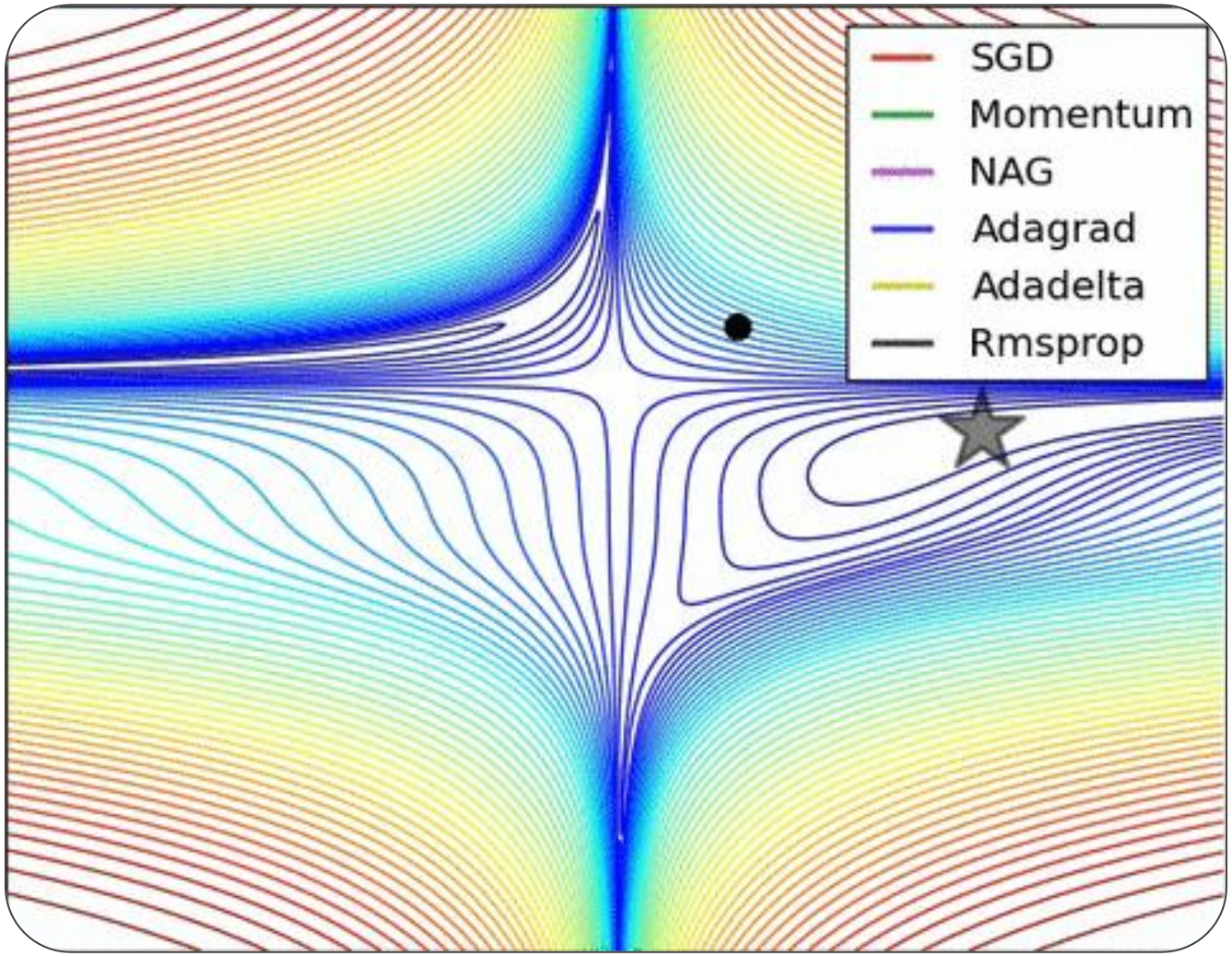
Also look up fancy optimizers like ADAM, RMSProp, AdaGrad, AdaDelta

Various Optimization Algorithms For Training Neural Network | by Sanket Doshi | Towards Data Science

Optimizers Galore

Modify the update rule based on gradient heuristics

- An approximation for local convexity!
- A way to deal with noise!
- A way to make things work with lesser data!
- Read up on `ADAM`, `AdaGrad`, `AdaDelta`, `RMSProp`

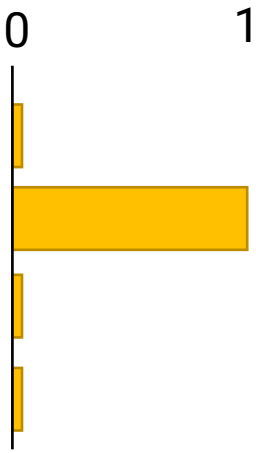


A journey into Optimization algorithms for Deep Neural Networks | AI Summer (theaisummer.com)

What Is $\mathcal{L}$?

- y is a class label such as “dog” or “cat”
- For multi-class case, can be many names and hence can be encoded as numbers
- But even better way is One-Hot Encoding!
- Sometimes people use “out of dataset” class as well!

“2” becomes



For 4 classes

- Raw output of network when values are $\mathbb{R}^{N\times 1}$ are called **Logits**!
- Logits can be converted to probabilities using softmax
- Be careful of what loss function you are using

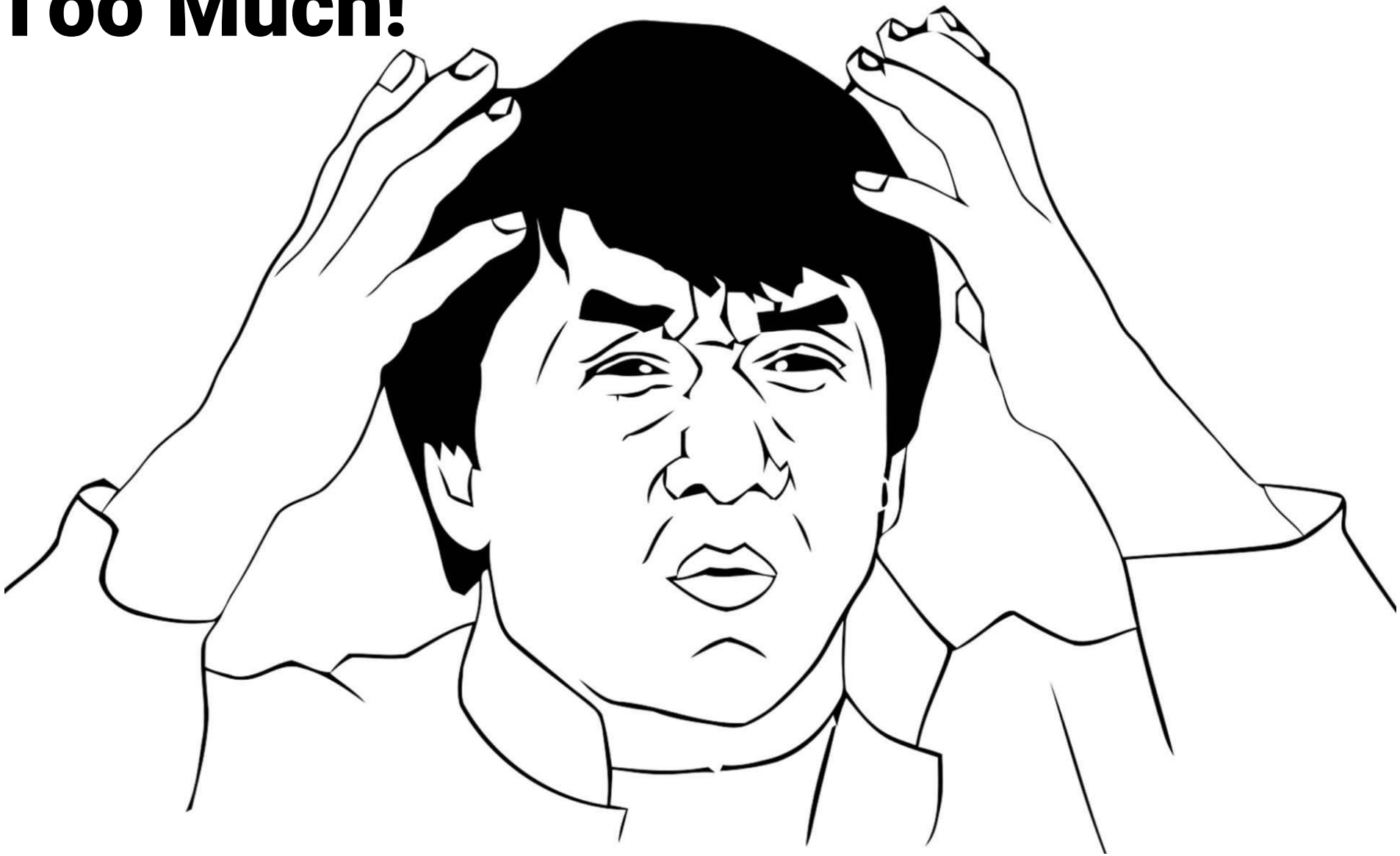
Some common loss functions (for classification) are:

- Cross entropy
- MSE
- Hinge Loss
- KL Divergence Loss

Will be covered in recitation session!

How to Choose Loss Functions When Training Deep Learning Neural Networks (machinelearningmastery.com)

It’s Too Much!



Rule(s) Of Thumb

☹ **I don't know how to pick an architecture!**

- Number of neurons per layer
- Number of layers

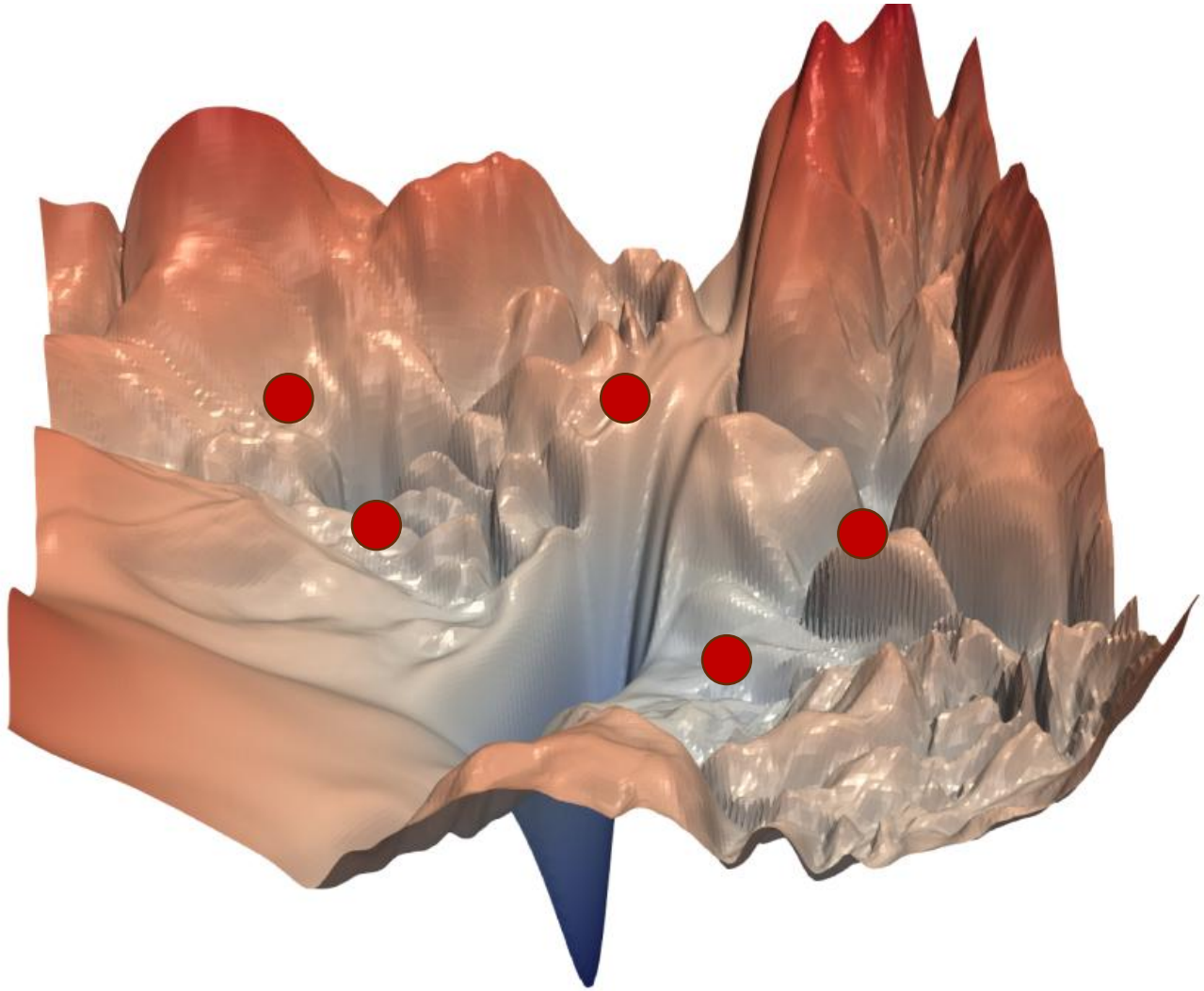
☺ **Lookup papers which do similar things**

- Adapt to fit your compute hardware
- Adapt to fit your problem statement
  - Change input/output dimensions

Rule(s) Of Thumb

☹ **I don't how to initialize my weights**
- Starting point of training

☺ **Adapt pre-trained weights**
☺ **Adapt better weight initialization**
- Xavier Initialization
- Uniform Initialization
- Try a few initializations and choose best!

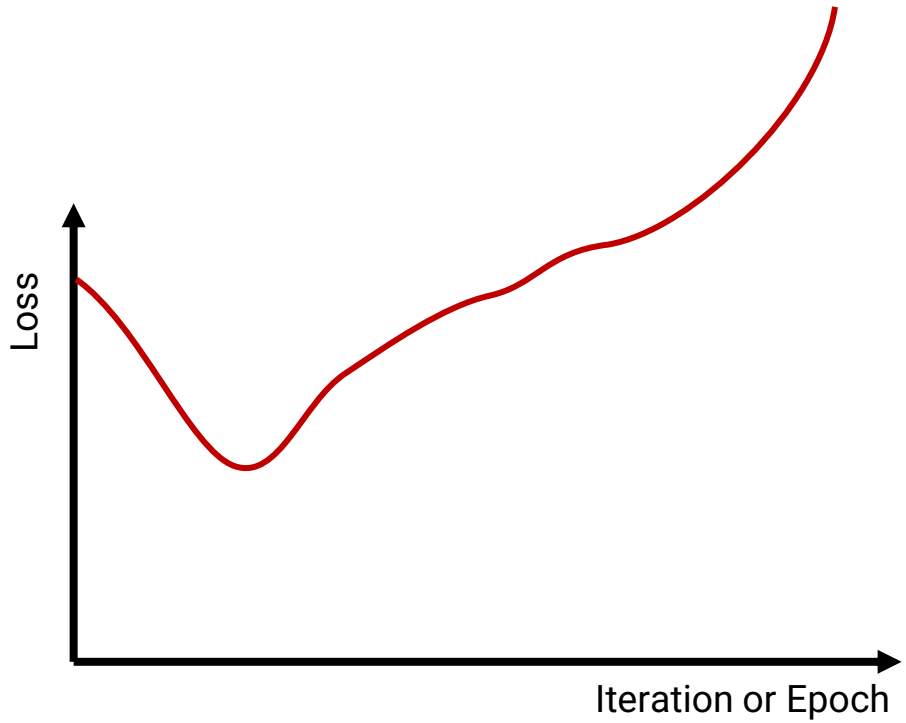


Rule(s) Of Thumb

☹ **My network output/gradient is exploding to ∞**

- After a few iterations, weights and outputs become large!

☺ **Keep weight initialization small**
☺ **Decrease learning rate**
☺ **Choose a different activation function σ**
☺ **Adapt better weight initialization**
☺ **Check your gradient calculation**

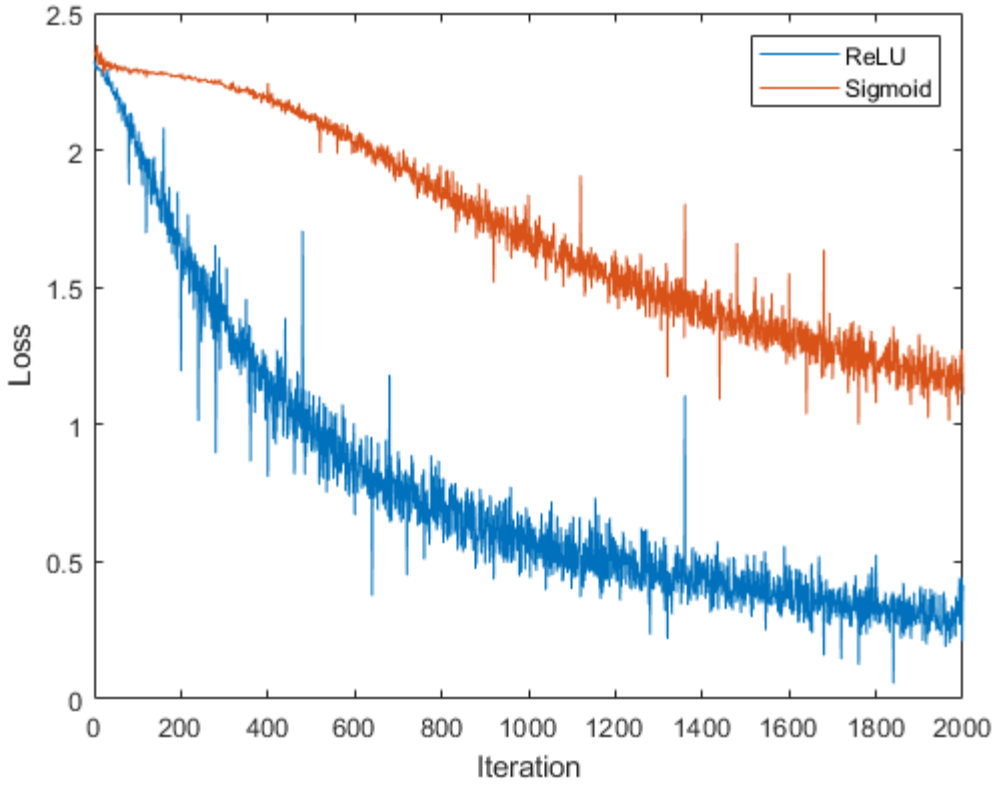


Rule(s) Of Thumb

☹ **My network loss drop is very slow**

- Learning is slow
- Gradients are tiny

☺ **Increase weight initialization norm size**
☺ **Increase learning rate**
☺ **Choose a different activation function σ**
☺ **Change architecture to enable faster training**



Rule(s) Of Thumb

☹ **I tried everything! My network is not training!**

- Loss oscillating around some number

☺ **Check your dataset for issues**

- Labels are given as you expect
- No artifacts in the dataset

☺ **Check your loss function is setup properly**

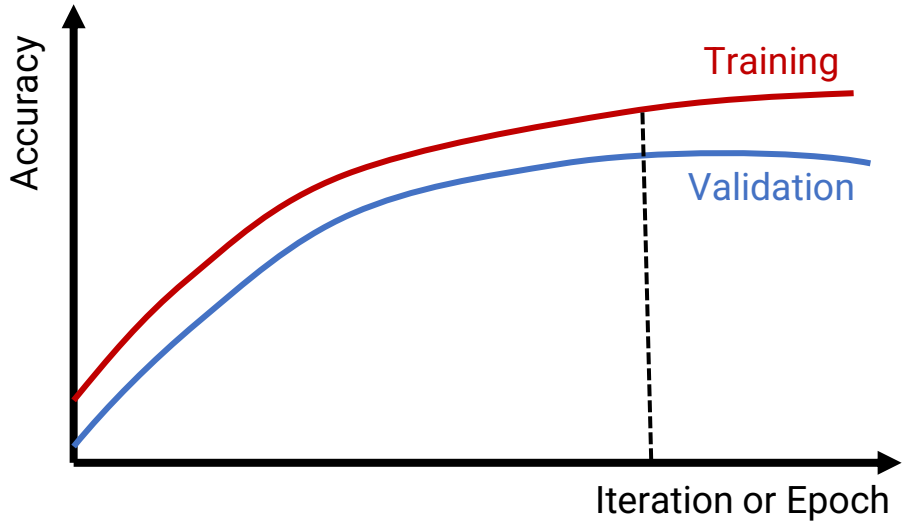
- Logits vs Softmax

☺ **Try changing your optimizer, almost always use `ADAM`**

Let's Recap

Iteration: One forward + backward pass (generally every mini-batch)
Epoch: Number of iterations to make up training set size such that you have used entire training set for training!

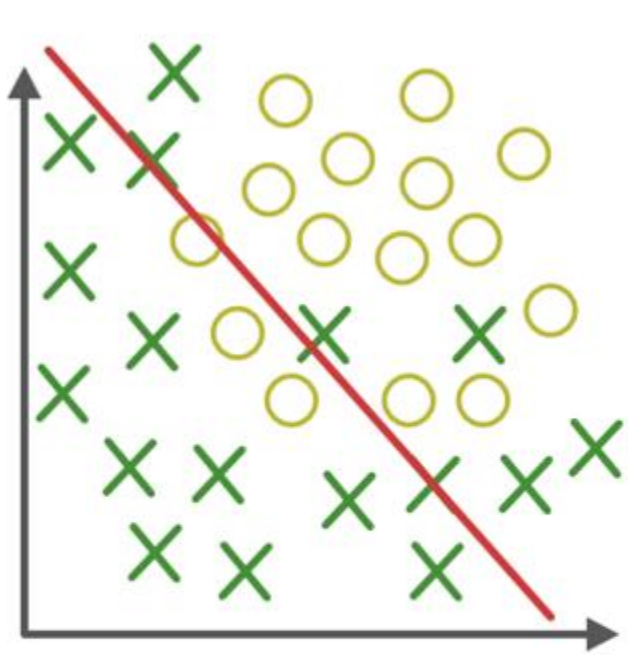
- **Step 1:** Obtain your data, split it into Training and Validation
  - Use any Cross-Validation method
- **Step 2:** Build Your Network of Choice
  - MLP or CNN or anything else and hyperparameters
- **Step 3:** Choose your Loss Function $\mathcal{L}$
  - We'll talk about this later
- **Step 4:** Perform Training Procedure
  - Perform forward pass per batch
  - Perform backward pass (Backpropagation)
    - Computing Gradients
    - Gradient Descent
  - Perform forward + backward pass either per iteration or per epoch
- **Step 5:** Check Validation accuracy every so often
  - When you determine overfitting or training not improving or exhausted max. iterations – stop training
- **Step 6:** Deploy your model on Test Set



Overfitting vs Underfitting

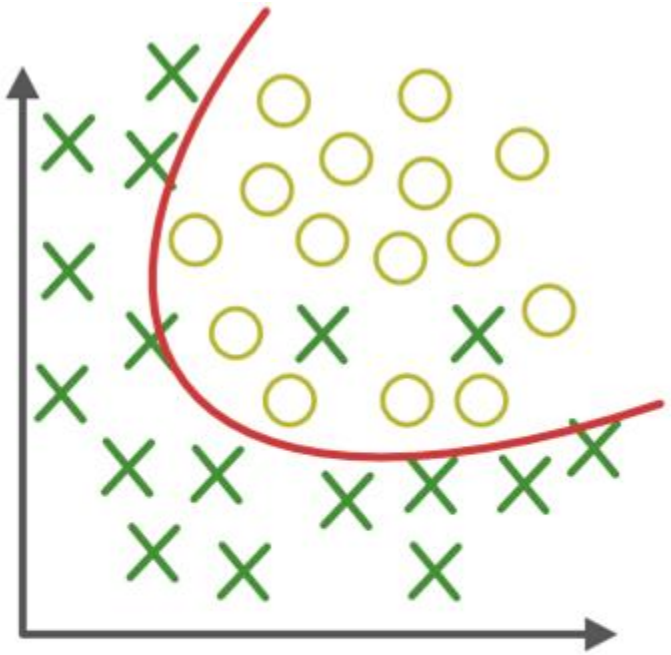
- **Overfitting**: when NN is good at learning its training set
  - But not able to generalize its predictions to additional unseen examples
- **Underfitting**: When NN is not able to accurately predict for the training set

Overfitting Vs Underfitting

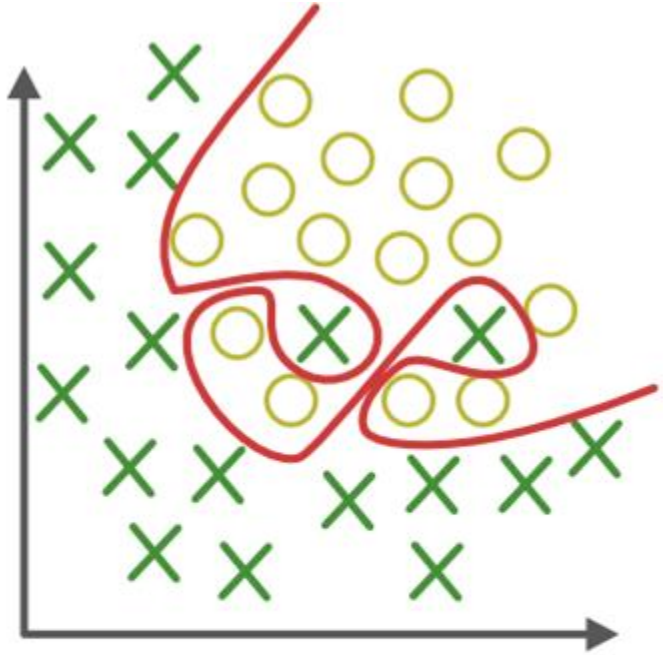


Under-fitting

(too simple to explain the variance)



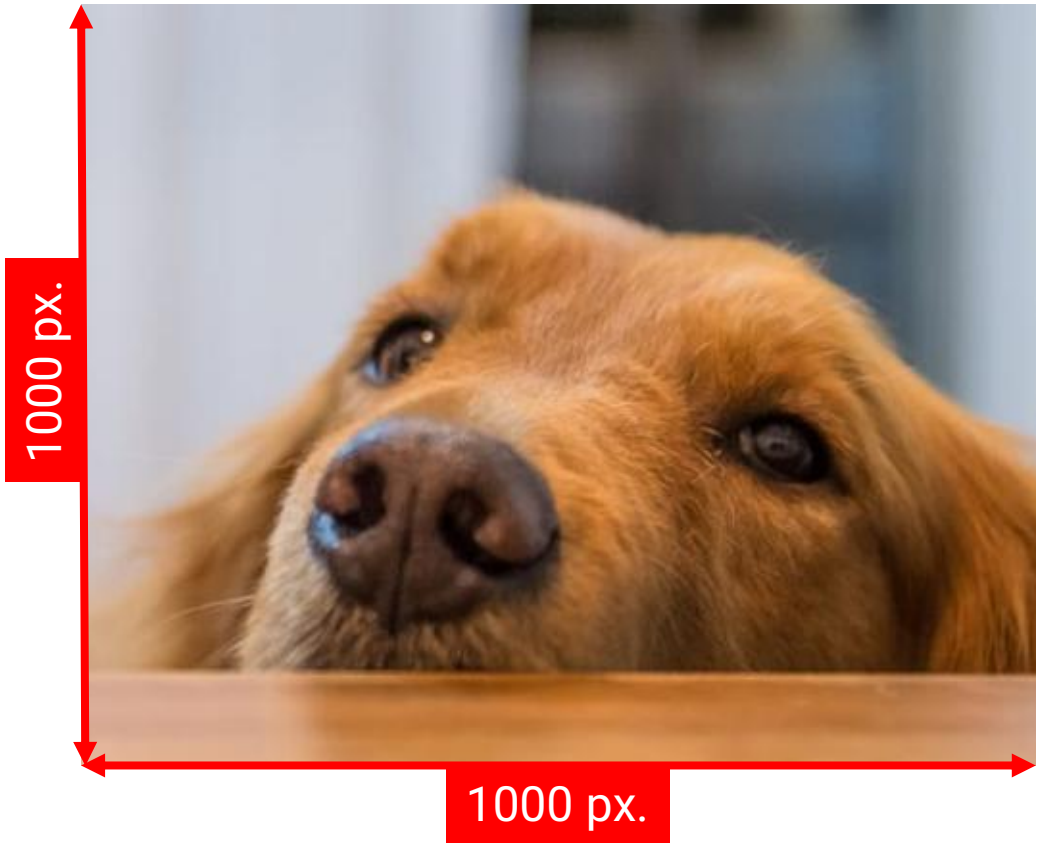
Appropirate-fitting



Over-fitting

(forcefitting--too good to be true)

Issues With This?



1MP image!
1 Million neurons in first layer!

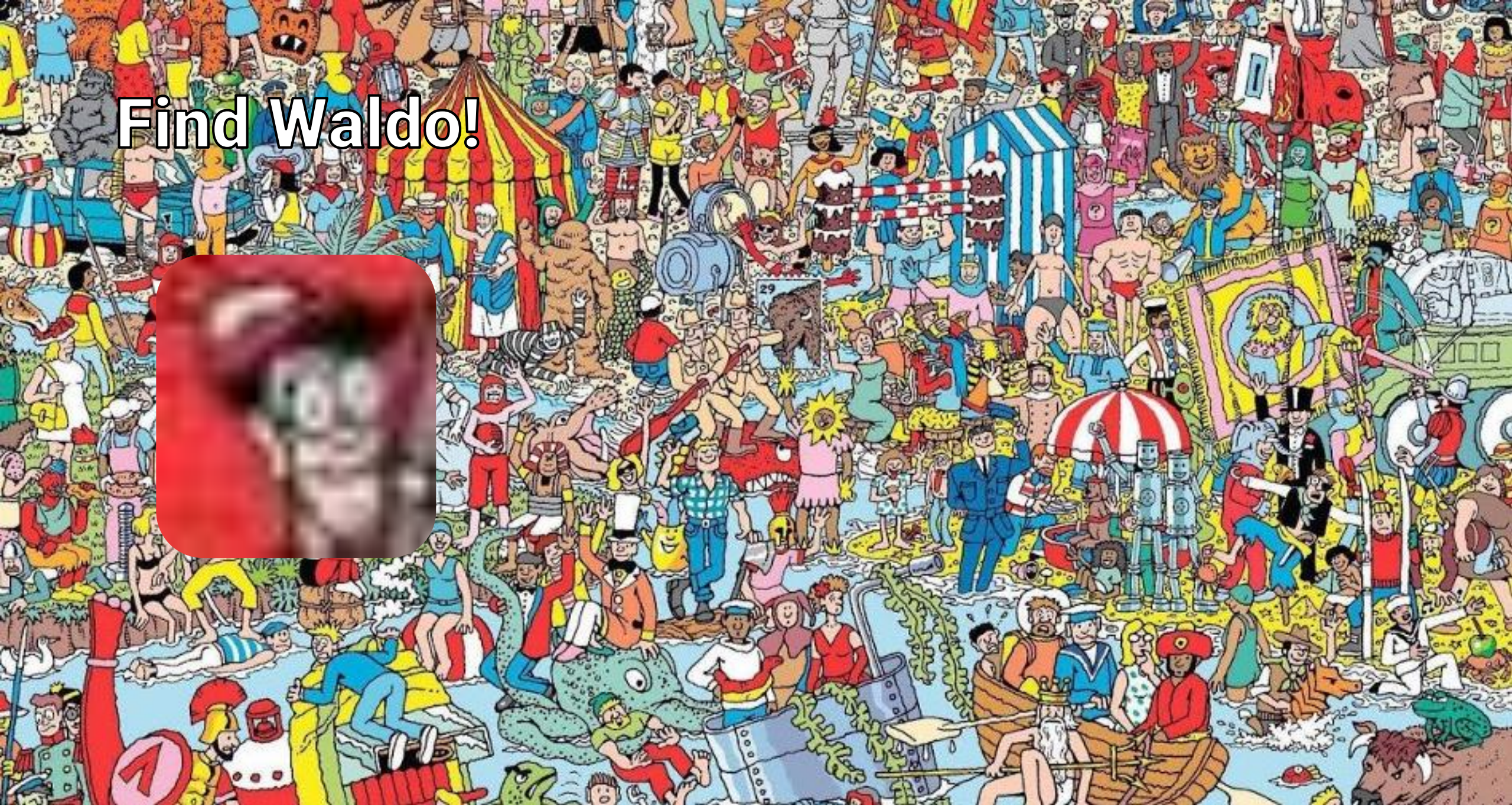
What to we do?

- Down sample
  - Lose features
- Pre-extract features?
  - Point of a Neural Network is lost!

A Great Invention!

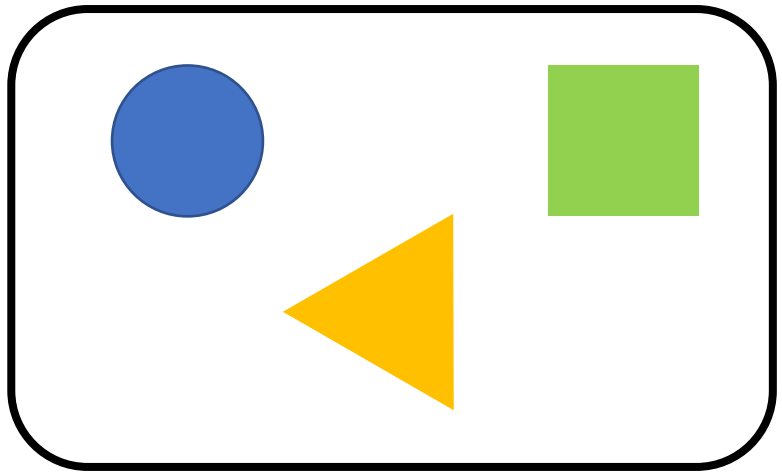
Turning Award in 2018! **A cool two decades later!**

Patterns Galore!

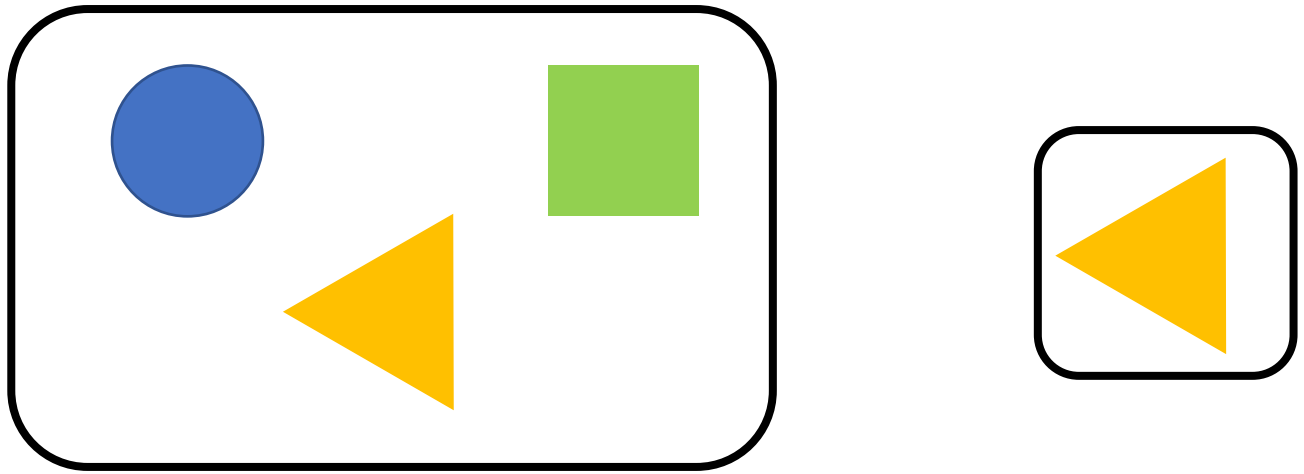

Find Waldo!


Find Waldo!

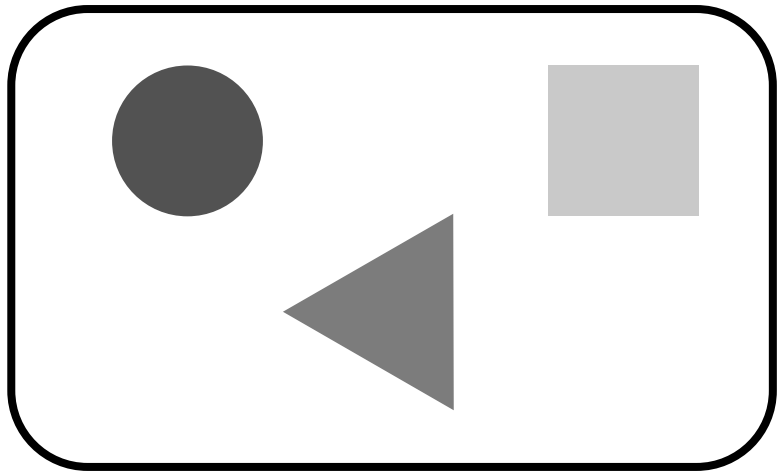
How Can A Computer Do This?



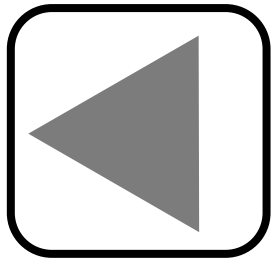
Sliding Windows



Sliding Windows

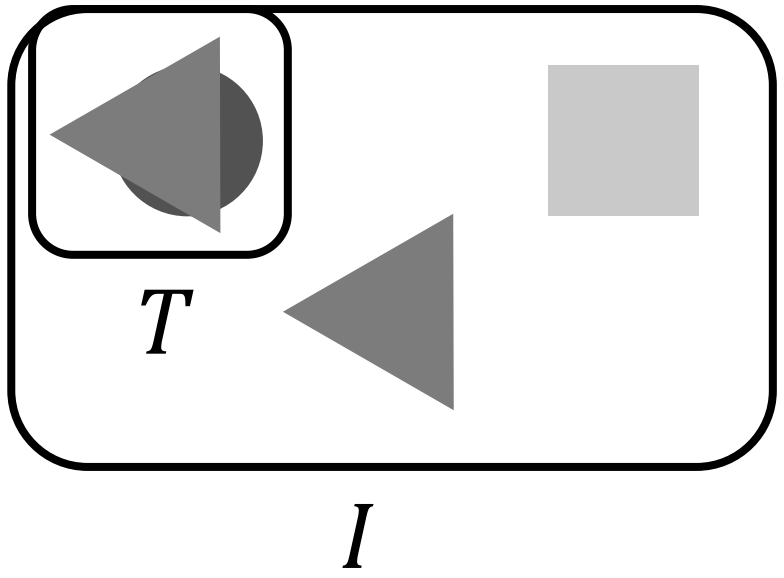


I

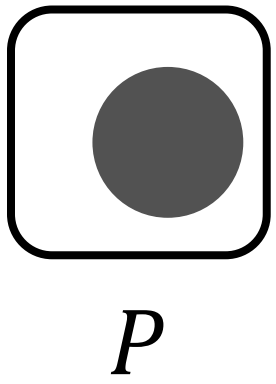


T

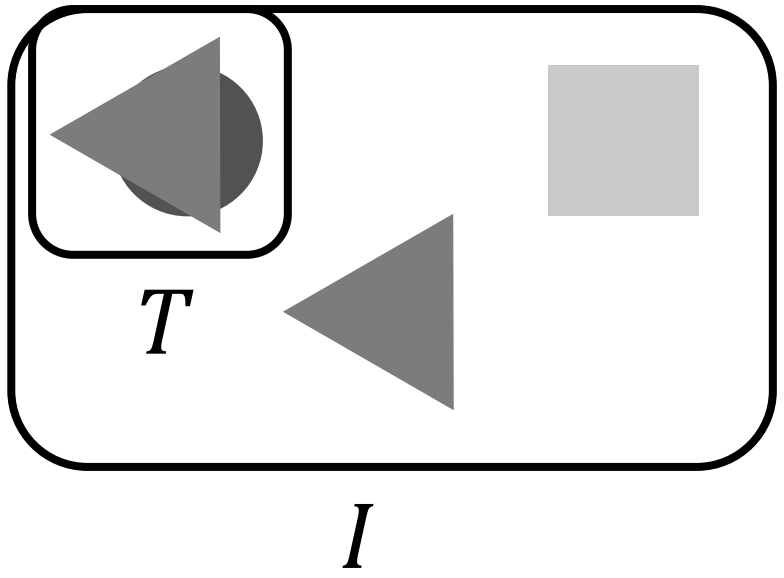
Sliding Windows



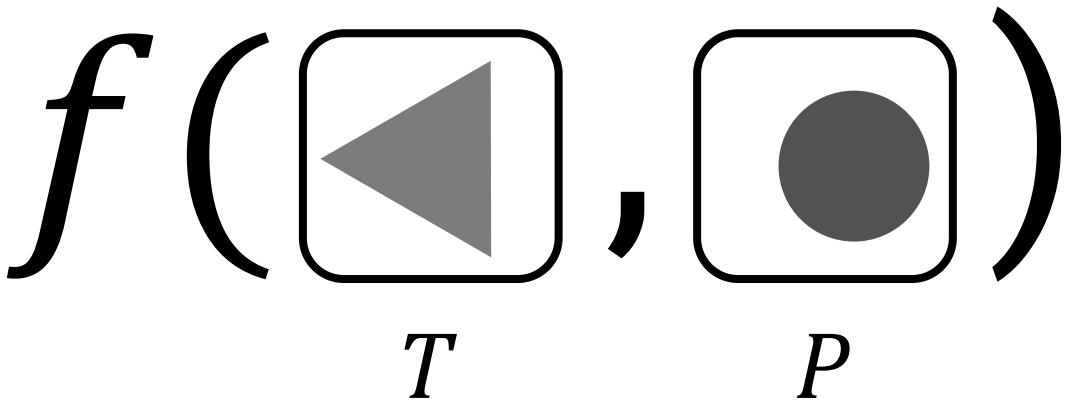
$f(P, T)$: Similarity between P and T
P is the patch/crop of I below T



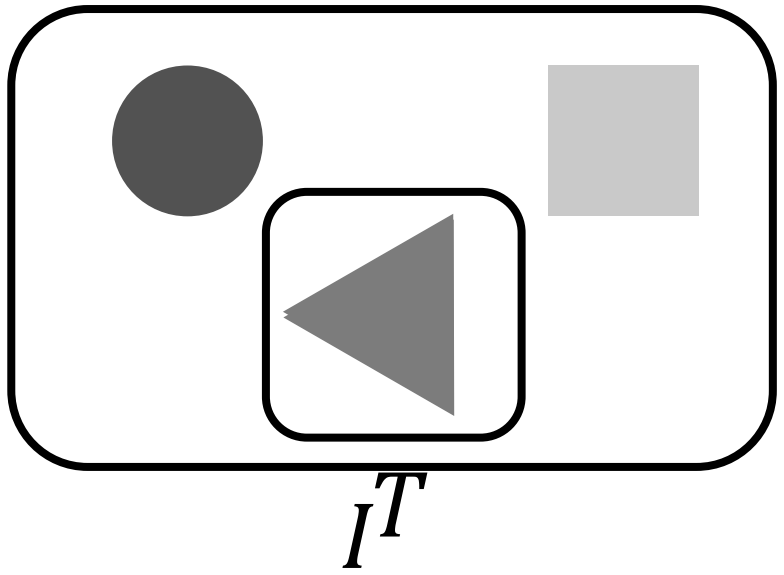
Sliding Windows



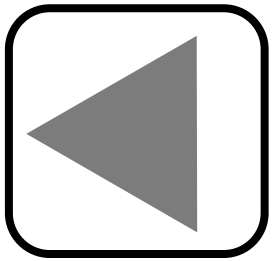
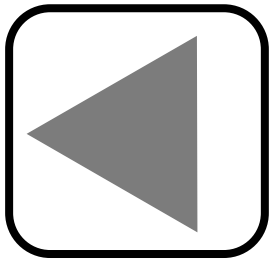
$f(P, T)$: Similarity between P and T
P is the patch/crop of I below T



Sliding Windows



$f(P, T)$: Similarity between P and T
P is the patch/crop of I below T

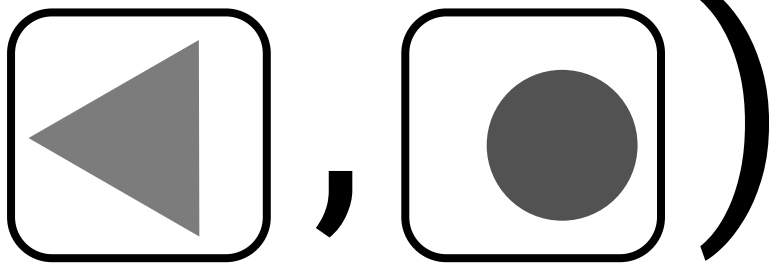
$$f\left(\begin{array}{c}\text{  }\\ T\end{array}, \begin{array}{c}\text{  }\\ P\end{array}\right)$$

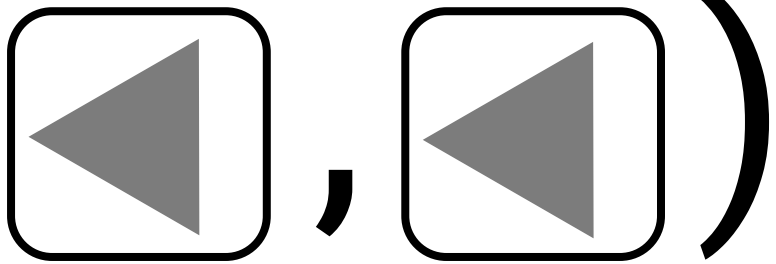
How Do We Measure Similarity?

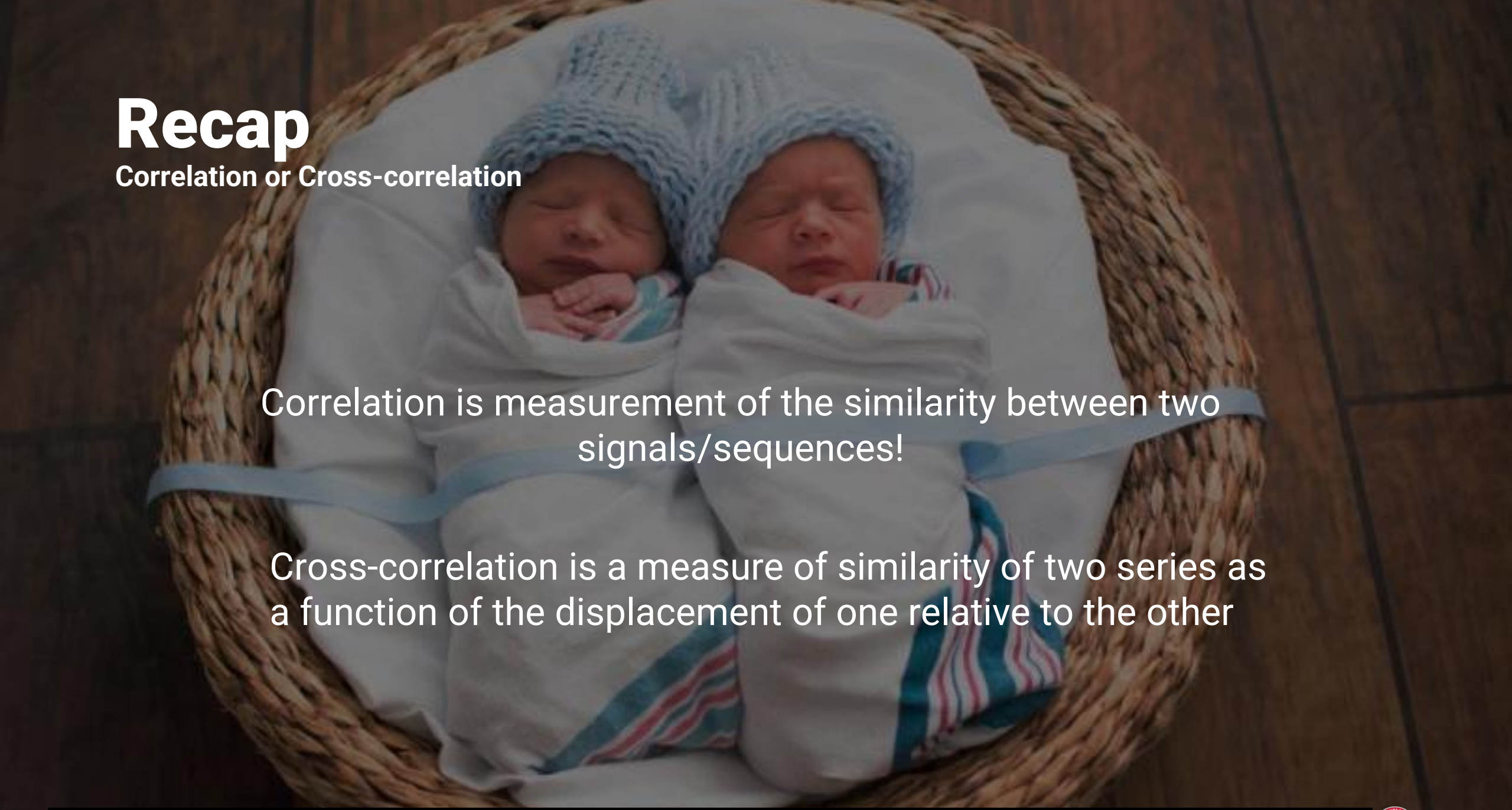
AKA what is f?

Correlation

$$f(x,y) = \sum_{\forall x'} \sum_{\forall y'} T(x,y) \cdot P(x+x', y+y')$$

f  Small Value! ⇒ Bad match!

f  Large Value! ⇒ Amazing match!



Recap

Correlation or Cross-correlation

Correlation is measurement of the similarity between two signals/sequences!

Cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other

Can we stack this method?

No! Correlation is not associative!

vs.

Recall Correlation

$$f(x,y) = \sum_{\forall x'} \sum_{\forall y'} T(x,y) \cdot P(x+x', y+y')$$

No flipping

Contrast to Convolution

$$f(x,y) = \sum_{\forall x'} \sum_{\forall y'} T(x,y) \cdot P(x - x', y - y')$$

Contrast to Convolution

$$f(x,y) = \sum_{\forall x'} \sum_{\forall y'} T(x,y) \cdot P(x - x', y - y')$$

Flipping

Contrast to Convolution

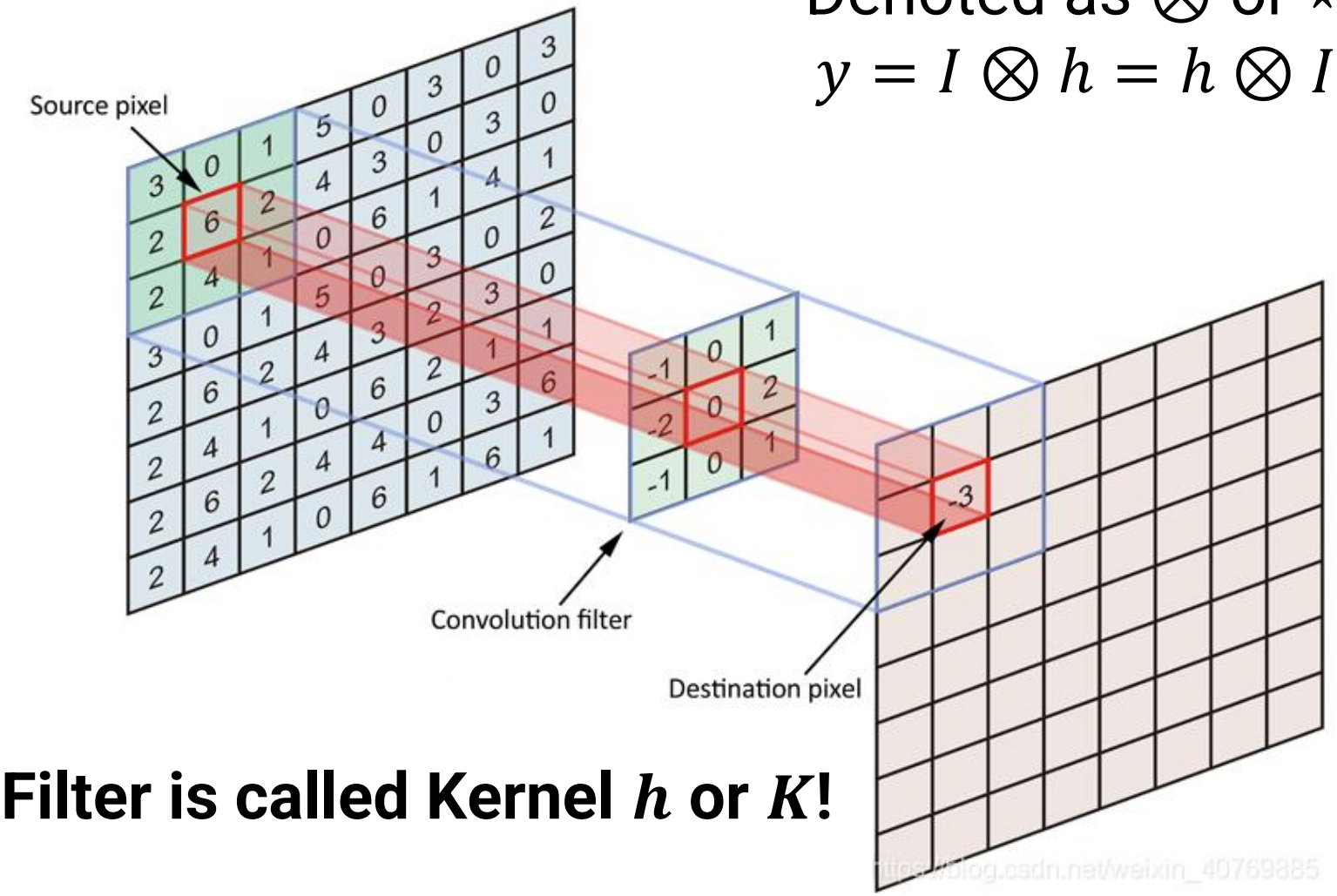
$$f(x,y) = \sum_{\forall x'} \sum_{\forall y'} T(x,y) \cdot P(x - x', y - y')$$

Flipping

Convolution

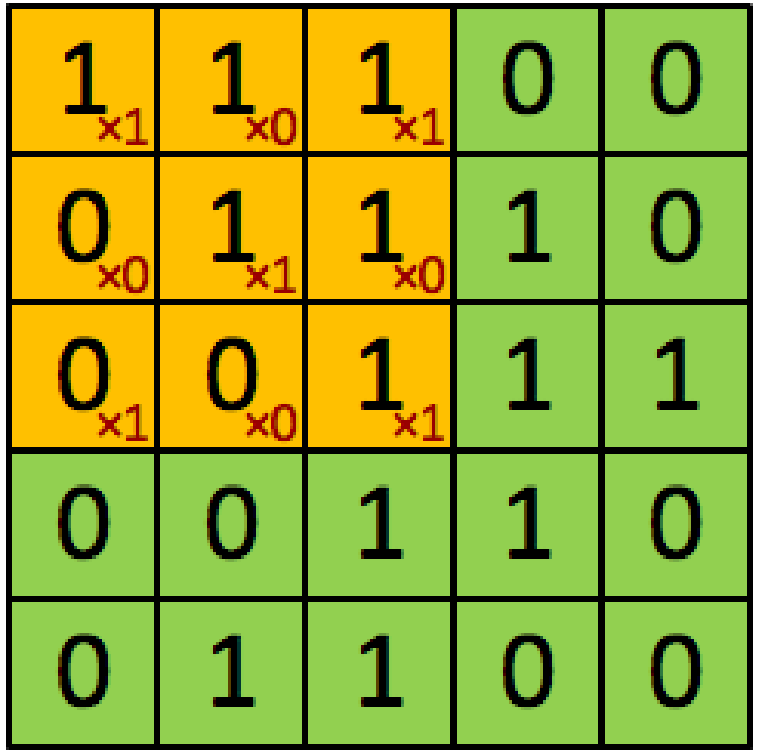
Denoted as $\otimes$ or $\star$

$$y = I \otimes h = h \otimes I$$

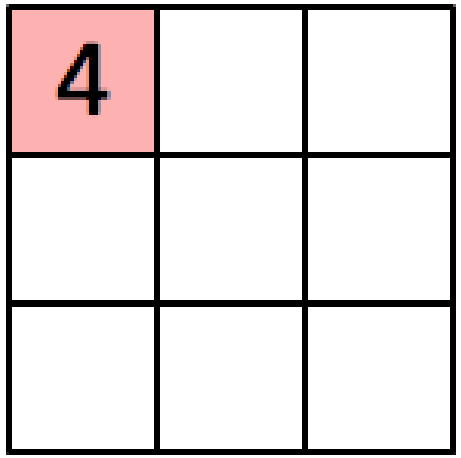


Filter is called Kernel h or K!

Recap



Image



Convolved Feature

Fun With Filtering

AKA Building Blocks of Photoshop

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

What does it do?

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Returns the original image!

What does it do?

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Gaussian Blurs the original image!

What does it do?

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpens the original image!

“Best” Thing About Convolution!

“Best” Thing About Convolution!

$$S_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

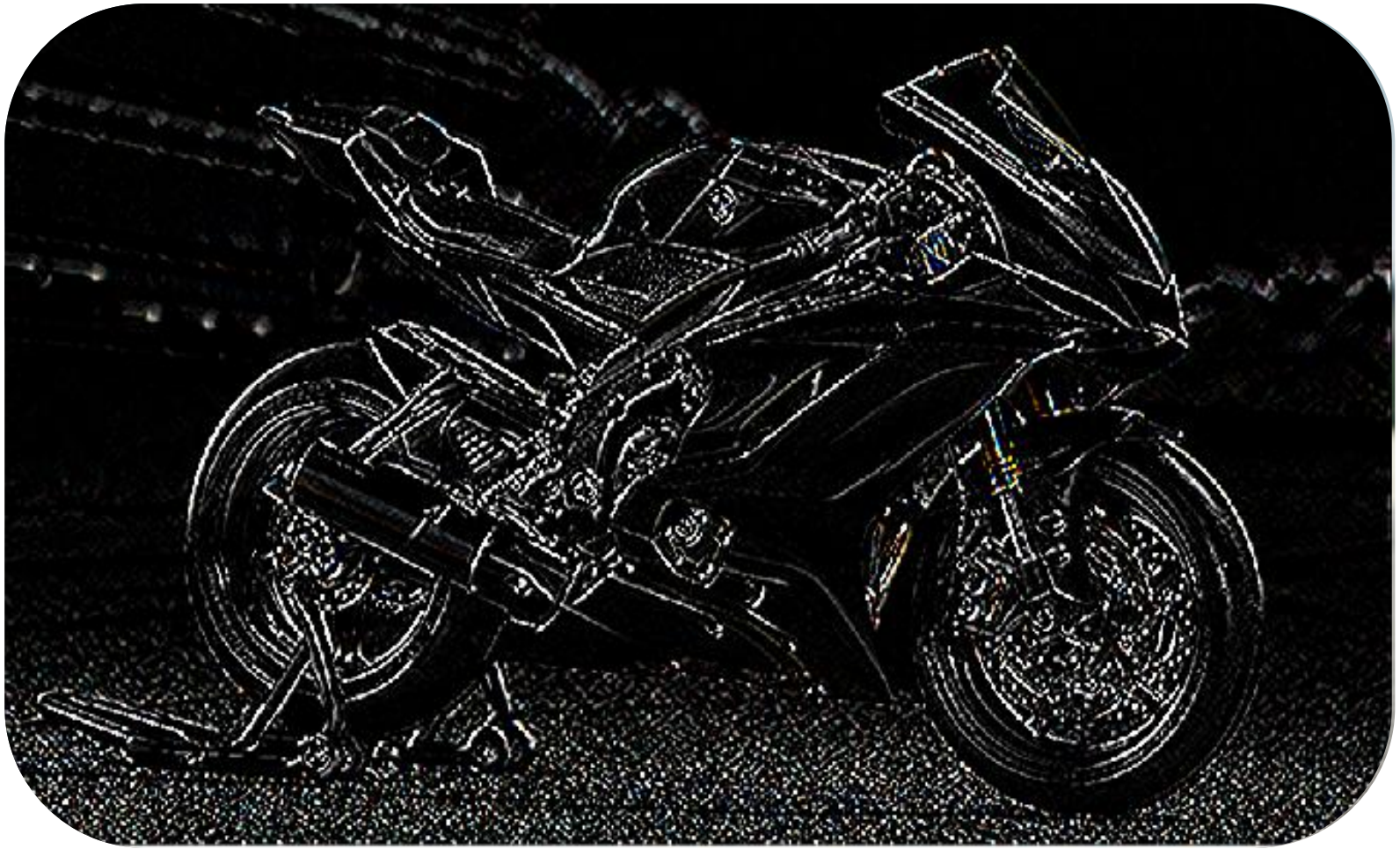
Horizontal Sobel

“Best” Thing About Convolution!

$$S_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Vertical Sobel

“Best” Thing About Convolution!



$$S_x \otimes S_y$$

Both Horizontal and Vertical Sobel

“Best” Thing About Convolution!

It is Associative!

So you can pre-combine and pre-compute filters!

$$I \otimes g \otimes h = I \otimes (g \otimes h) = I \otimes K$$

Recap

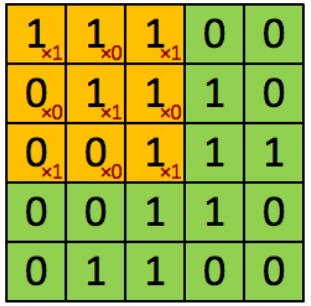
Convolution

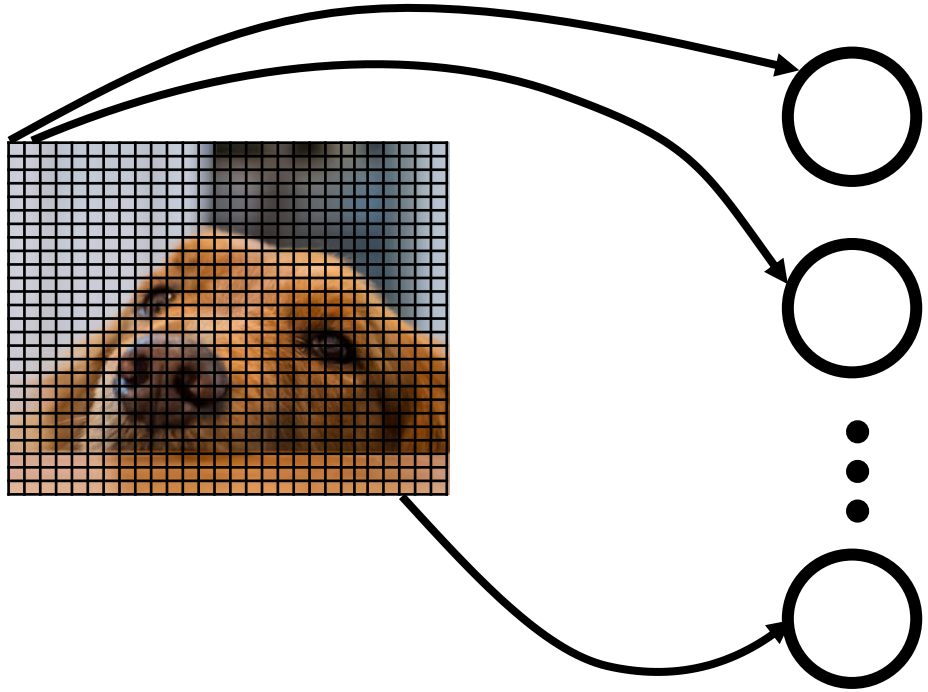
Convolution is the calculation of the area under the product of two signals and works as a filtering operation!

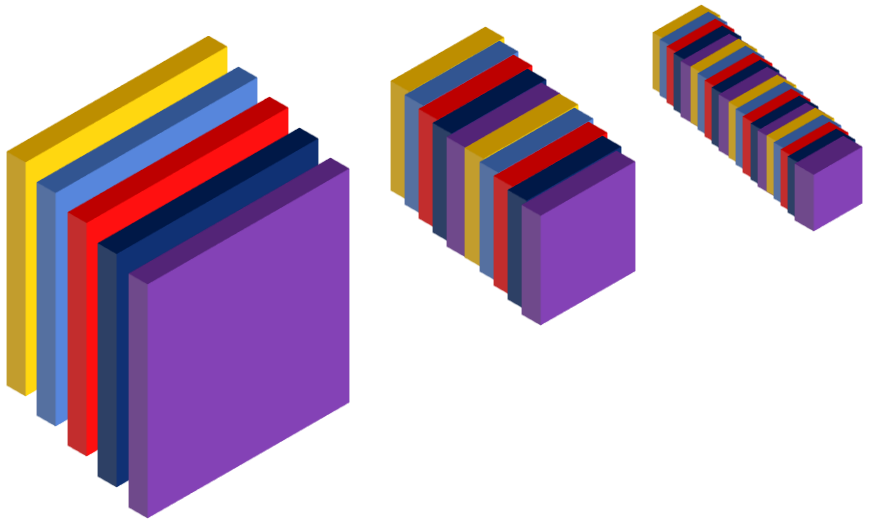
Correlation and Convolution.doc (umd.edu)

Dog Vs Cat Neural Network

A Better Way



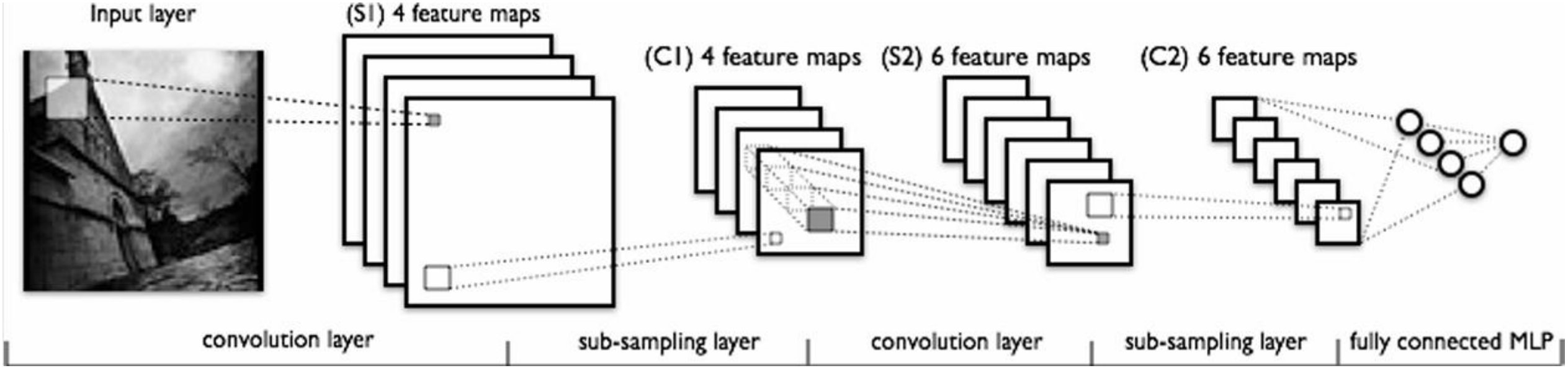




Convolutional Neural Network or ConvNets or CNN

LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

CNN Backprop



$y_1 = \sigma(x_i W_1)$

Regular MLP

Conv matrix

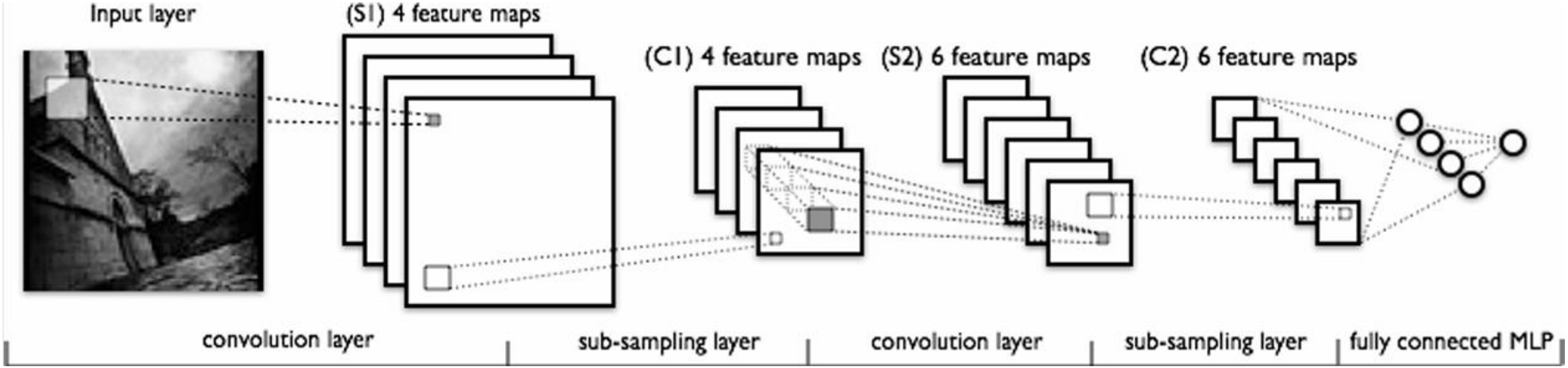
Conv kernel/stencil

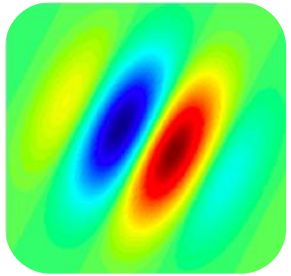
$y_1 = \sigma(x_i K_1) = \sigma(x_i \otimes k_1)$

Fancy-schamcy ConvNet

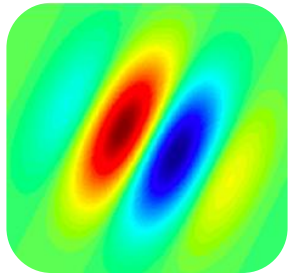
Slide adapted from UMD's CMSC764

CNN Backprop





Kx



$K^H x$

Conjugate or Hermitian Transpose

$$y = \sigma(\sigma(\sigma(x_i K_1)K_2)K_3)$$

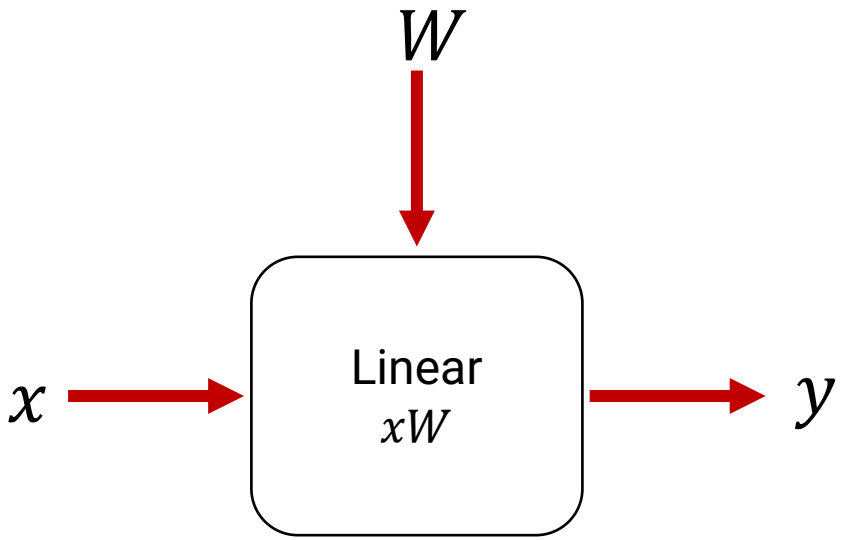
$$\nabla_{z_1}\mathcal{L} = \nabla_{z_3}\mathcal{L}K_3^T\sigma_2' K_2^T\sigma_1'$$

Adjoint of convolution (flip)

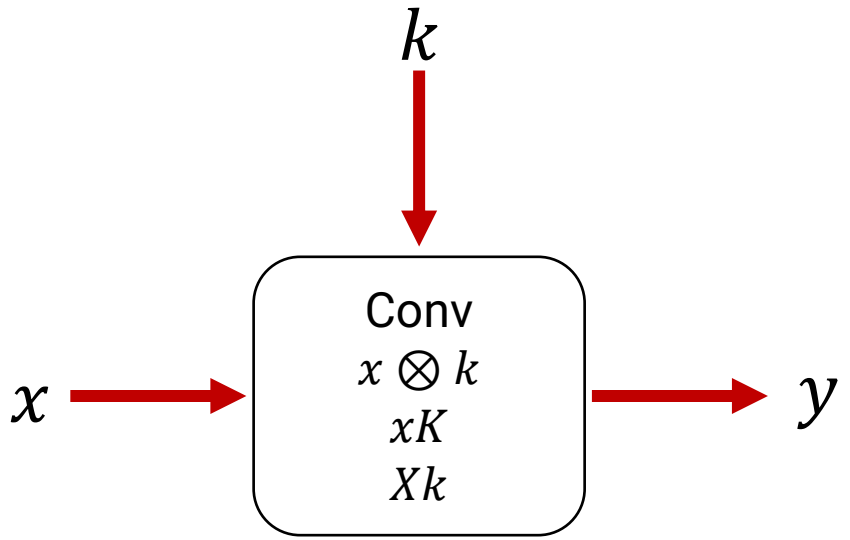
Remember $\nabla_x \sigma_i(xK_i) = \mathrm{diag}(\sigma_i')\, K_i^T$

Slide adapted from UMD's CMSC764

AutoGrad

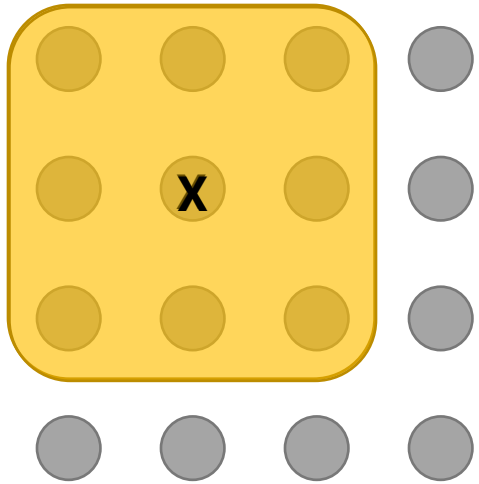


$$\nabla_x \mathcal{L}(xW) = \mathcal{L}'(xW)W^T$$
$$\nabla_W \mathcal{L}(xW) = x^T \mathcal{L}'(xW)$$



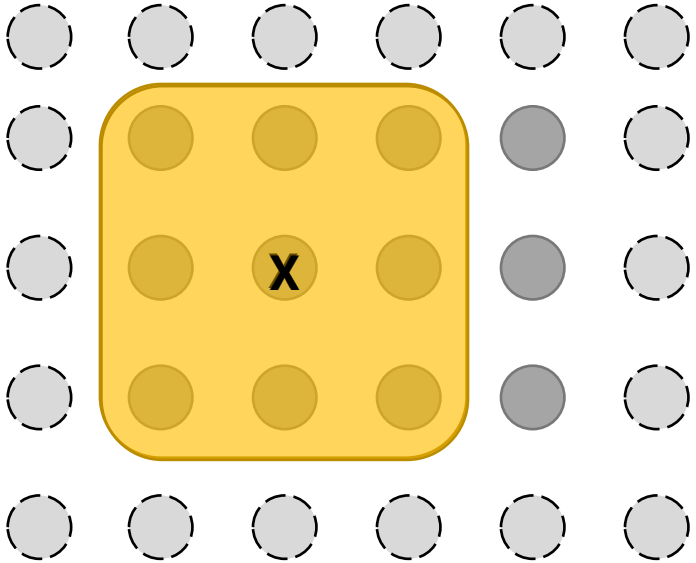
$$\nabla_x \mathcal{L}(xK) = \mathcal{L}'(xK)K^T$$
$$\nabla_W \mathcal{L}(Xk) = X^T \mathcal{L}'(Xk)$$

Convolution Variants



A simple 3×3 convolution produces feature map that is 2 units (px. if images) smaller on both directions!

Convolution Variants

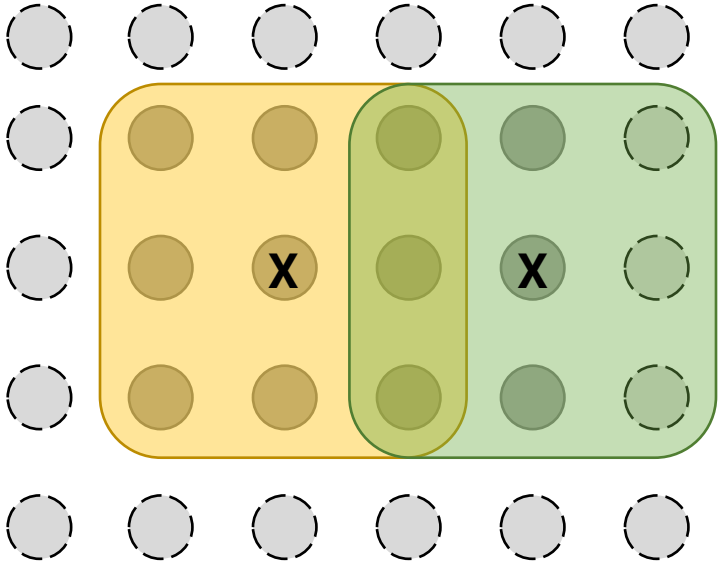


With Padding!
Output is same size as input!

What do you pad with?

- Zeros
- Reflect
- Copy Edge values!

Convolution Variants



Stride= N reduces dimension by $N!$ (factorial)

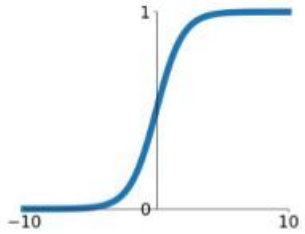
```
Output Size = (Input Size - Filter Size + 2 * Padding) / Stride + 1
```

What Is σ?

Activation Functions

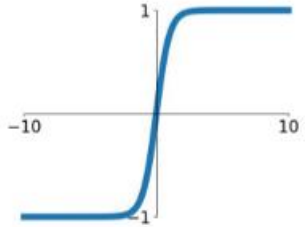
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



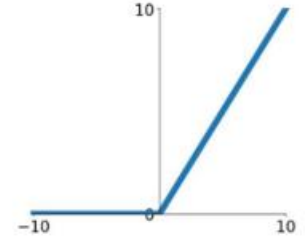
tanh

$$\tanh(x)$$



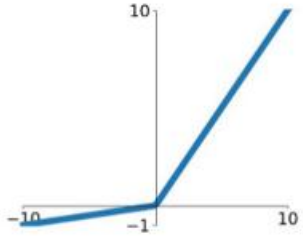
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

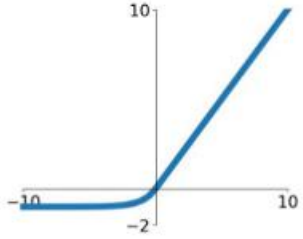


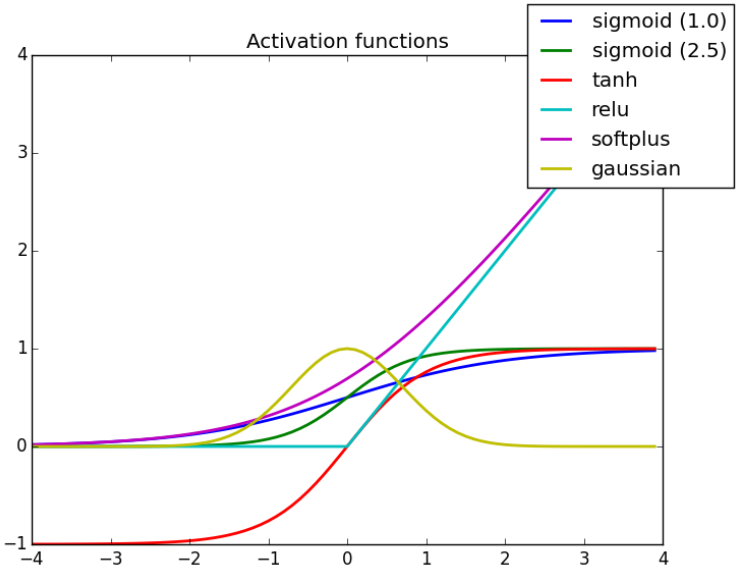
Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



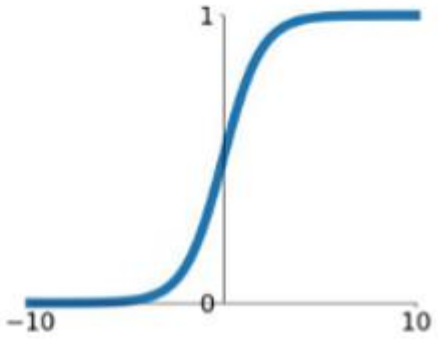


Quiz

Calculate the derivative of the sigmoid function

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

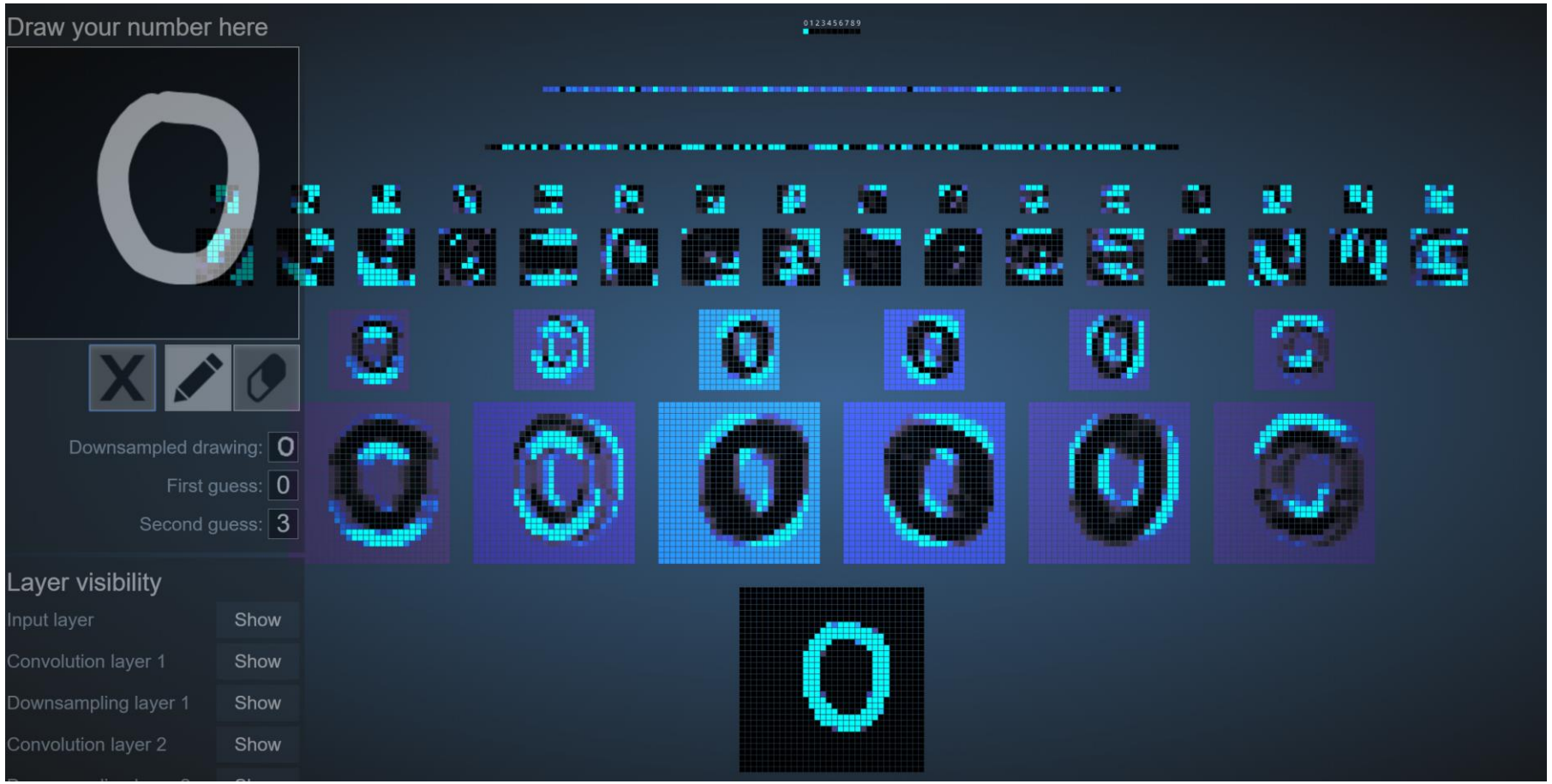


What Are Hyperparameters?

- Values which are chosen to design/train network!
- Can affect accuracy significantly, some more than others!
  - Learning Rate
  - Batch Size
  - Layers (Type, size, arrangement)
  - Filter Size
  - Padding
  - Number of filters/layers
  - Many more!

Dog Vs Cat Neural Network

What do CNNs see?



http://scs.ryerson.ca/~aharley/vis/conv/flat.html

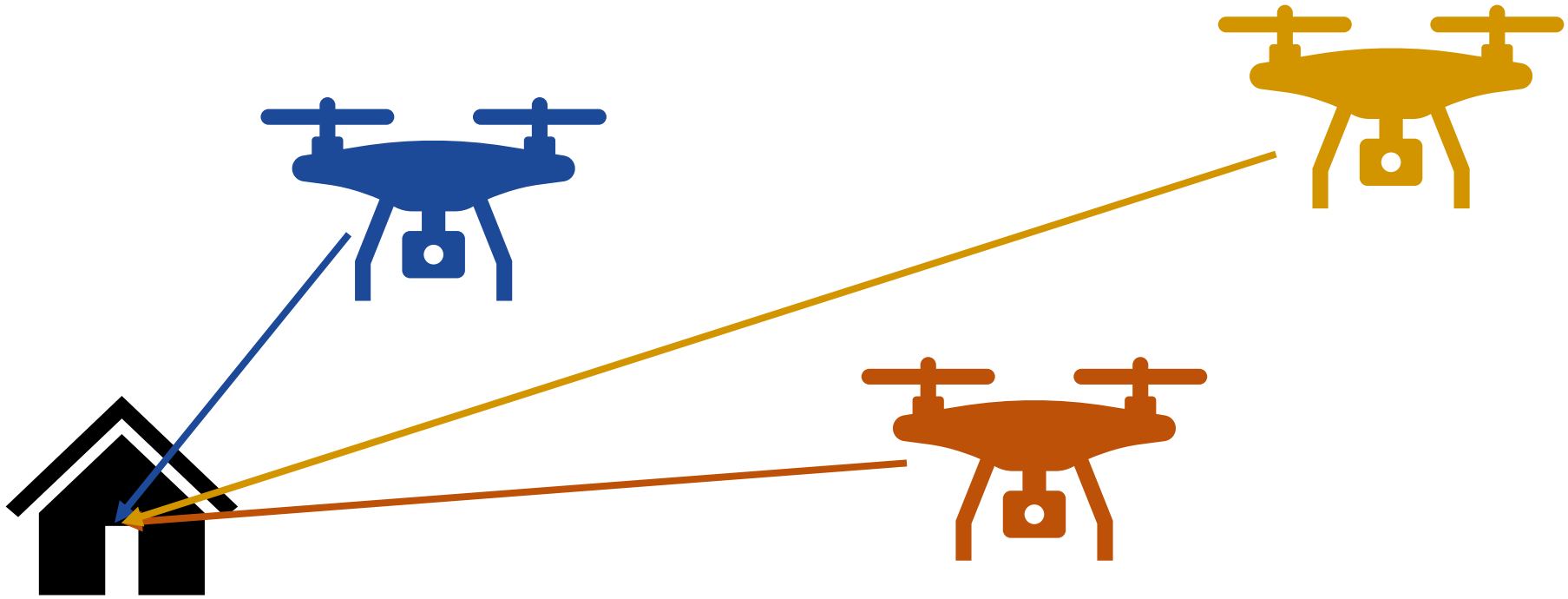
Harley, Adam W. "An interactive node-link visualization of convolutional neural networks." *International Symposium on Visual Computing*. Springer, Cham, 2015.

Classification Is All Good! What About Regression?

Regression Vs Classification?

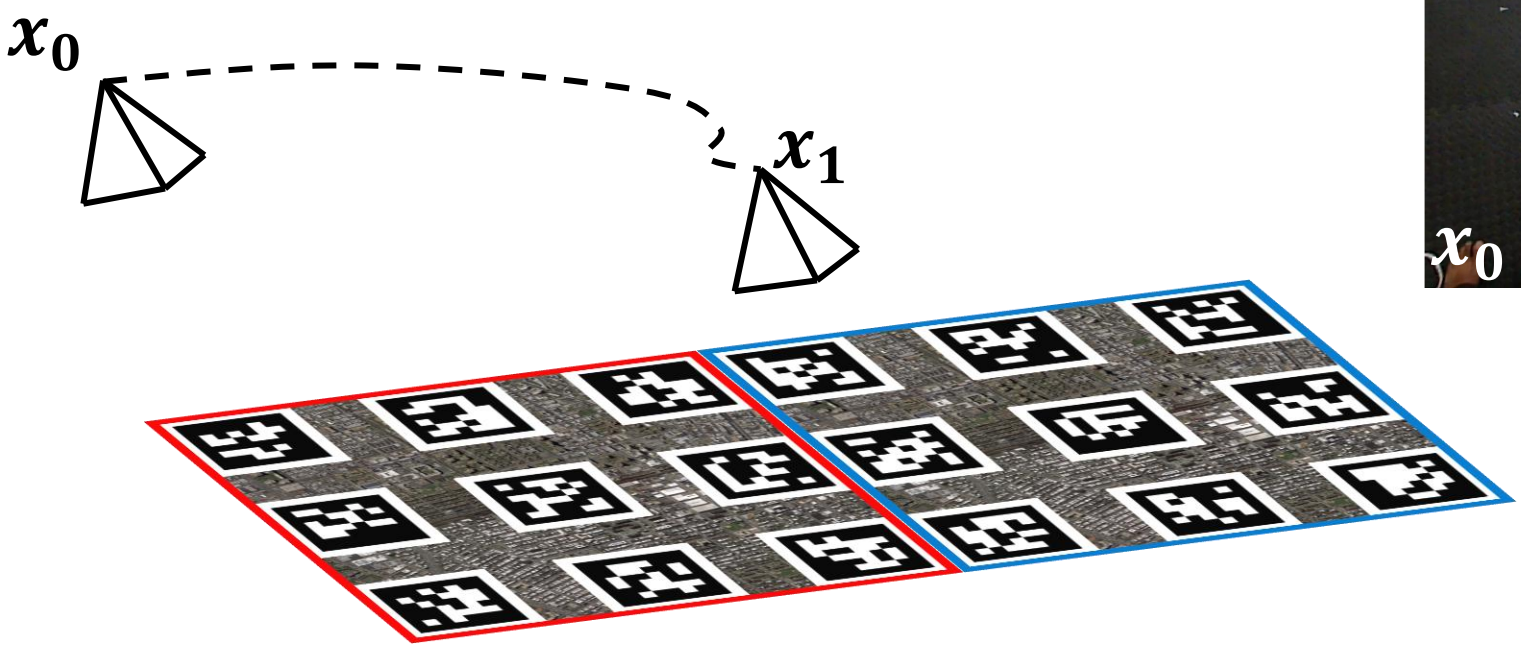
- Regression: Continuous Value (predict amount of throttle of a car)
- Classification: Discrete Values (predict whether to accelerate or brake)

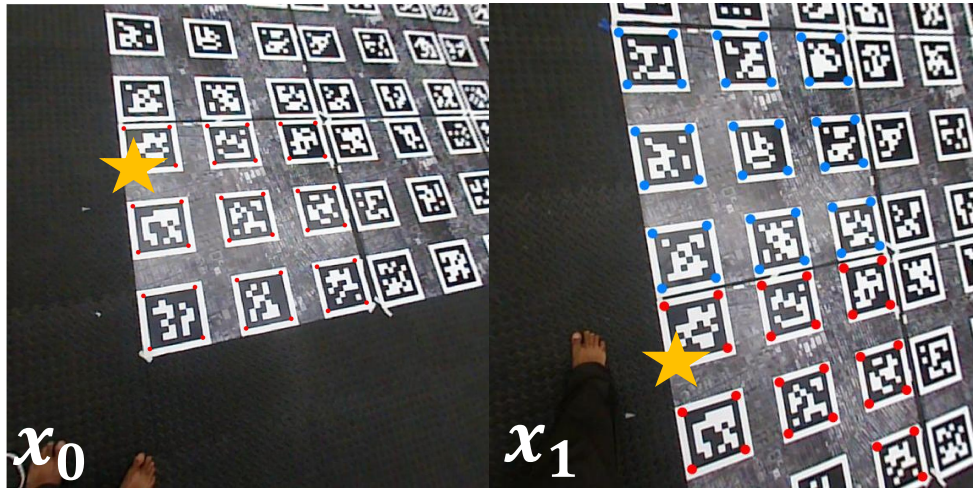
Homing Vector Estimation

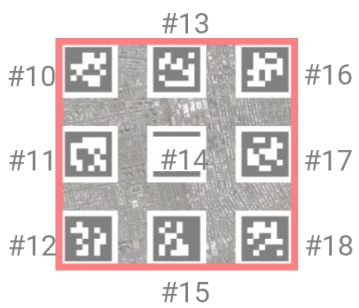


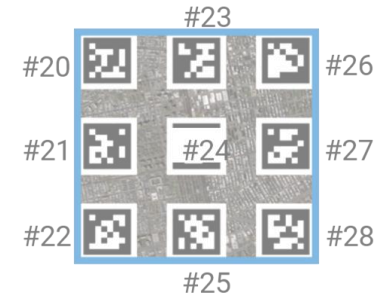
Direction vector pointing from to the “home direction”, something like a compass.

Homing Vector Estimation

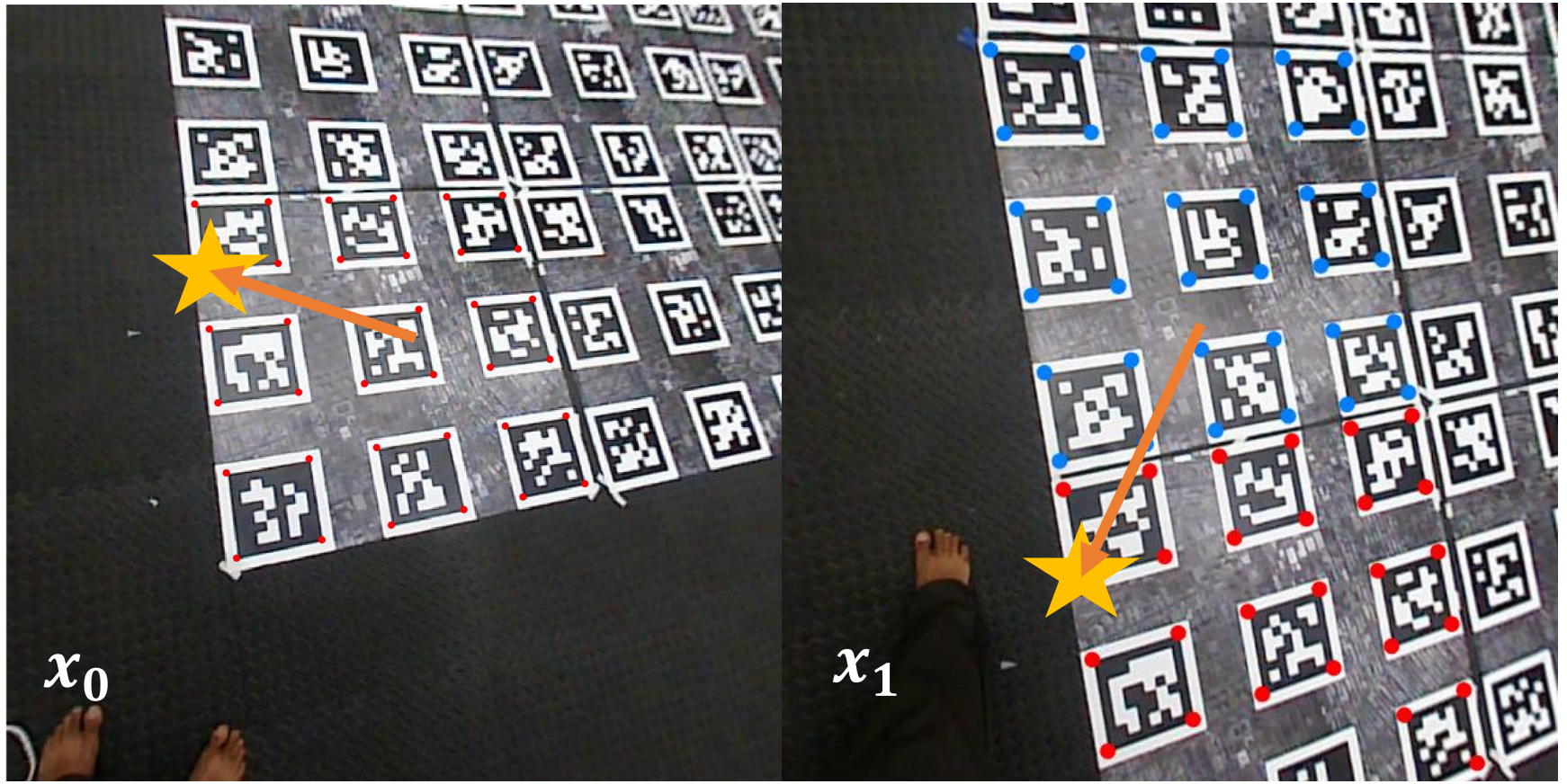




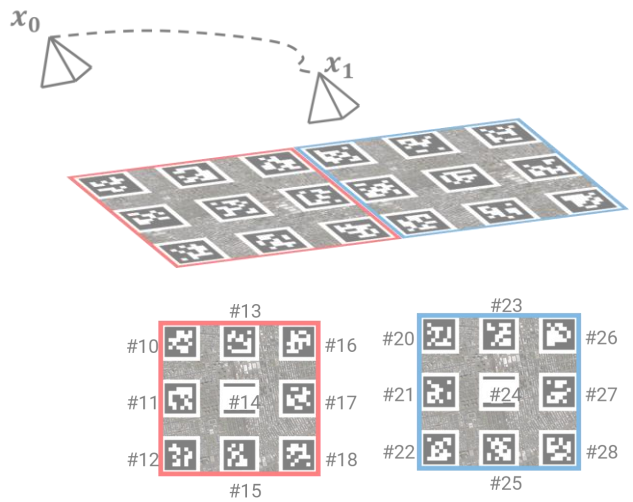




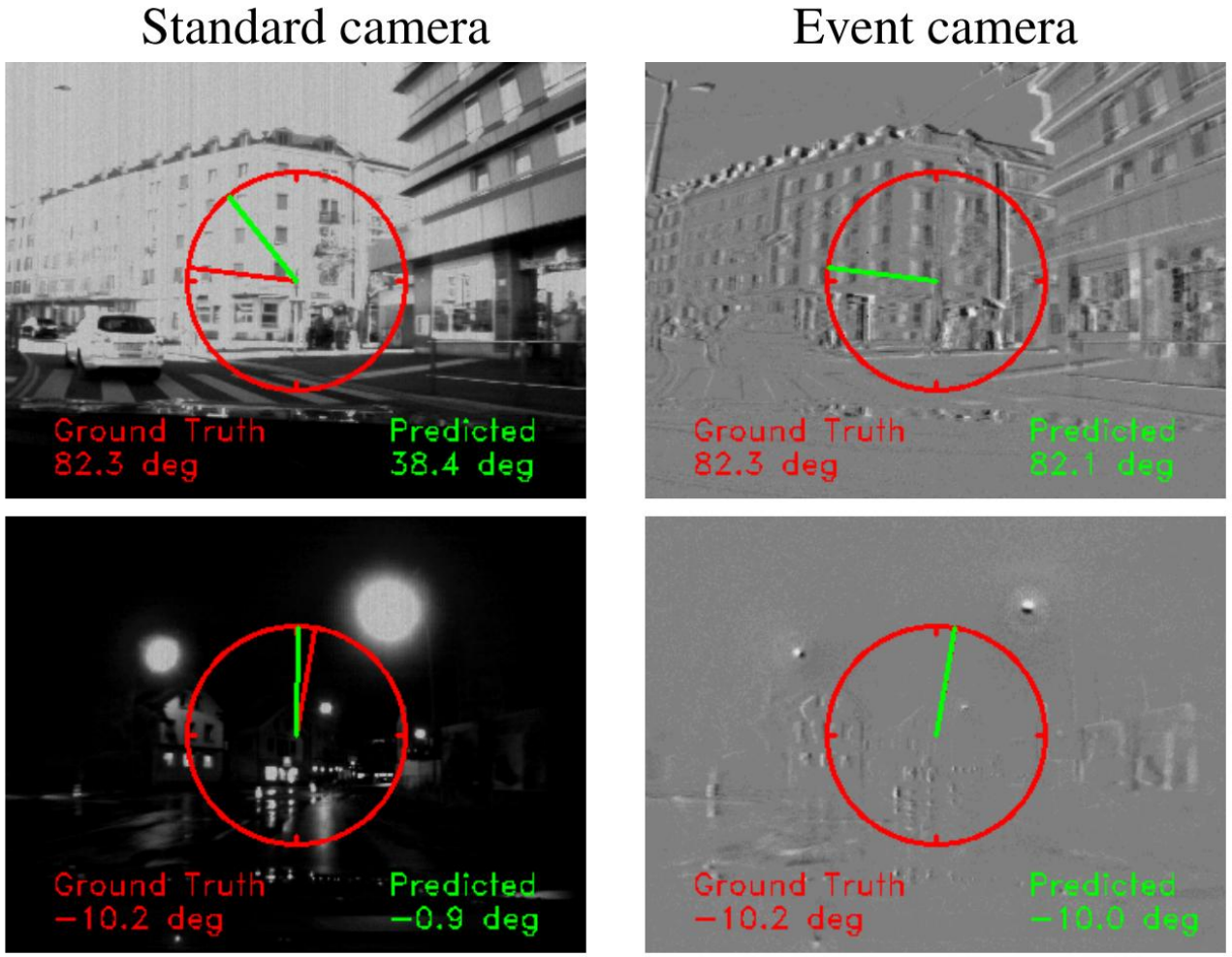
Homing Vector Estimation



Direction vector pointing from center of the image to the "home"!



Steering Angle Prediction



Maqueda, Ana I., Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. "Event-based vision meets deep learning on steering prediction for self-driving cars." IEEE CVPR 2018.

Next Class!

Advanced CNN Architectures And Image Warping