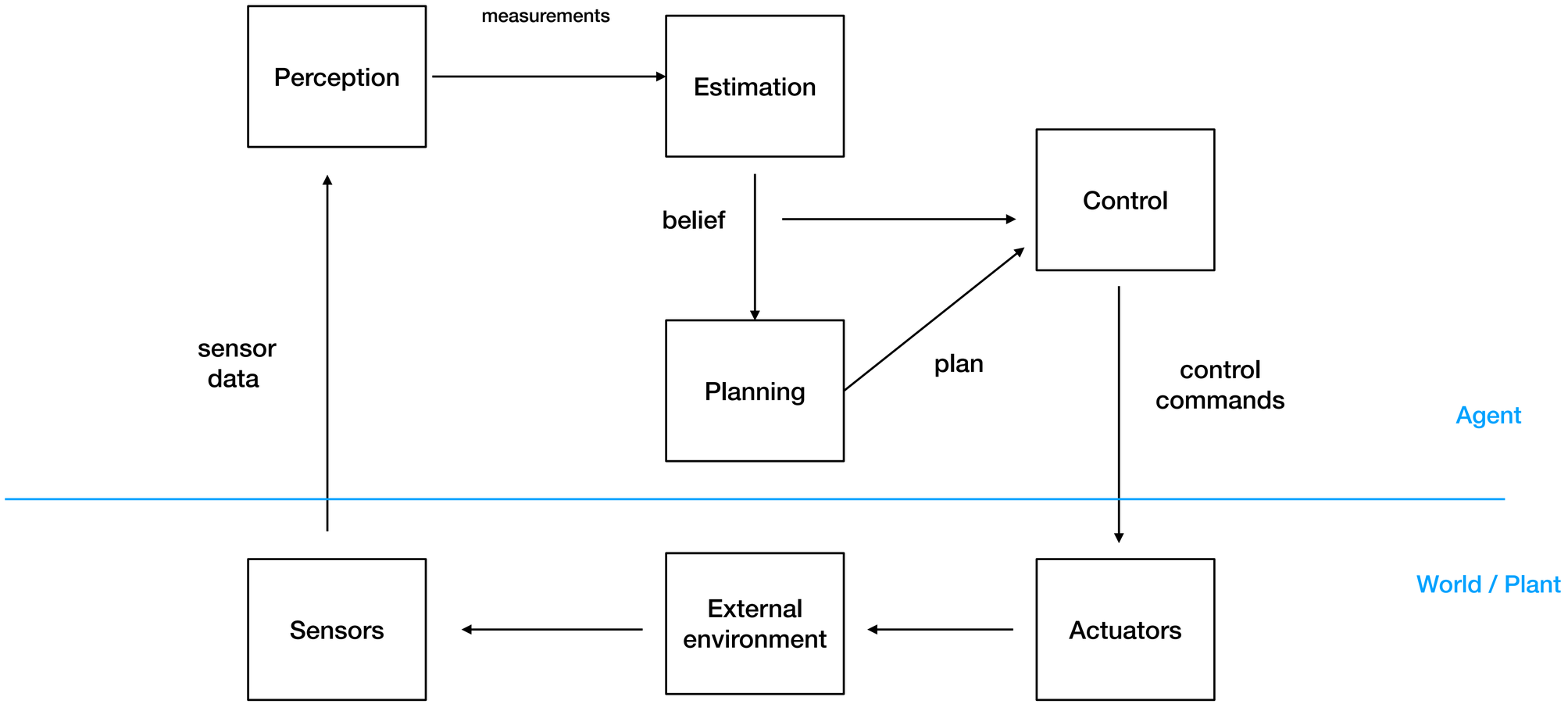
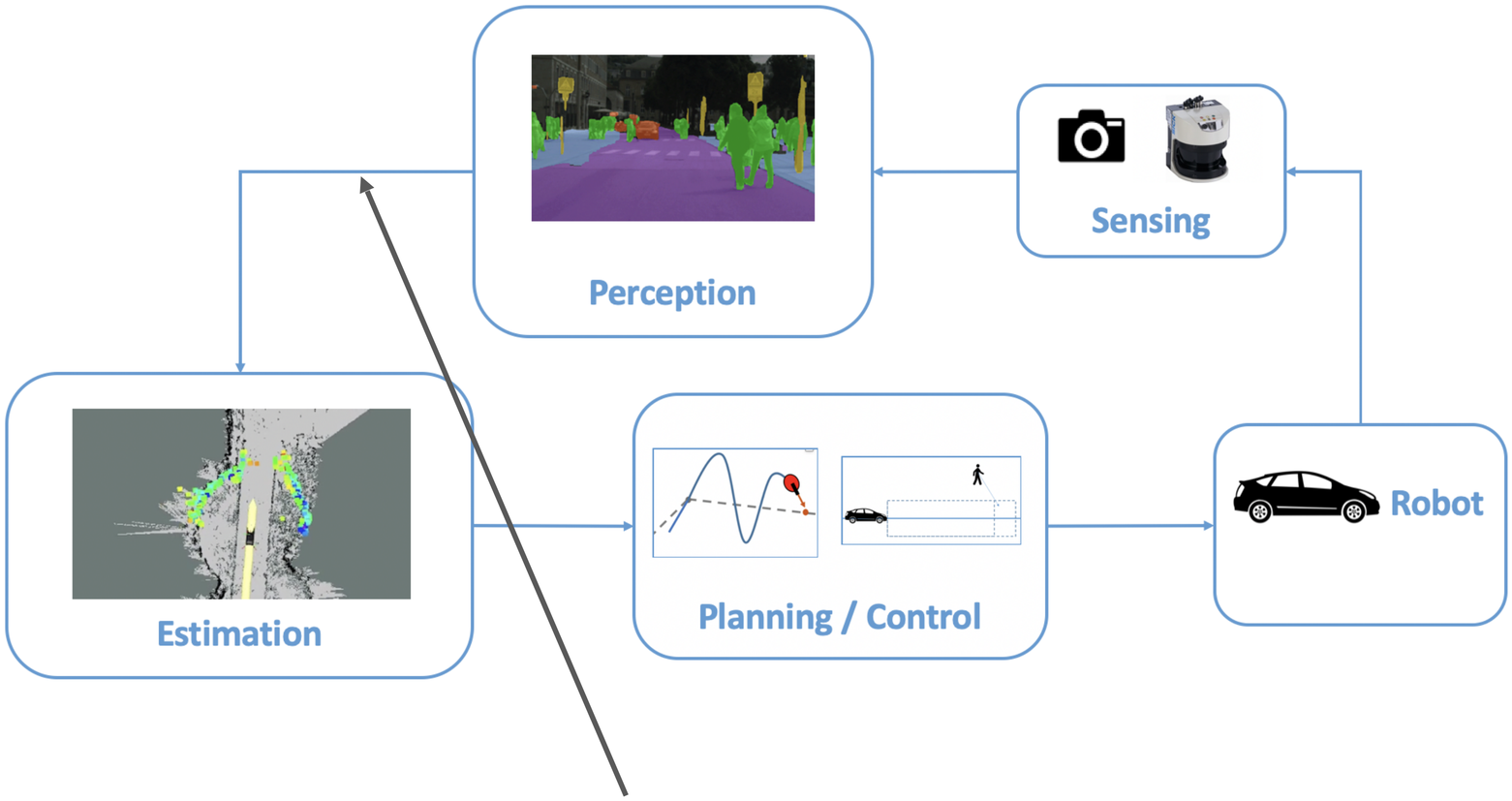
Advanced Visual Perception

Table of Contents


Perception
measurements
Estimation
Control
belief
Planning
plan
sensor
data
control
commands
Agent
Sensors
External
environment
Actuators
World / Plant

Perception as a sensor



$$p(\text{state}|\text{measurements}) = \frac{\text{measurement likelihood} * \text{prior}}{\text{evidence}}$$

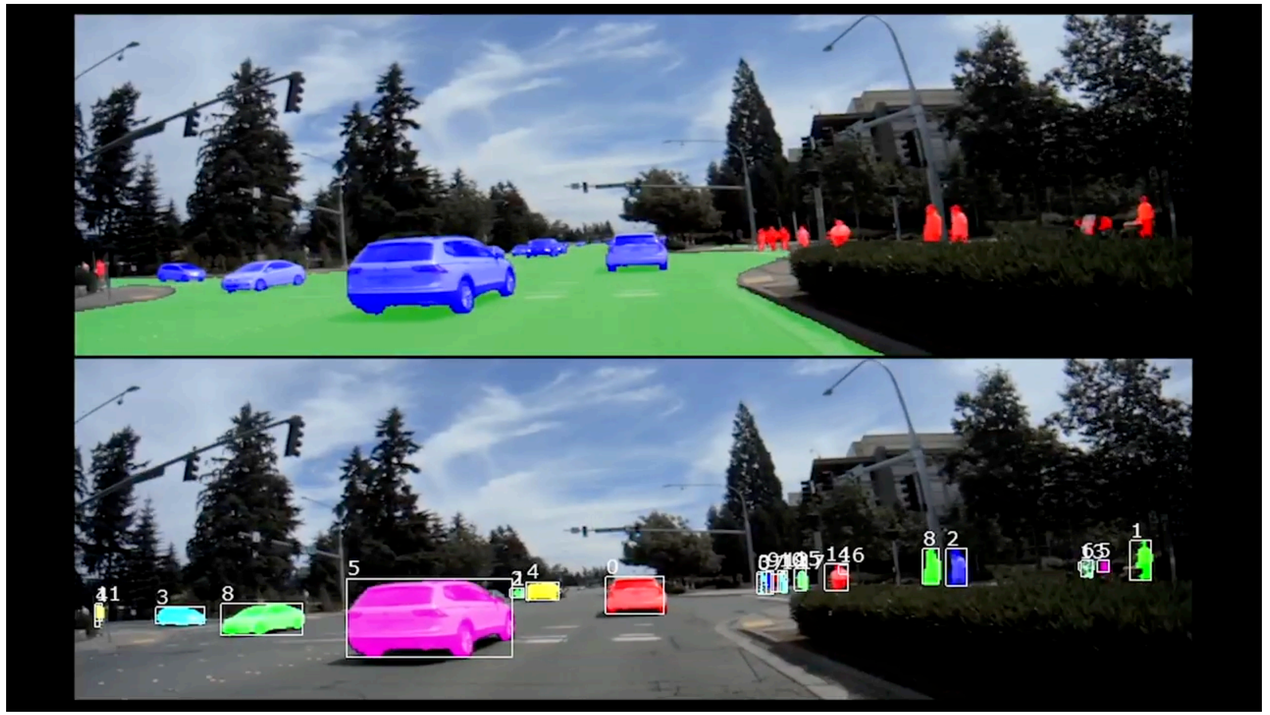

41
3
8
5
4
0
8
2
1

Image classification

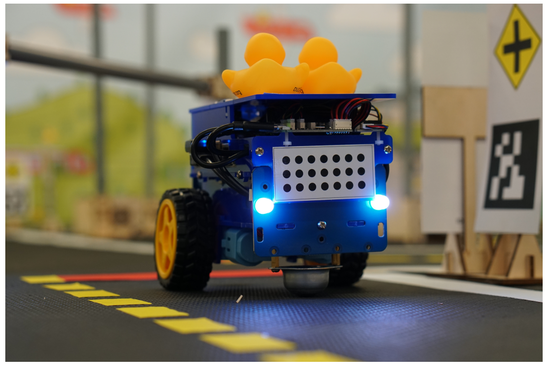
Duckie

Image segmentation

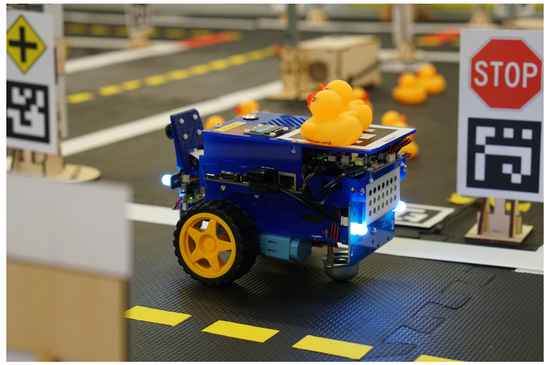
Object Detection

Instance Segmentation

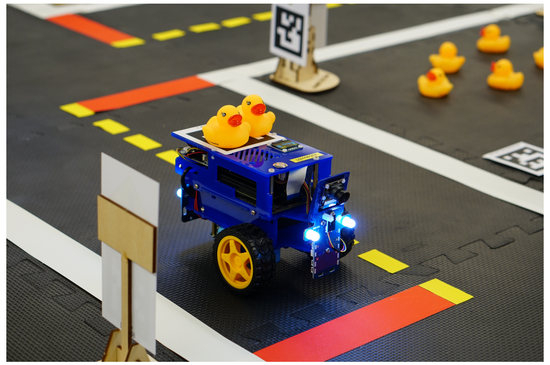
Classification Accuracy



Duckiebot



Duckiebot

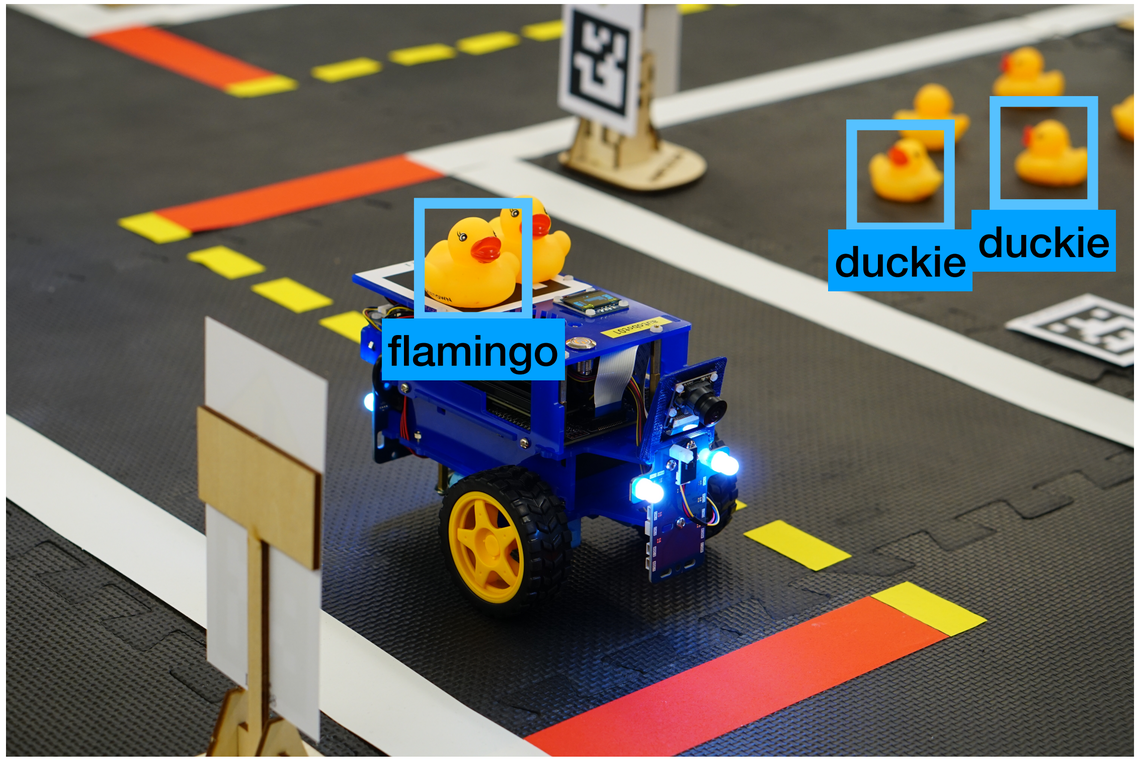


Duckiebot

Airplane

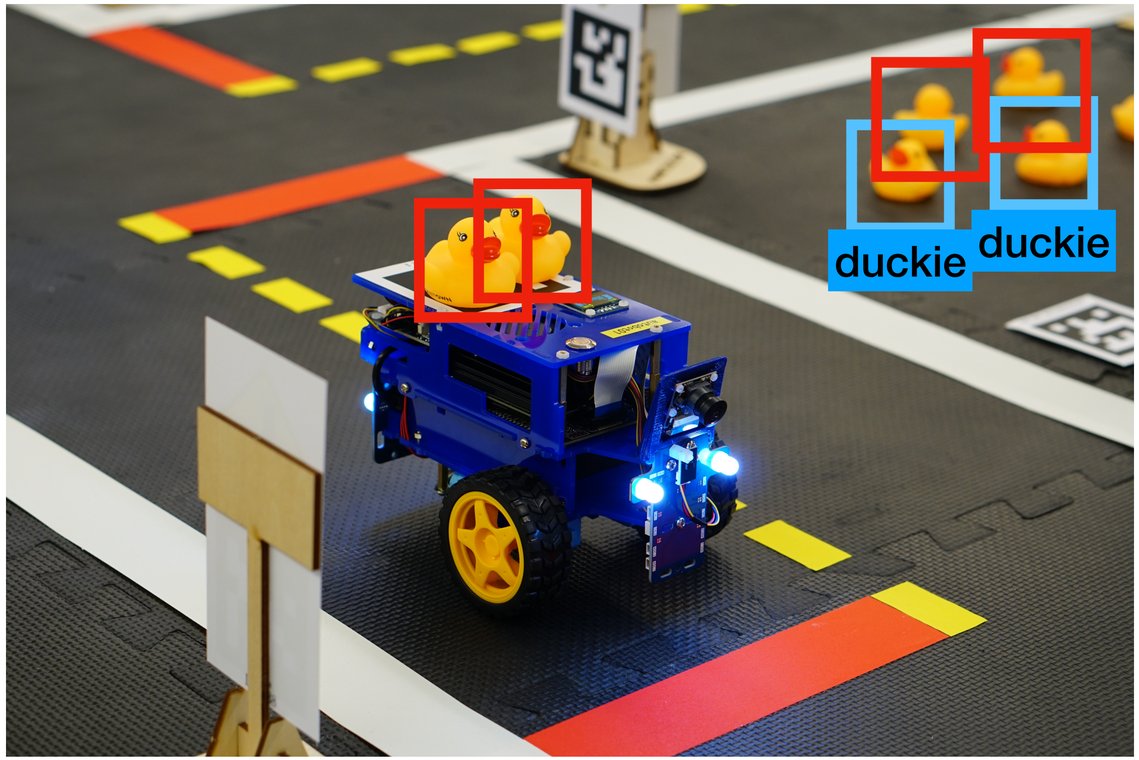
$$\text{Accuracy} = \frac{\text{\# correct}}{\text{\# total}} = \frac{3}{4}$$

Precision and Recall



$$\text{precision} = \frac{\text{\# True Positives}}{\text{\# True Positives + \# False Positives}} = \frac{2}{3}$$

Precision and Recall



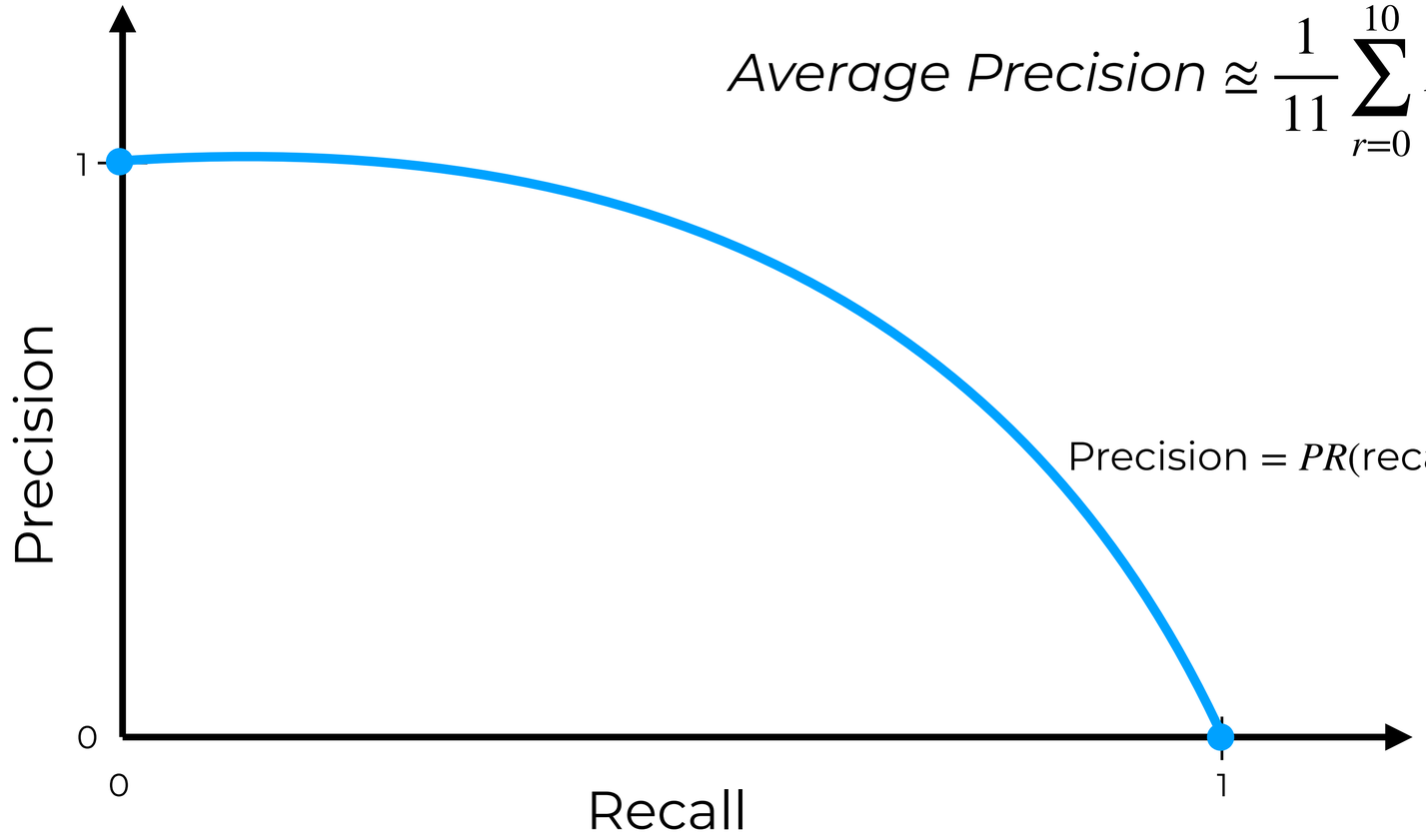
$$\text{precision} = \frac{\text{\# True Positives}}{\text{\# True Positives + \# False Positives}} = \frac{2}{3}$$

$$\text{recall} = \frac{\text{\# True Positives}}{\text{\# True Positives + \# False Negatives}} = \frac{2}{6}$$

Precision-Recall

$$\textit{Average Precision} = \int_0^1 PR(\text{recall}) d\text{recall}$$

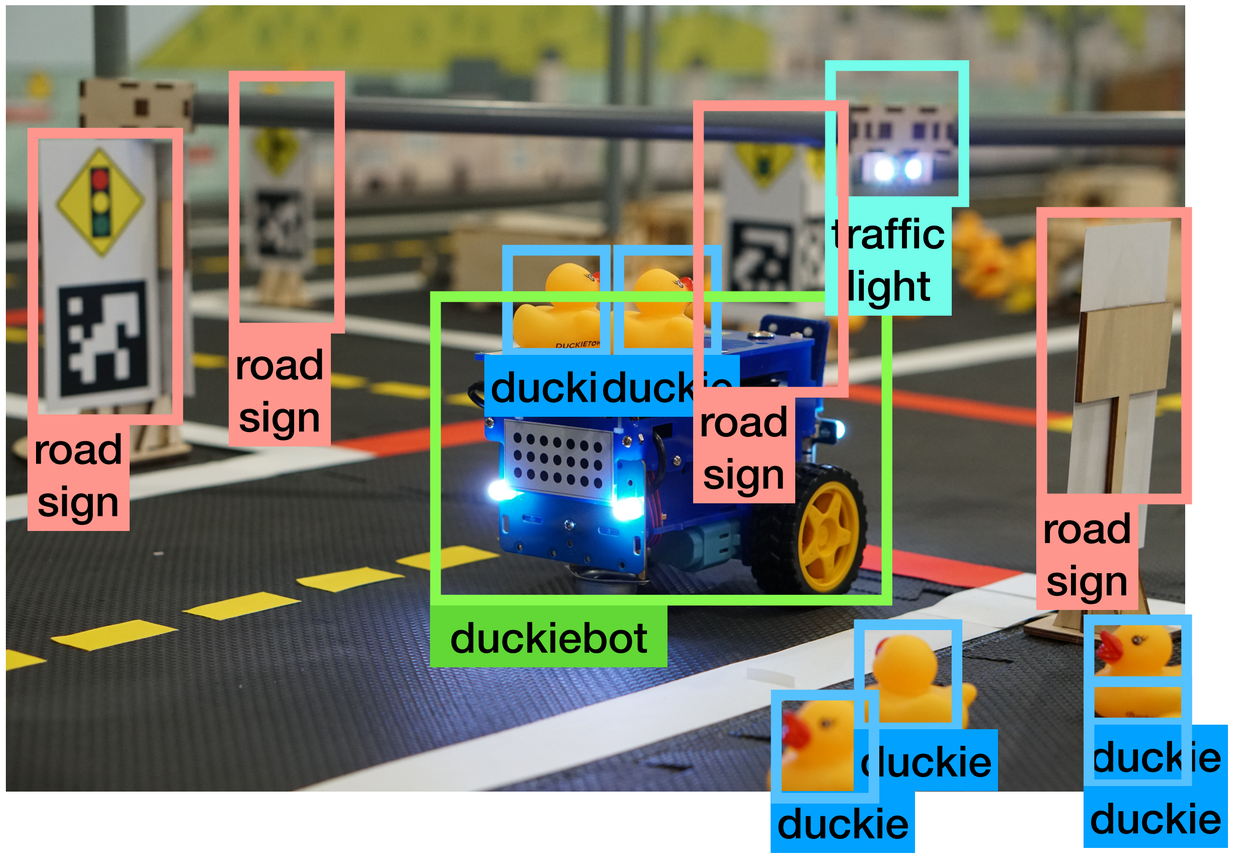
$$\textit{Average Precision} \approx \frac{1}{11} \sum_{r=0}^{10} PR(0.1 * r)$$



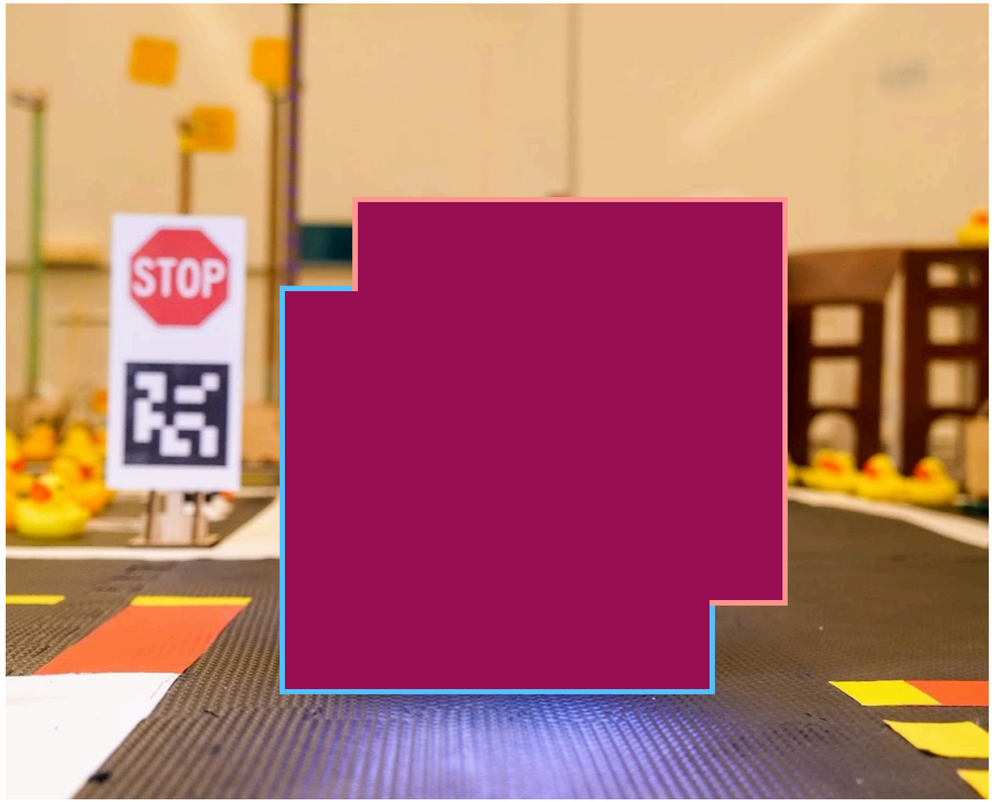
mean Average Precision (mAP)

- mean over all classes

$$mAP = \frac{1}{C} \sum_{c=1}^{C} AP_c$$

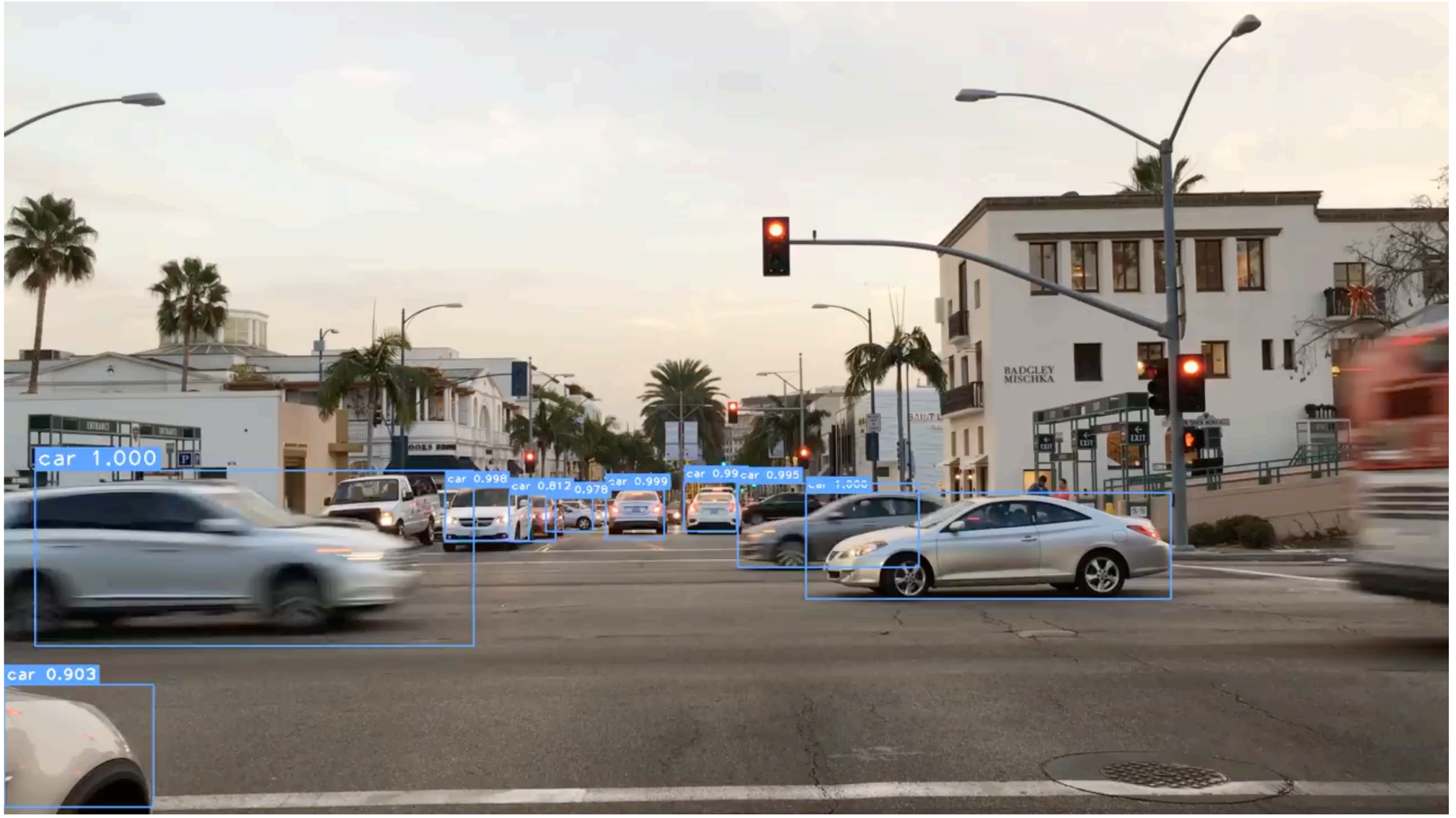


Intersection over Union



Annotated label

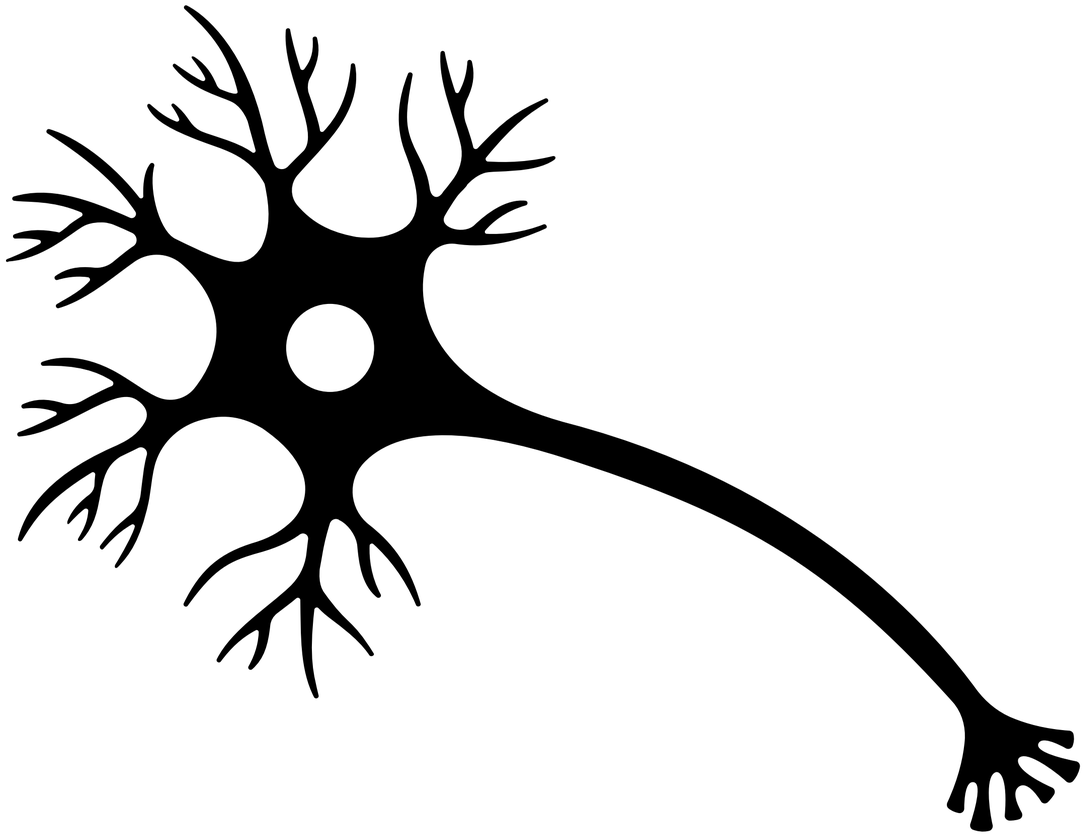
Model prediction


car 1.000
car 0.998
car 0.812
0.978
car 0.999
car 0.99
car 0.995
car 0.903
BADGLEY
MISCHKA

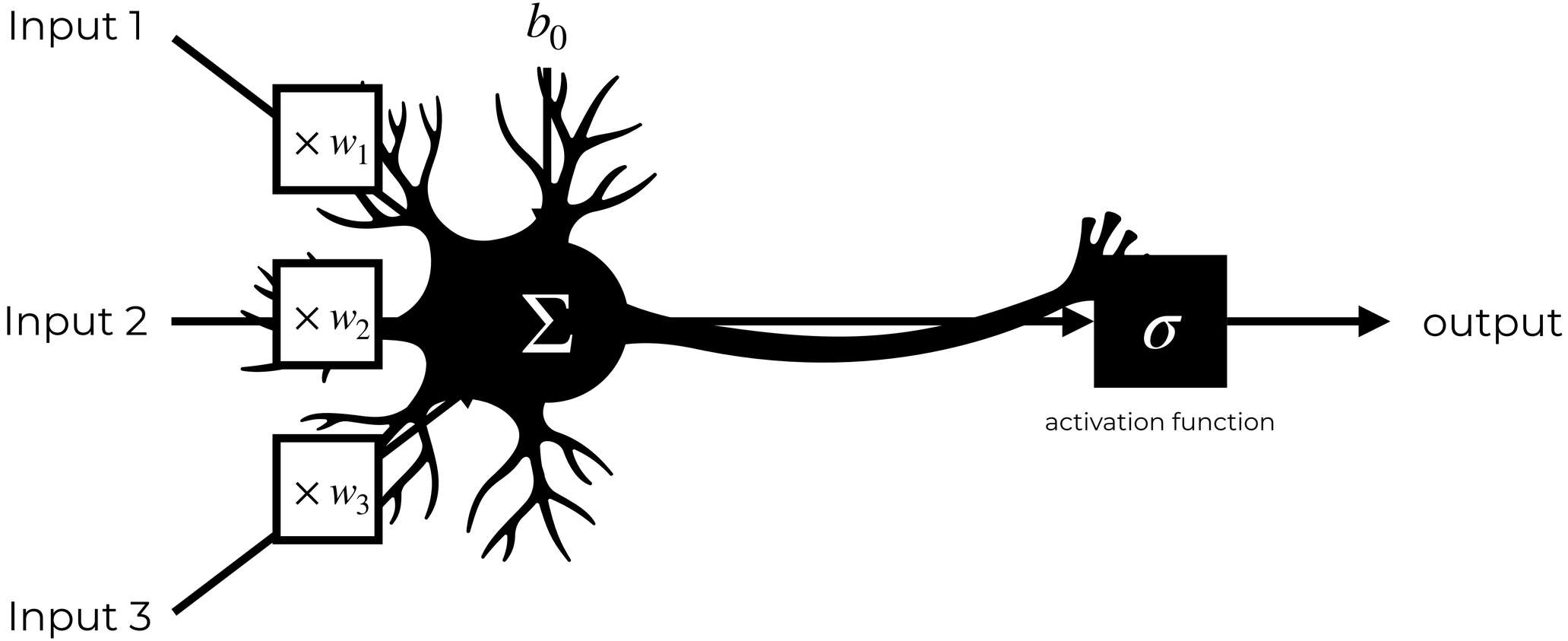
Advanced Visual Perception

Table of Contents

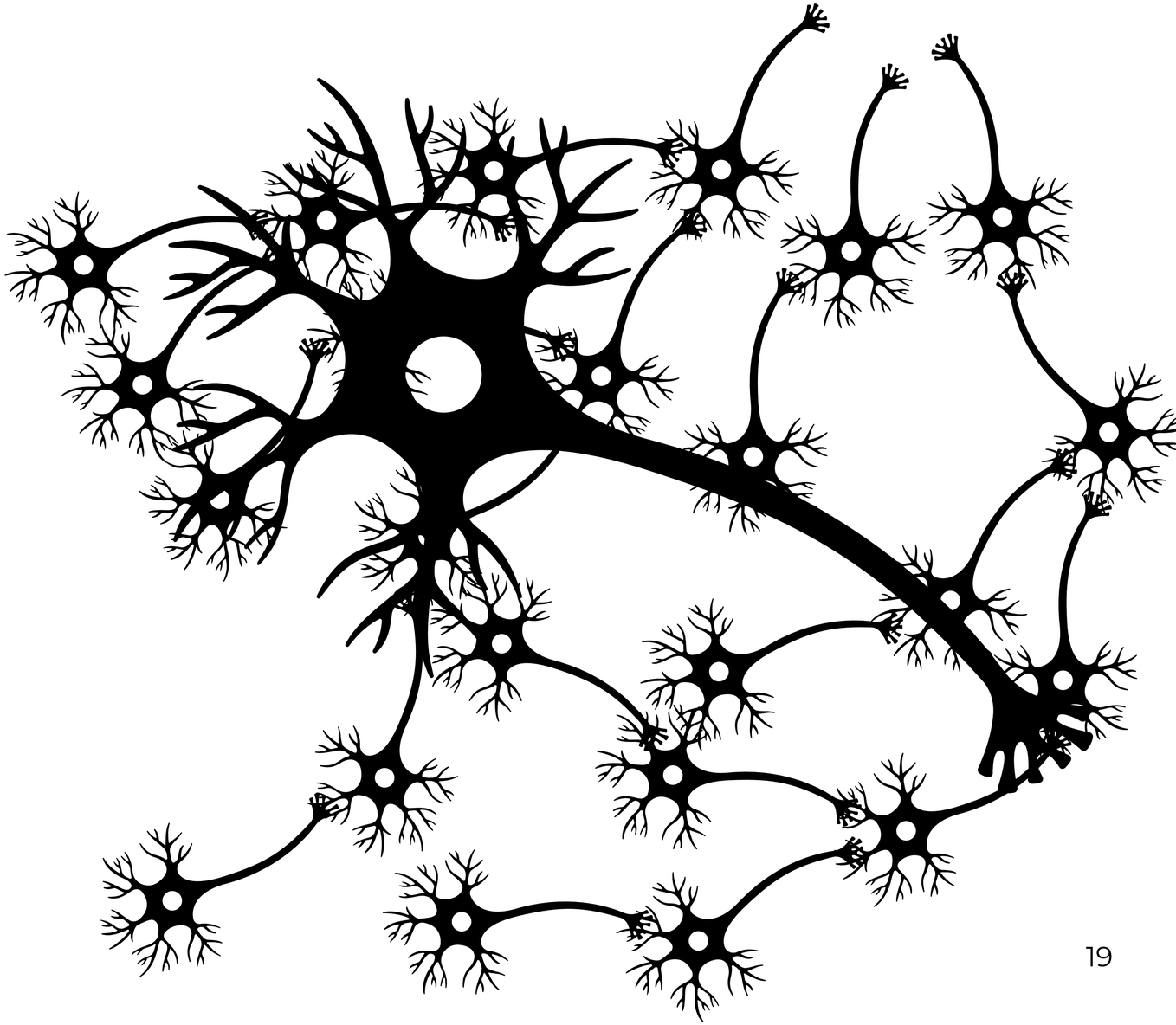
The neuron



The neuron

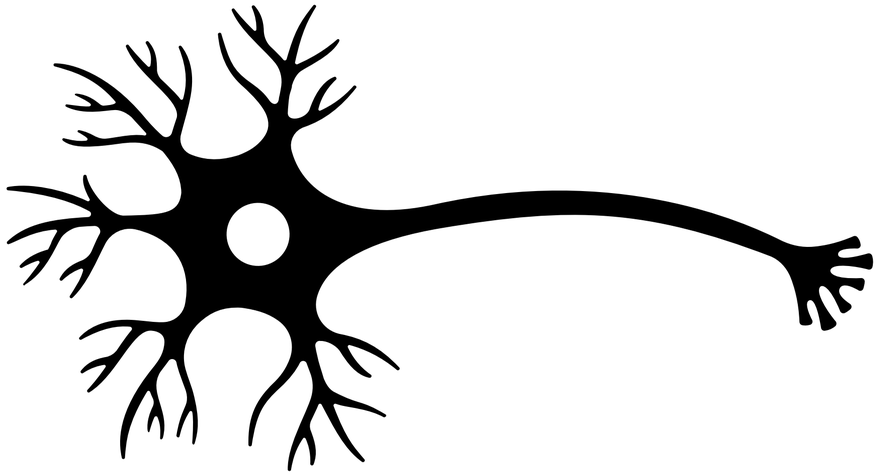

Input 1
Input 2
Input 3
b_0
$\times w_1$
$\times w_2$
$\times w_3$
Σ
σ
output
activation function

Compositionality



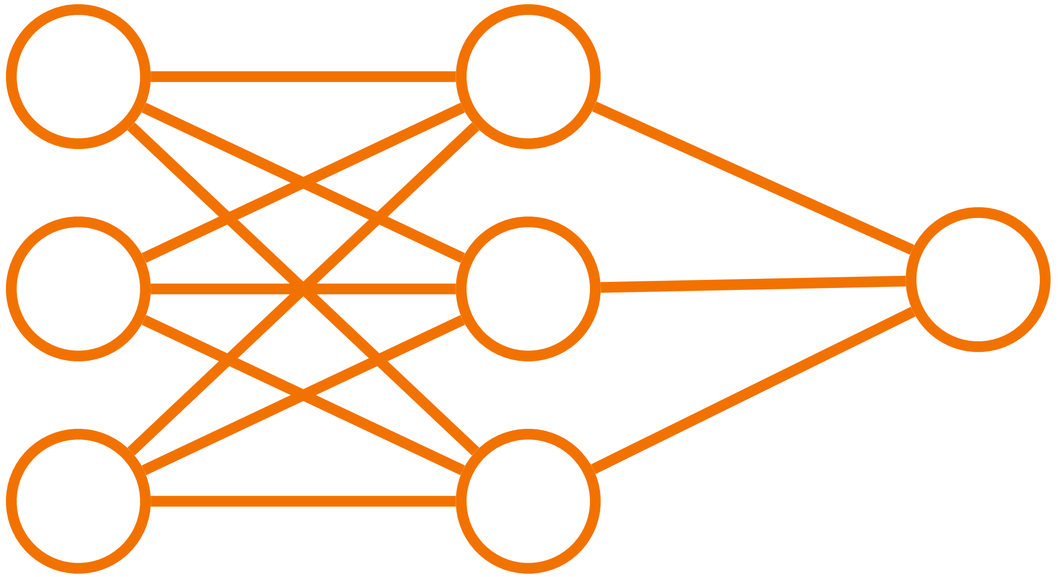
Differentiability

$$y = f(x)$$

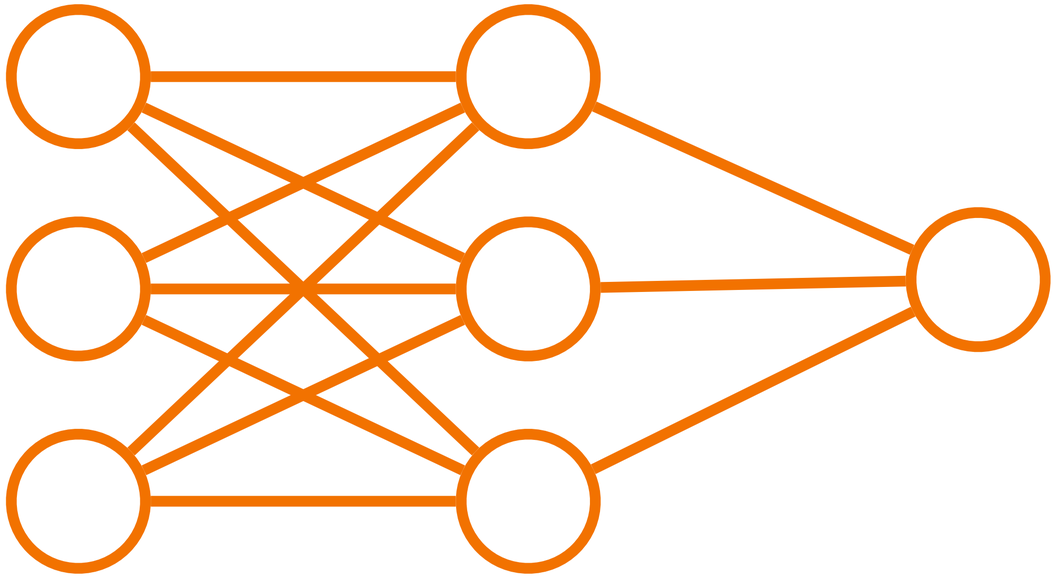


$$\frac{dy}{dx} = f'(x)$$

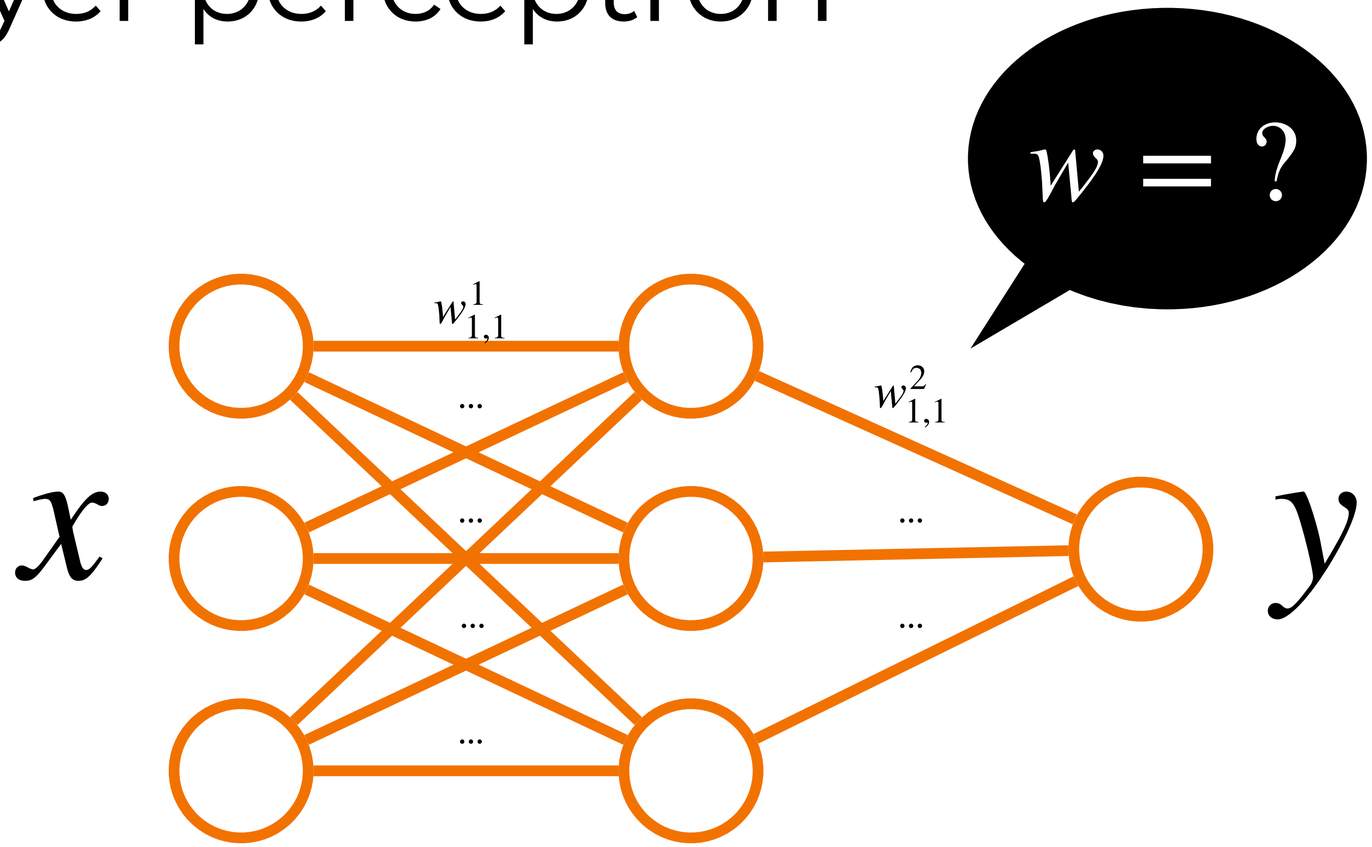

pos: (1.54, 0.00, 3.13), angle: -139.7 deg, steps: 1, speed: 0.00 m/s



Multi-layer perceptron



Multi-layer perceptron



$$y = f_w(x)$$

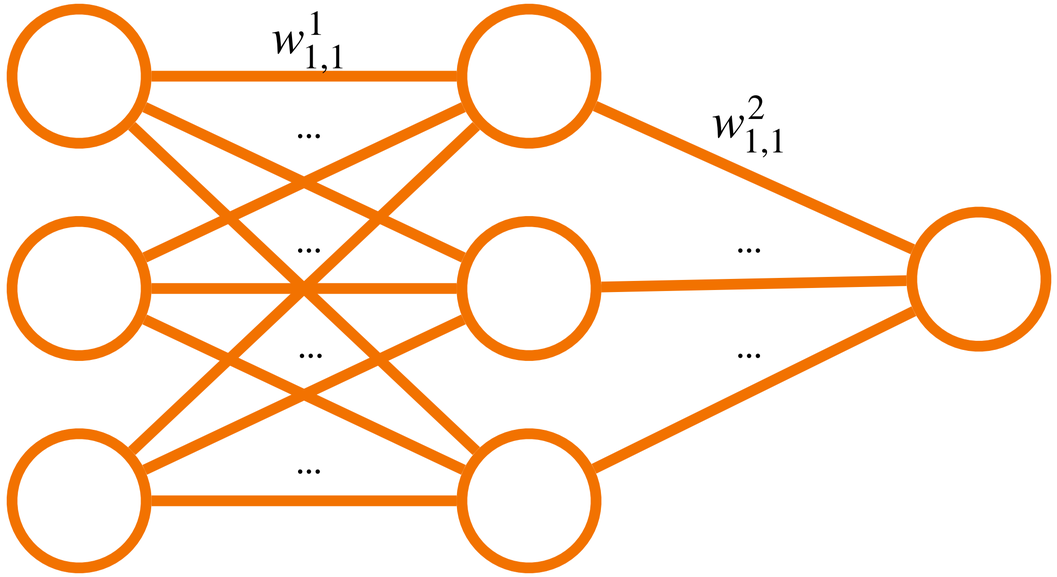
Annotated Dataset

Imagenet

- 14 million images

- thousands of categories

Loss functions



$f_w($  $)$

( , ✔)

$\mathscr{L}($, $)$

Loss functions

$$f_w(\text{  }) = \times \longrightarrow \mathcal{L} \uparrow$$

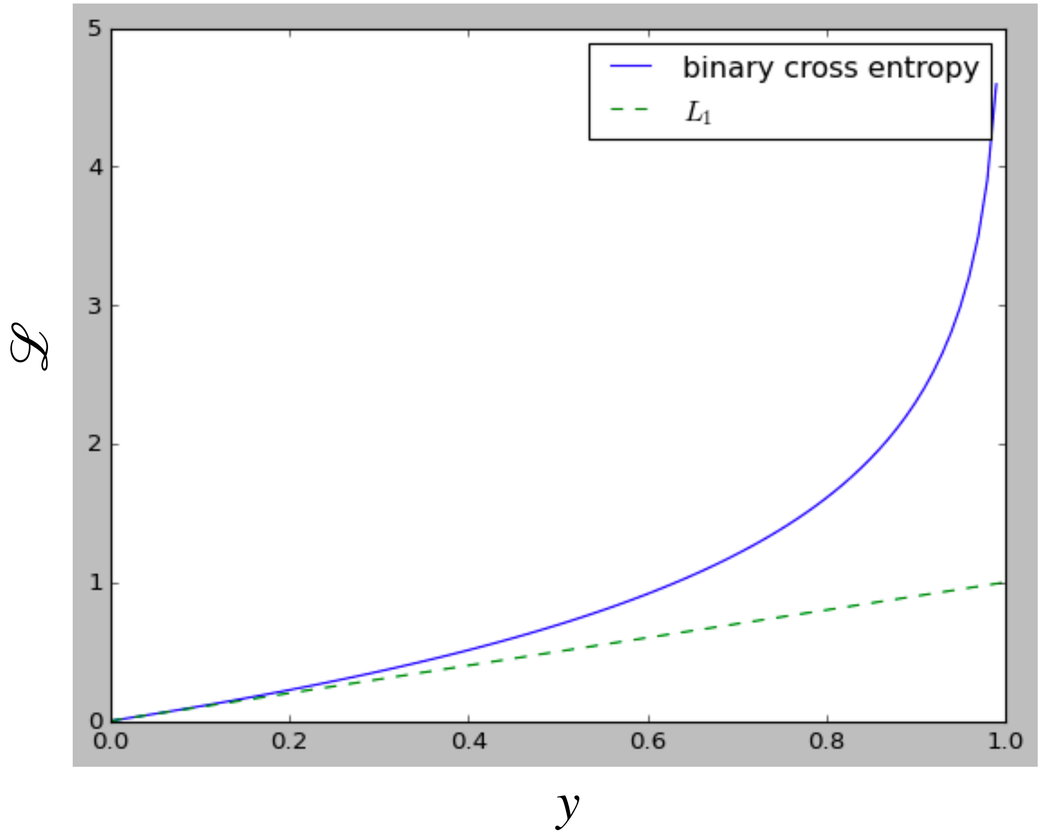
$$f_w(\text{  }) = \checkmark \longrightarrow \mathcal{L} \downarrow$$

Loss functions

The L_1 distance: $|y - y^*|$

The binary cross-entropy loss:

$$-(y^* \log(y) + (1 - y^*)\log(1 - y))$$

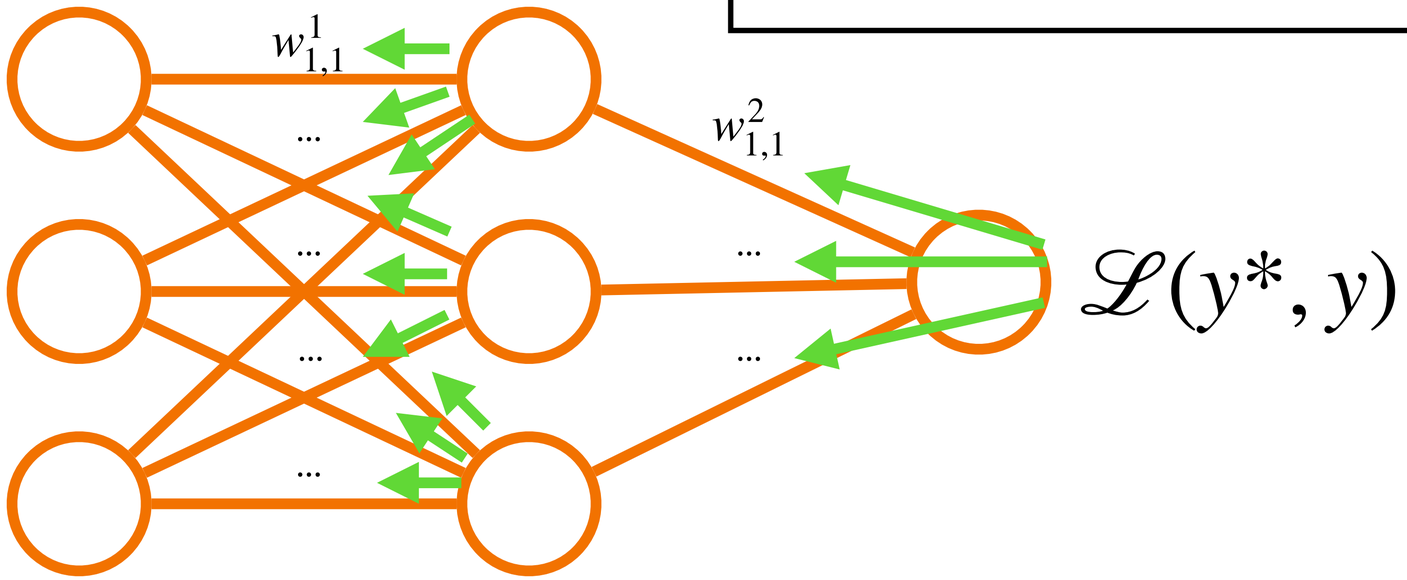


y^* : annotation

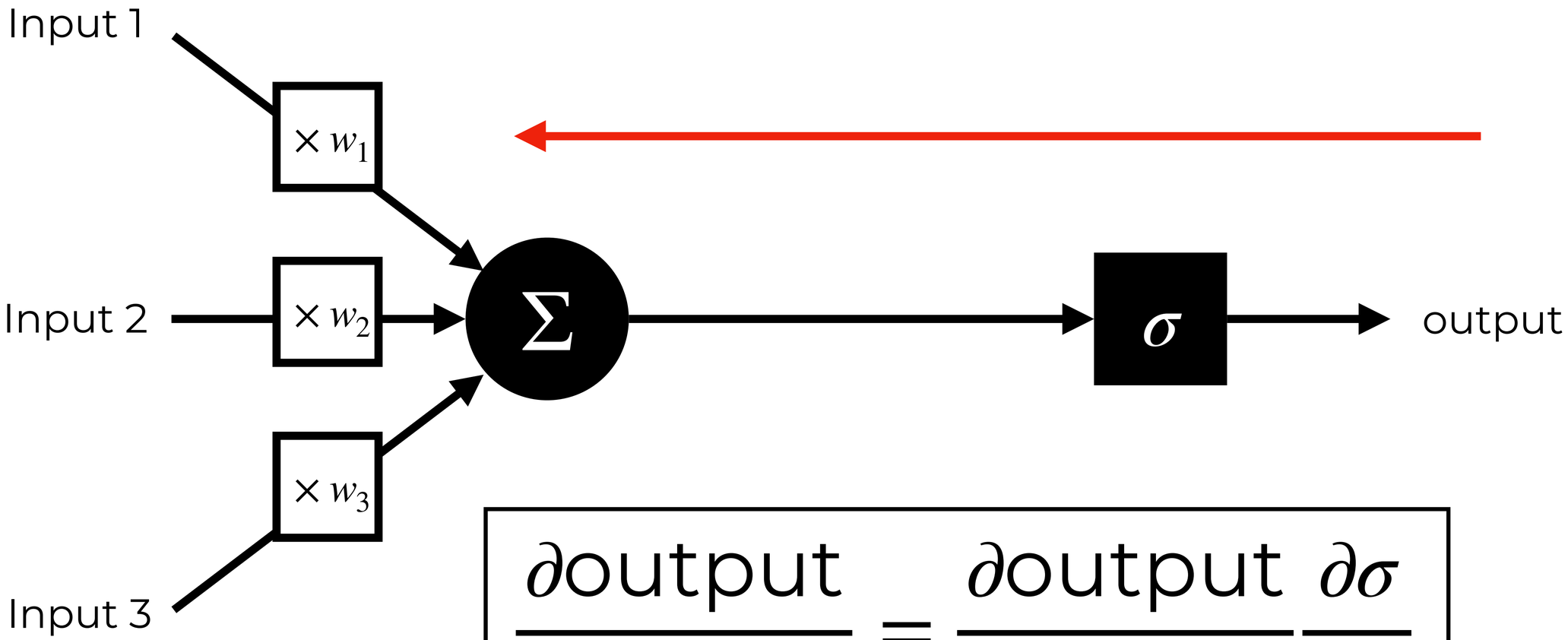
y : model prediction

Backpropagation

$$w \leftarrow w + \eta \frac{\partial \mathscr{L}}{\partial w}$$

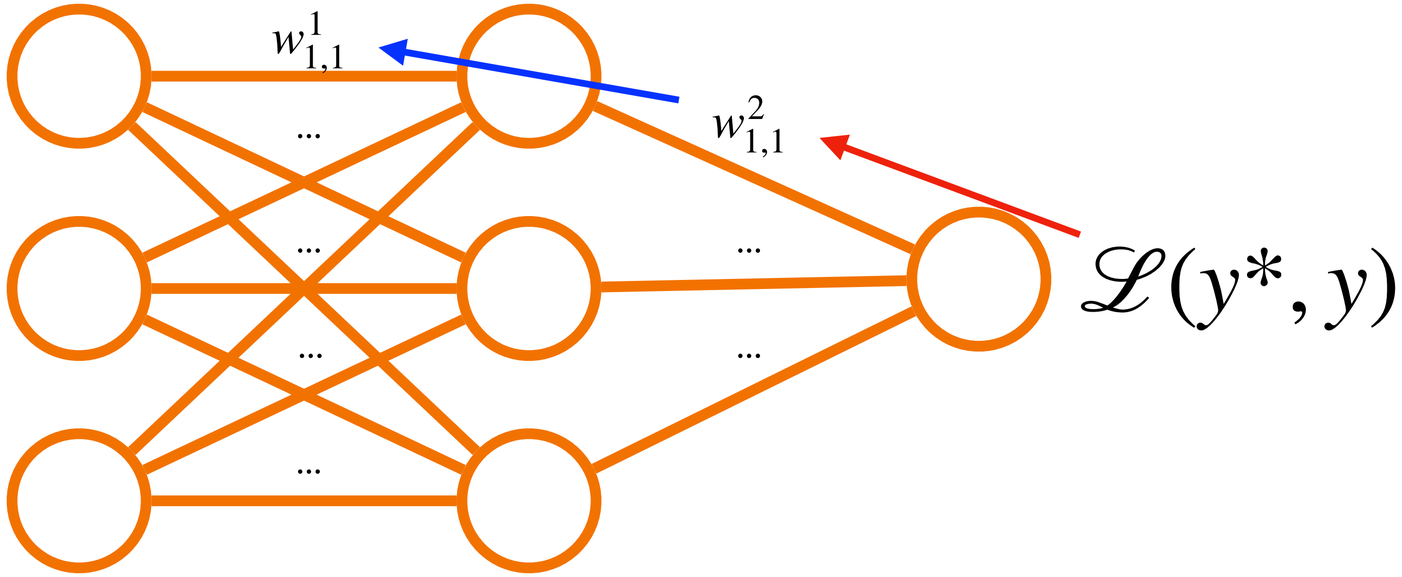


Backpropagation

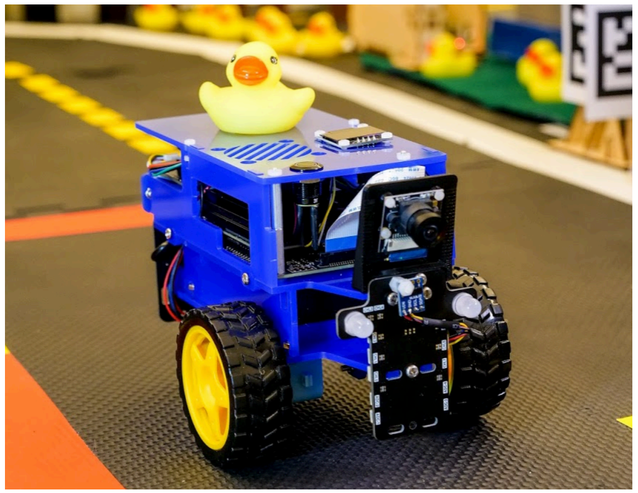


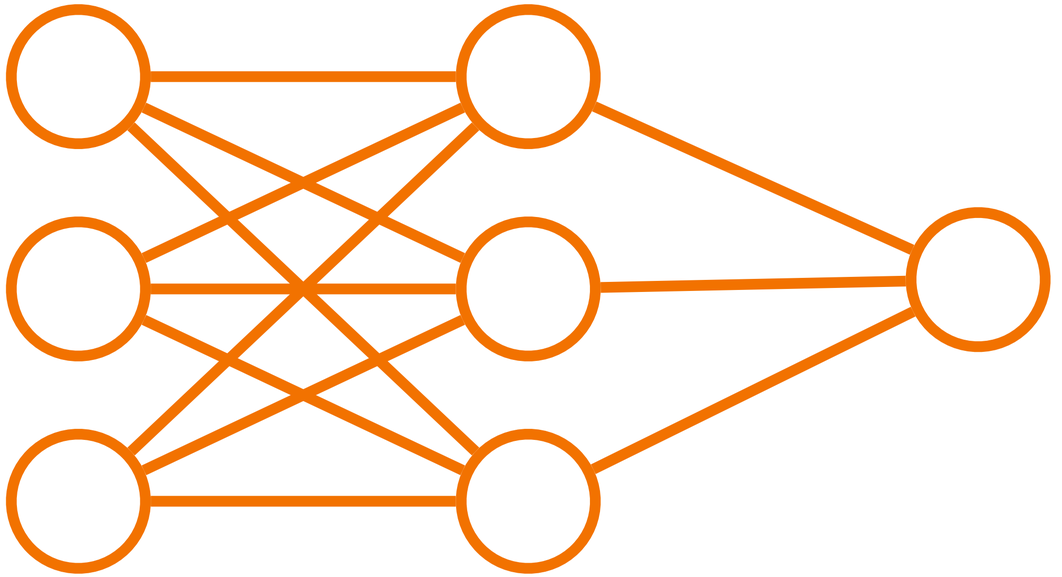
$$\frac{\partial \text{output}}{\partial w} = \frac{\partial \text{output}}{\partial \sigma}\frac{\partial \sigma}{\partial w}$$

Backpropagation



$$\frac{\partial \mathscr{L}}{\partial w^1_{1,1}} = \frac{\partial \mathscr{L}}{\partial w^2_{1,1}} \frac{\partial w^2_{1,1}}{\partial w^1_{1,1}}$$

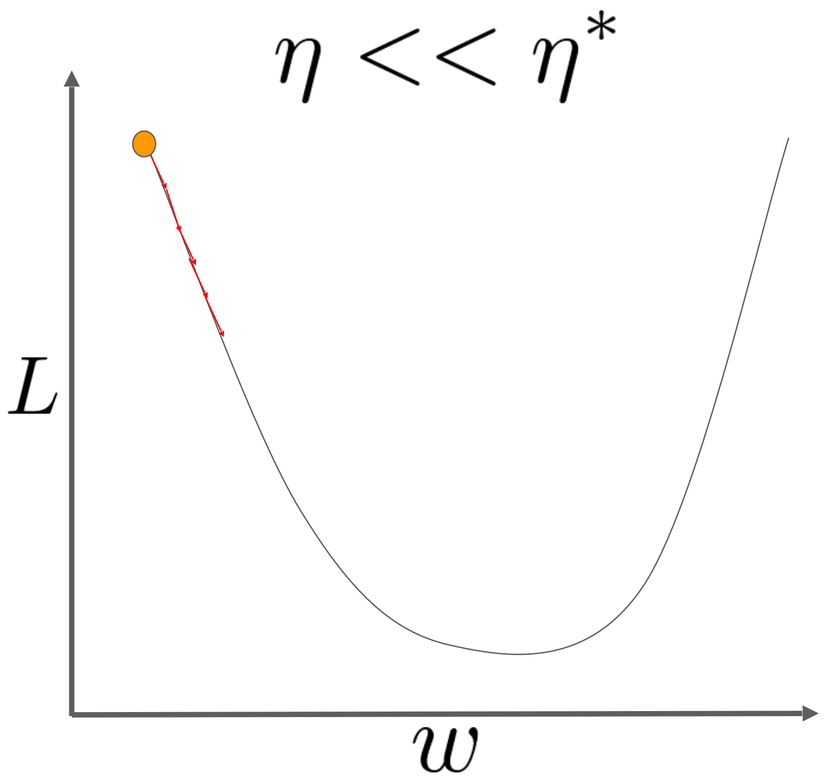


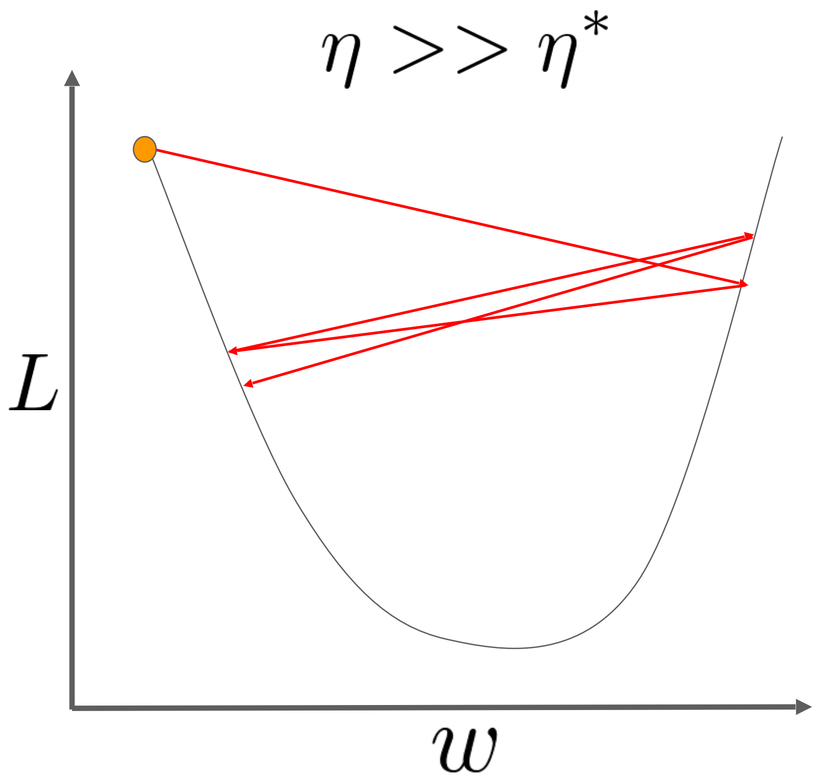


$$w \leftarrow w + \eta \frac{\partial \mathcal{L}}{\partial w}$$

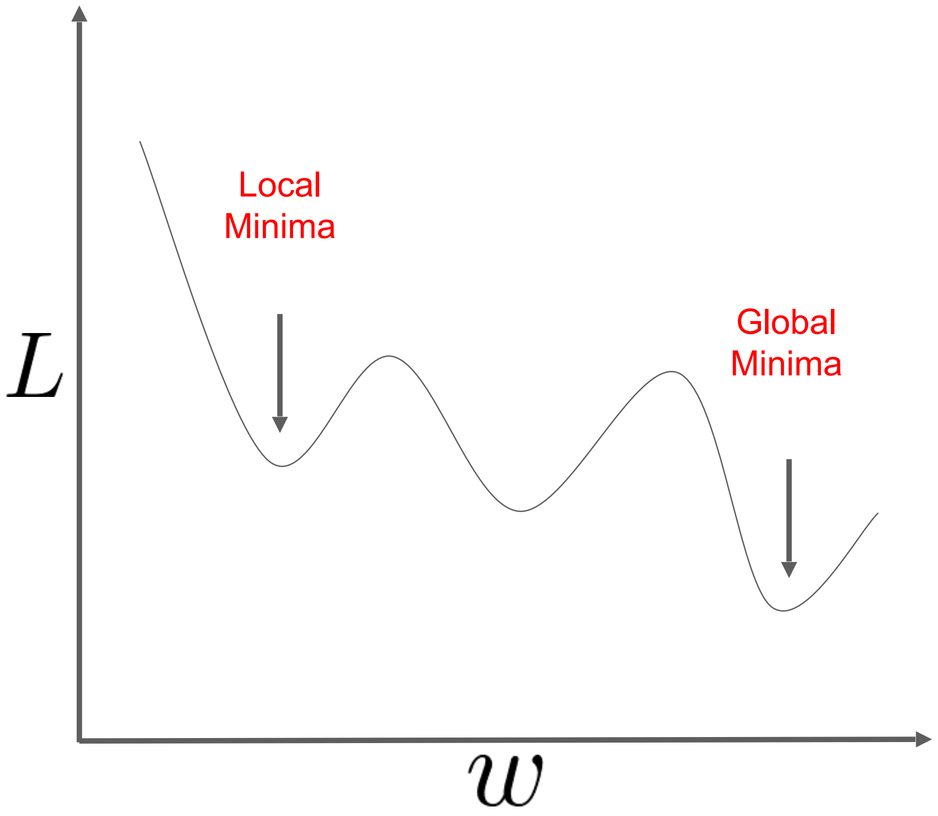
step size

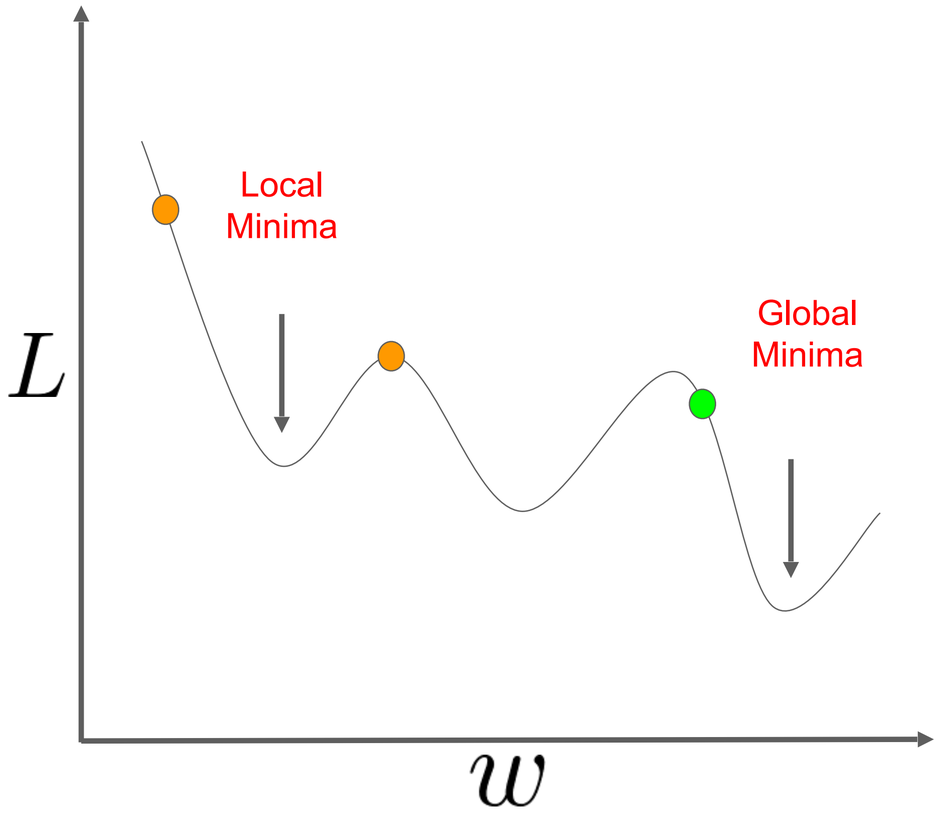
The step size



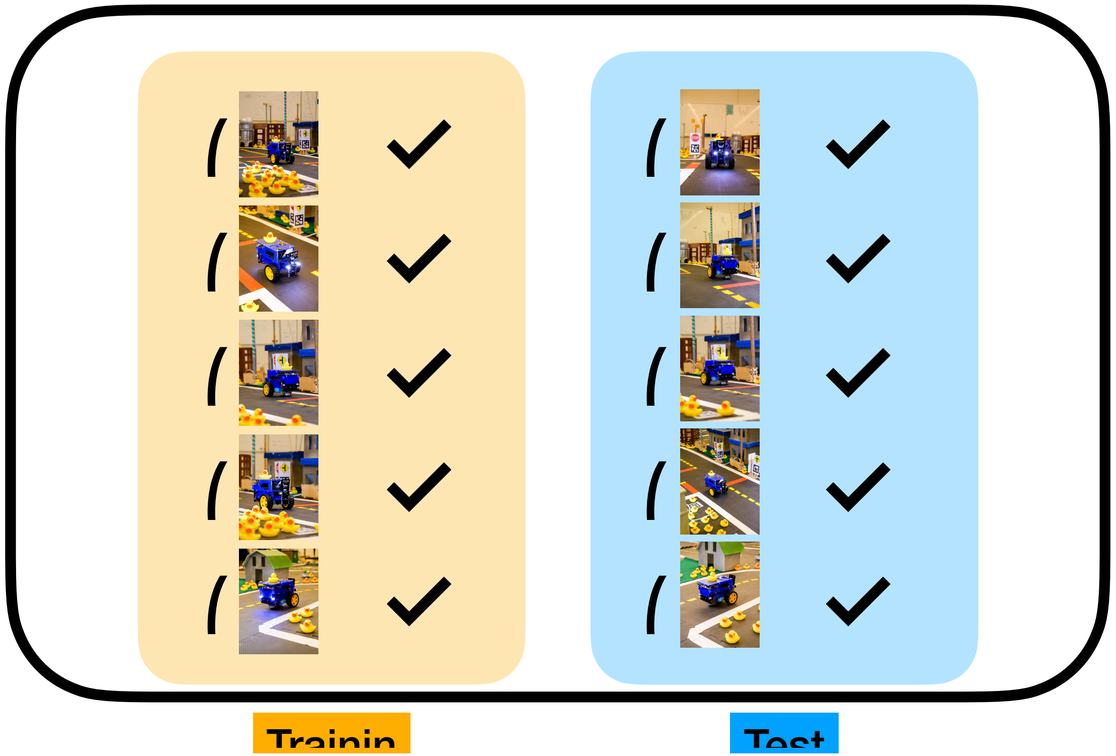


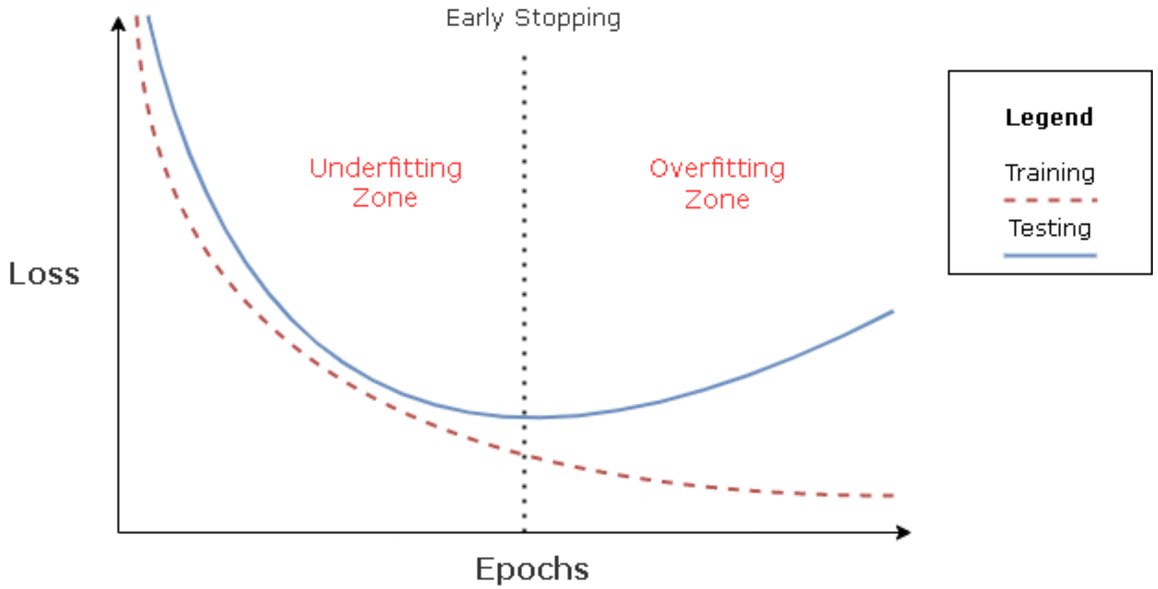
Convexity


Local
Minima
Global
Minima
L
w

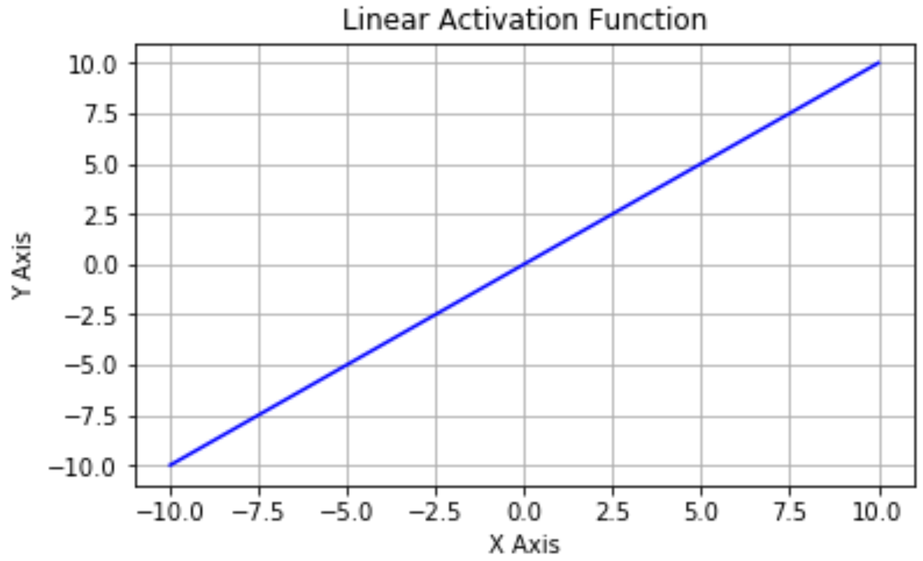

Local
Minima
Global
Minima
L
w

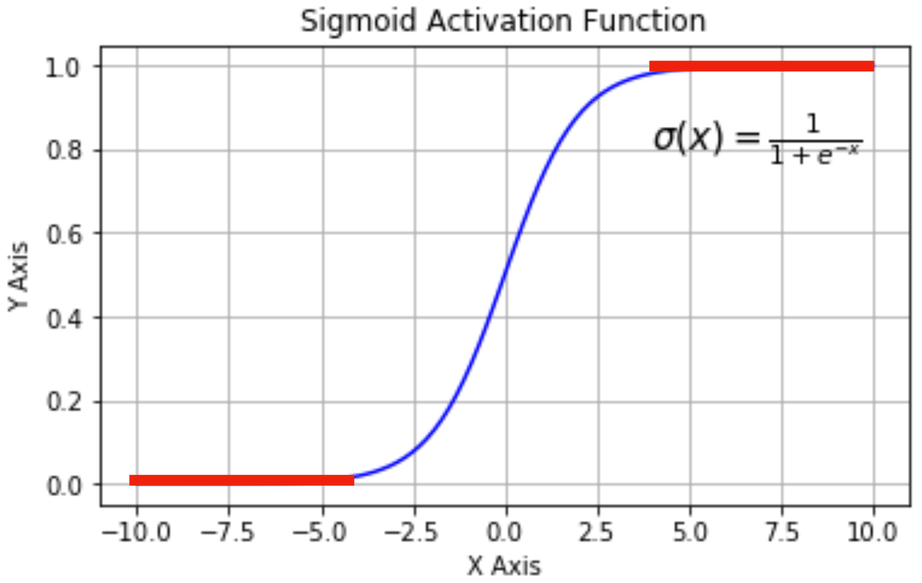
Annotated Dataset

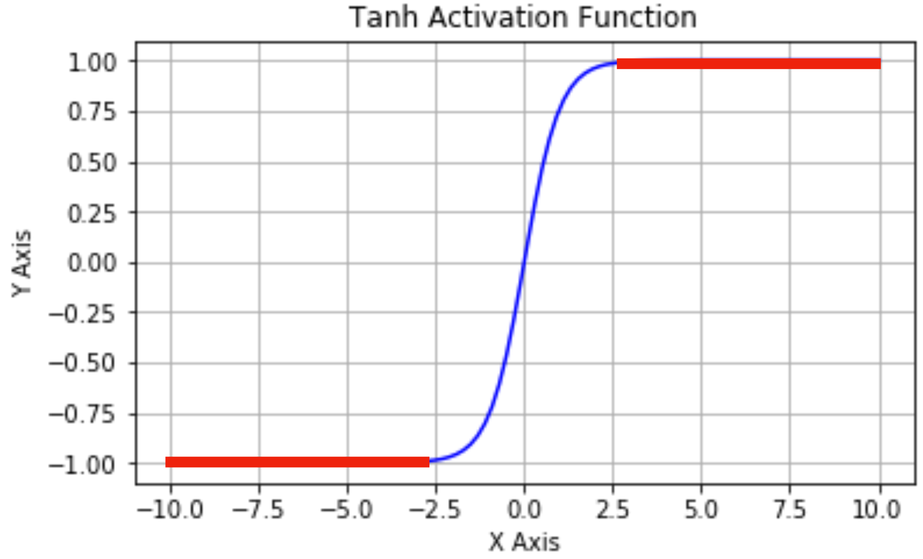




Activation functions

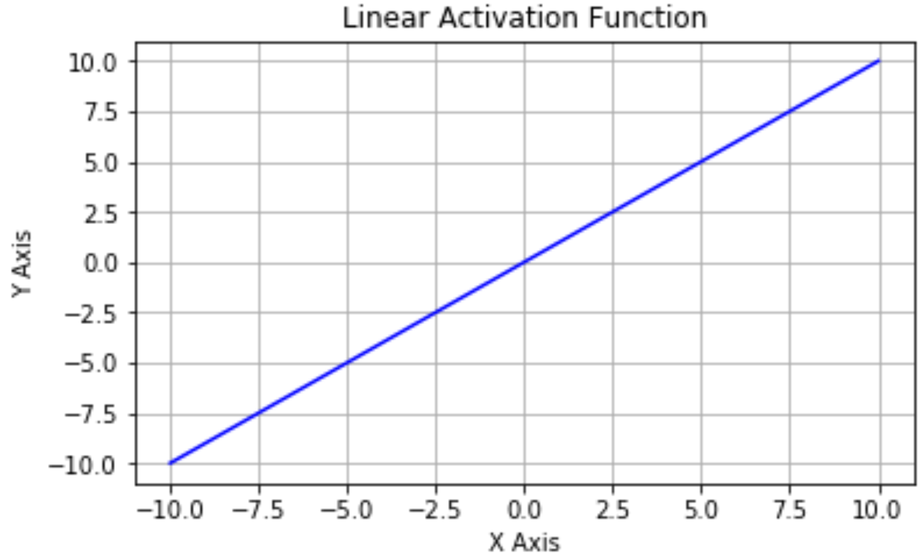


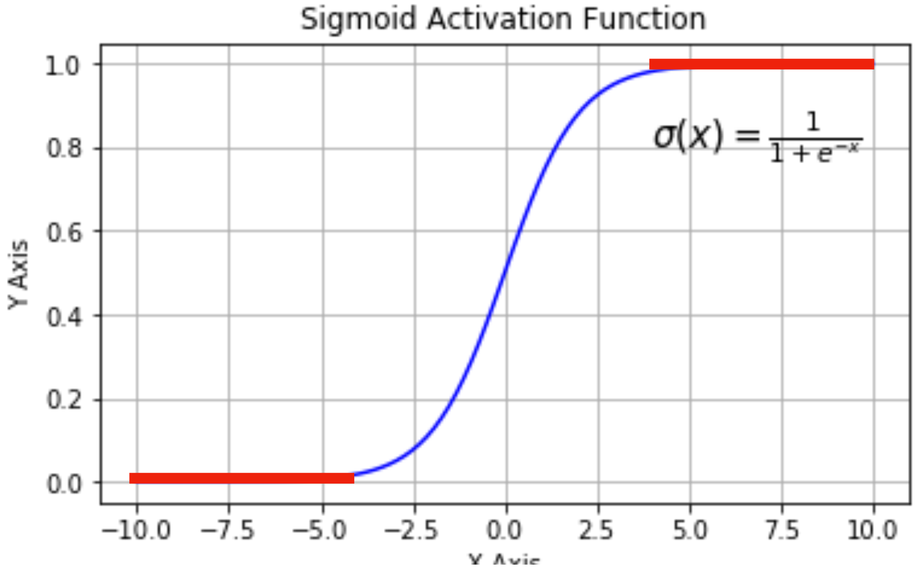


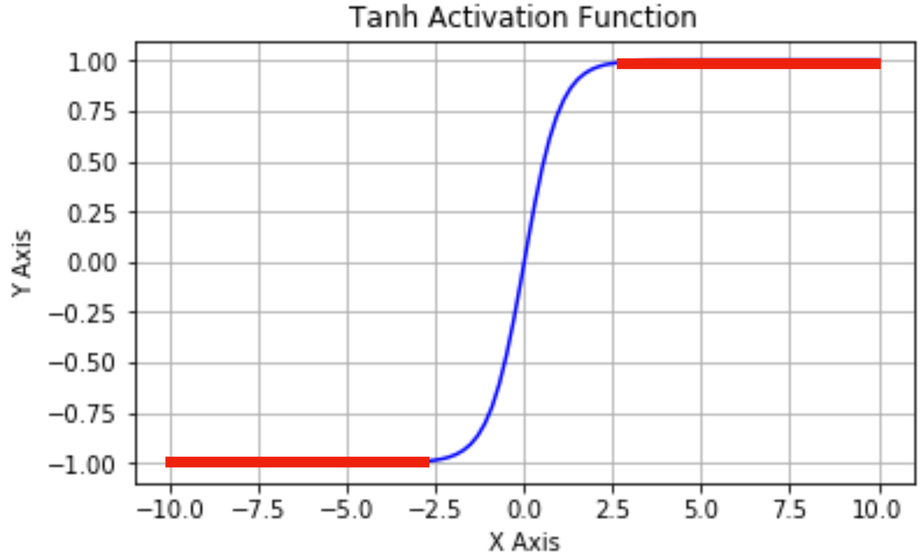


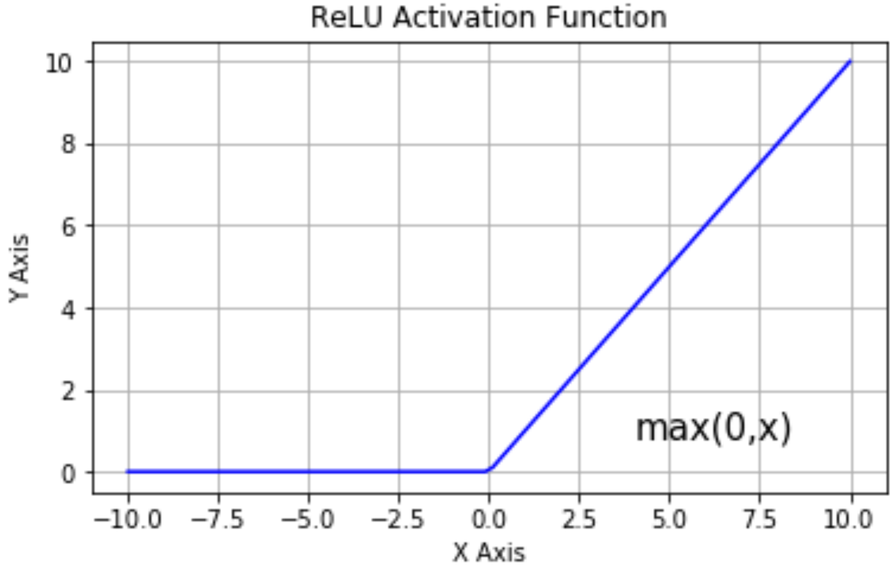
Saturation -> Vanishing Gradients

Activation functions



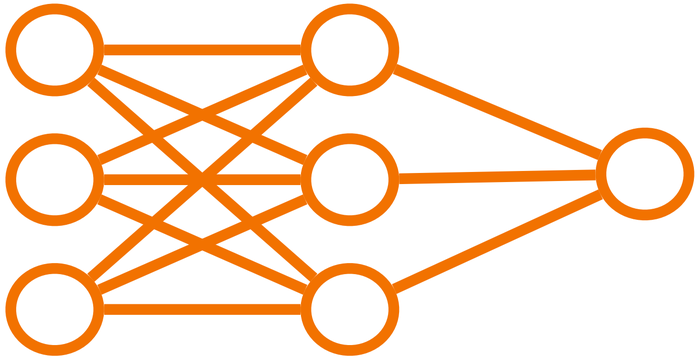


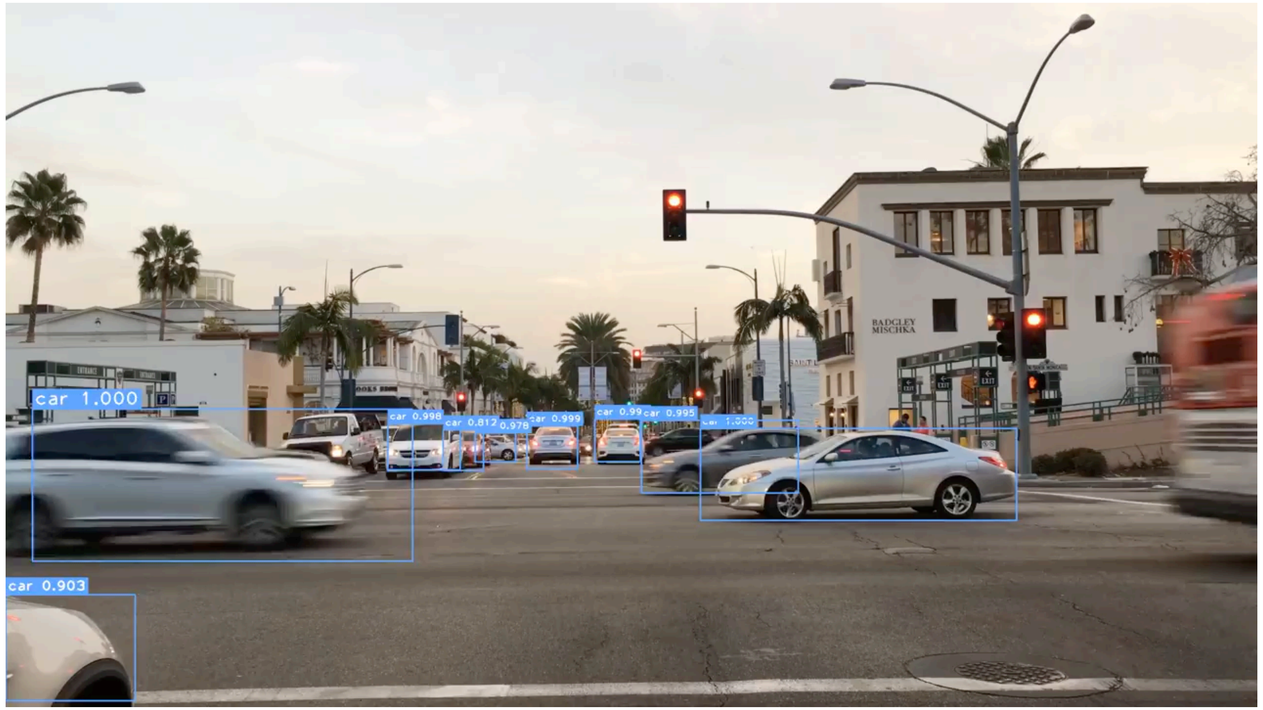




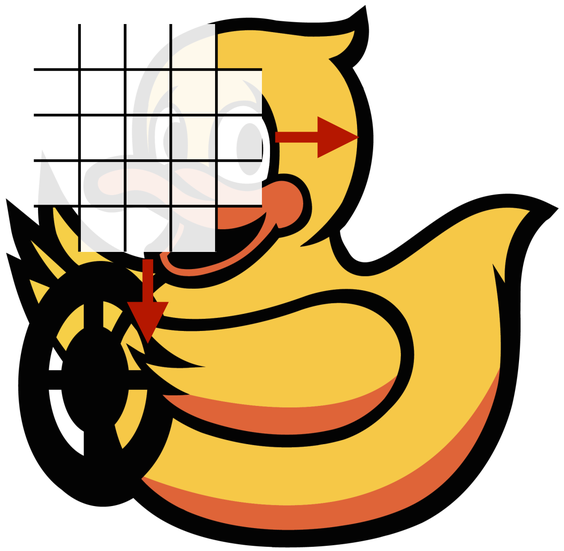
Advanced Visual Perception

Table of Contents



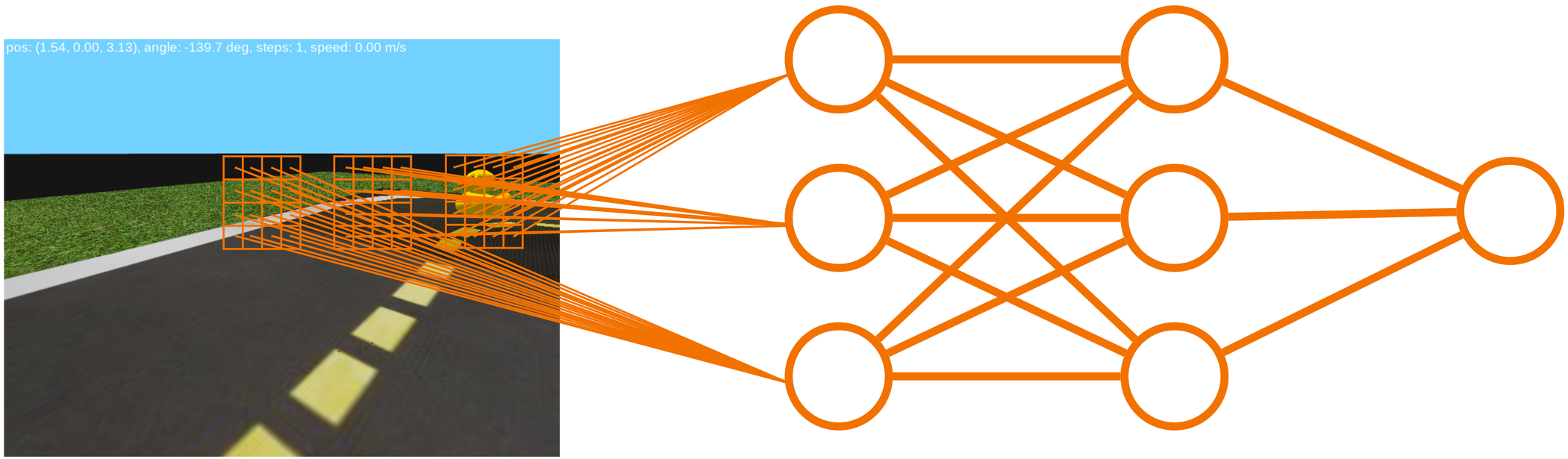

car 1.000
car 0.998
car 0.812
0.978
car 0.999
car 0.99
car 0.995
car 1.000
car 0.903
BADGLEY MISCHKA

Convolution

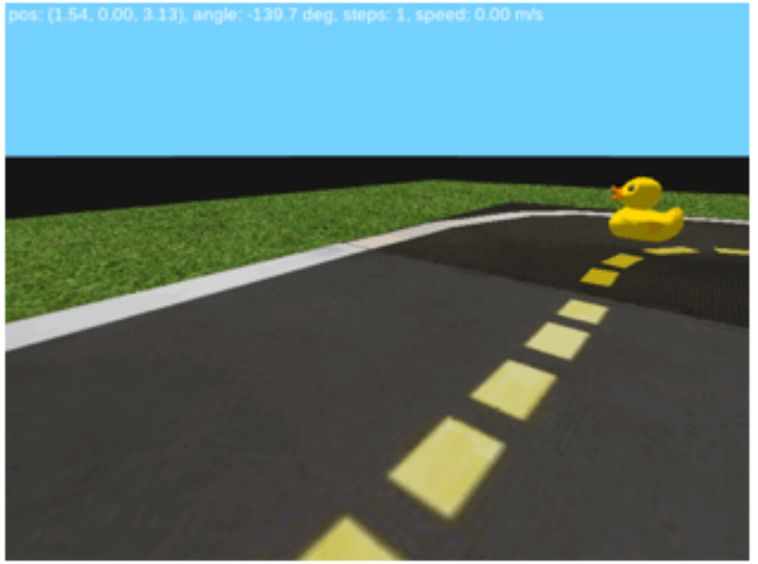


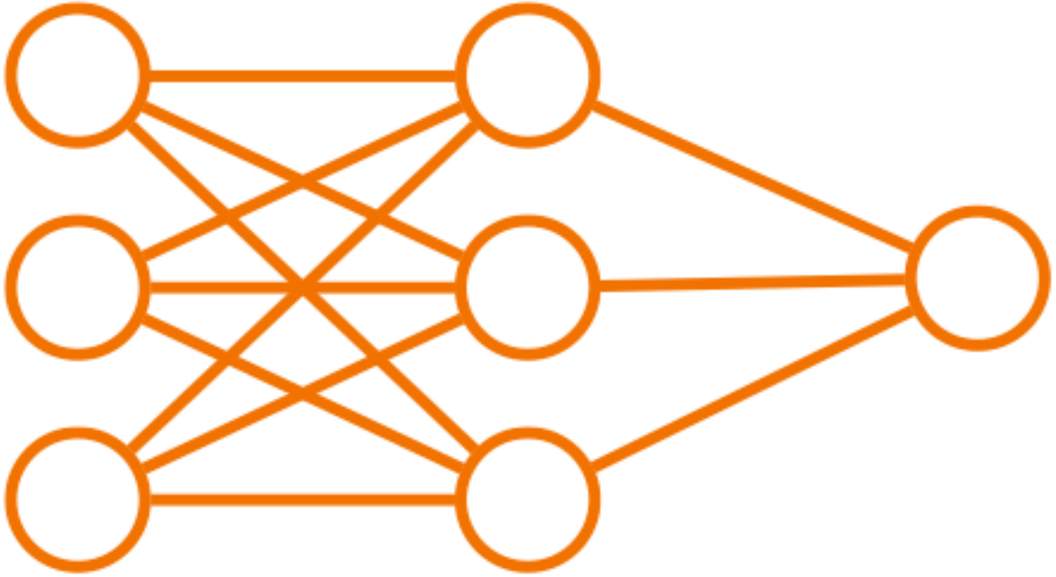
Convolutional layers

Convolutional layers

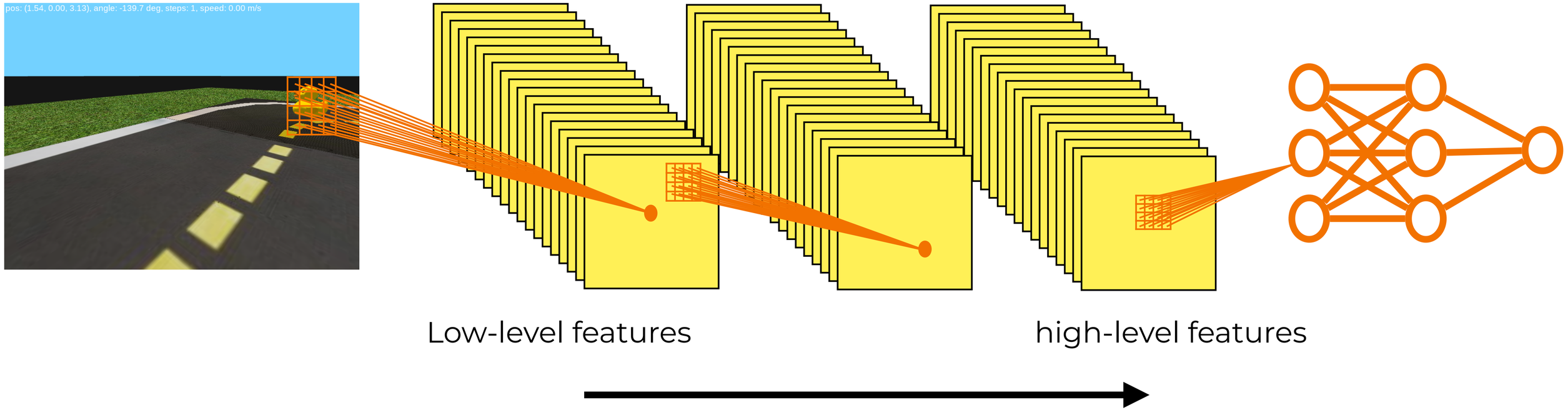


Convolutional layers

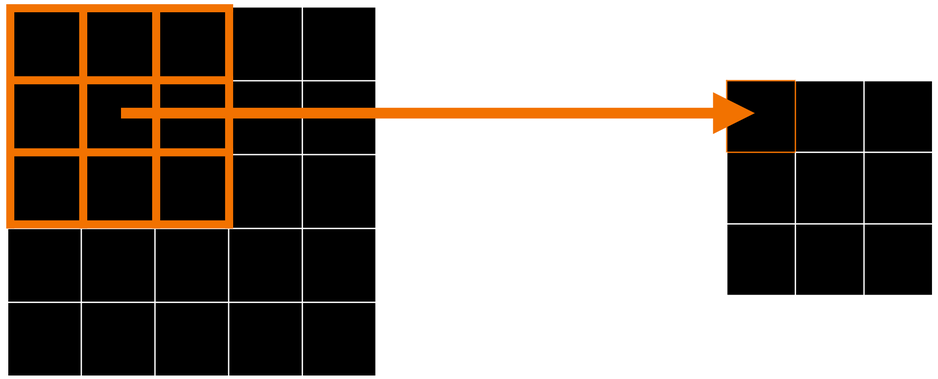




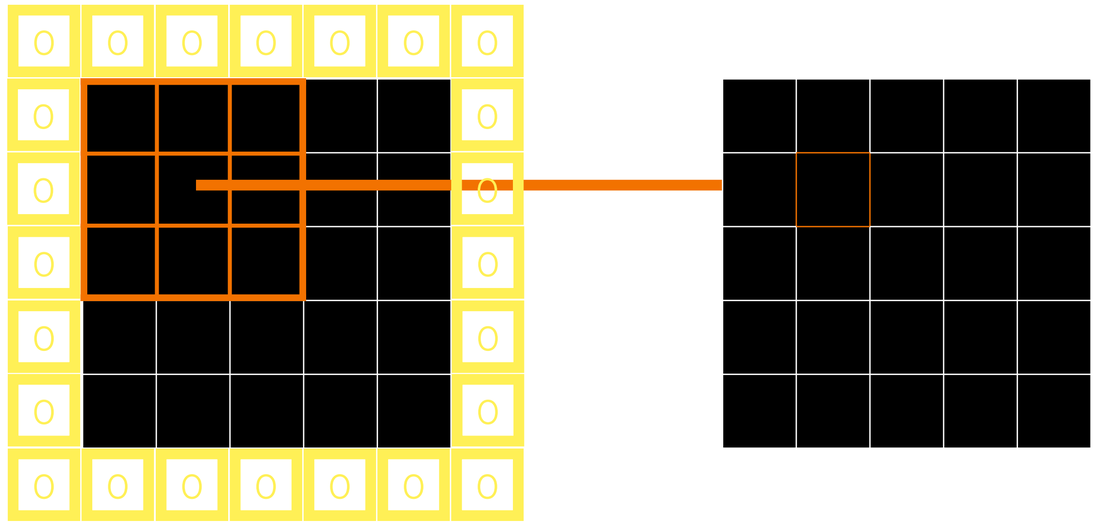
Convolutional layers



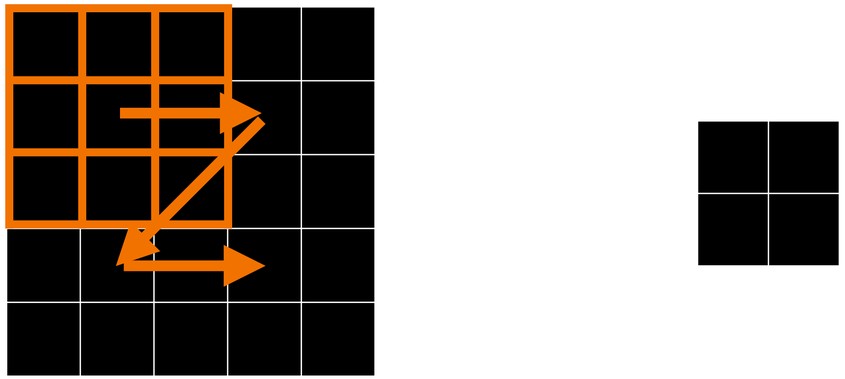
Padding



Padding

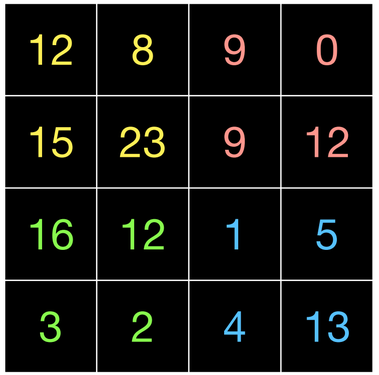


Stride



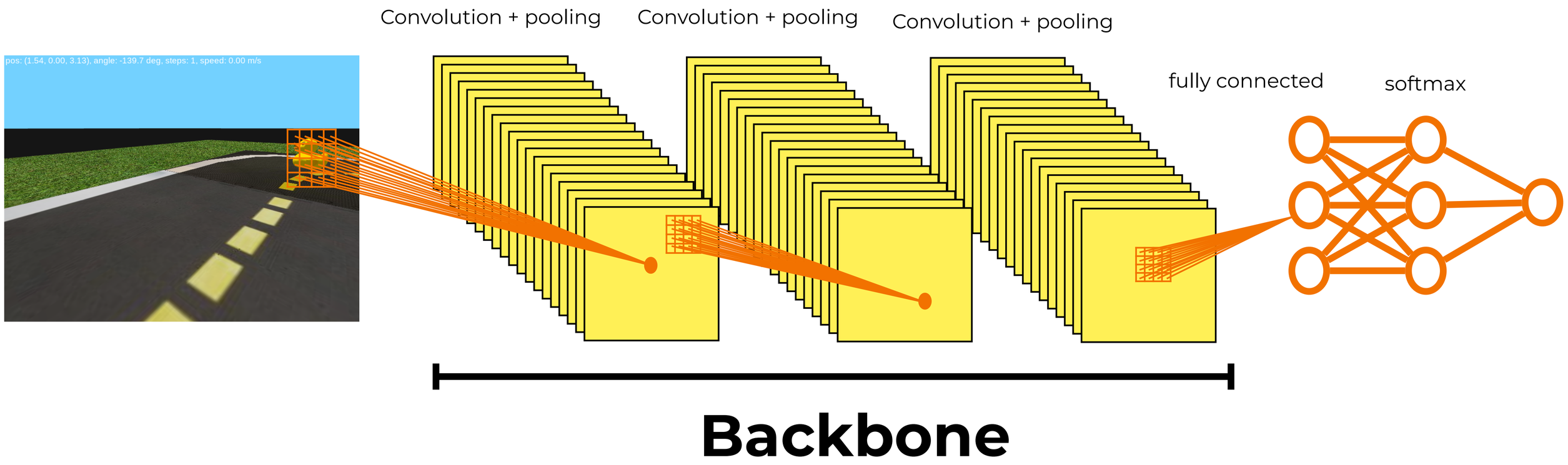
Stride = 2

Pooling

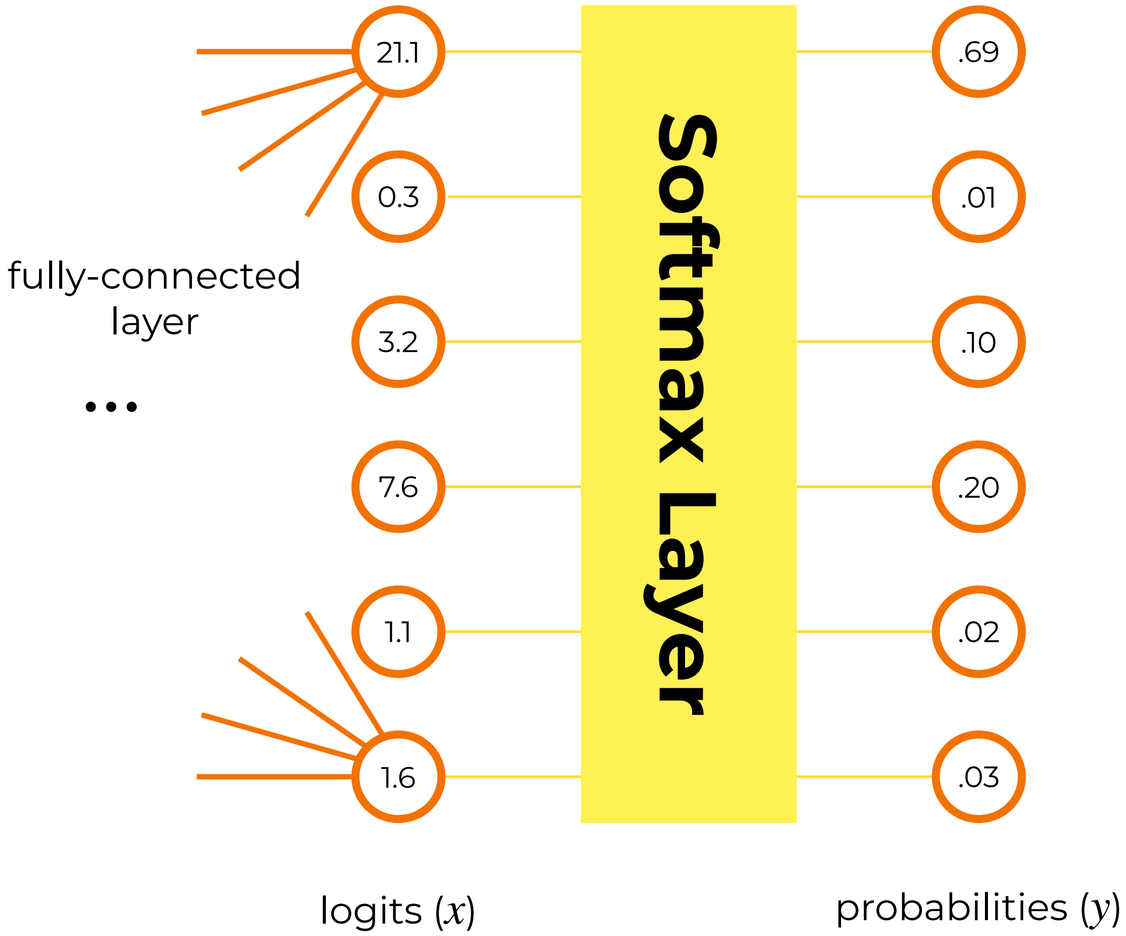


Stride = 2

Convolutional layers

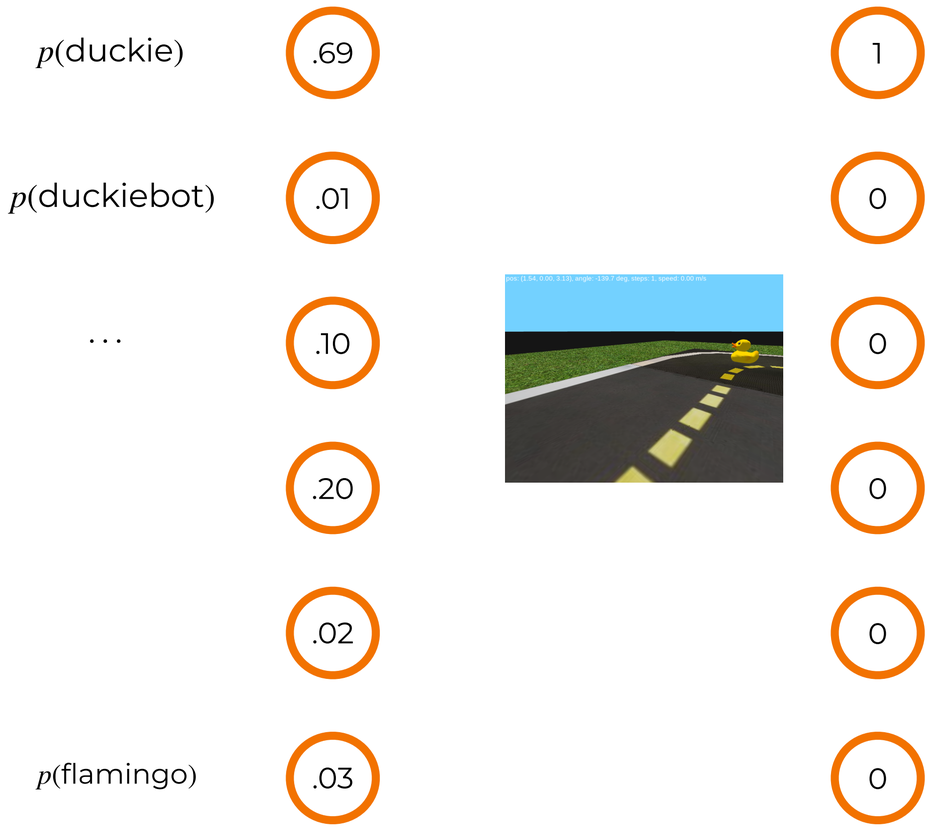


Softmax



$$\text{softmax}(y_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$

Cross-entropy loss



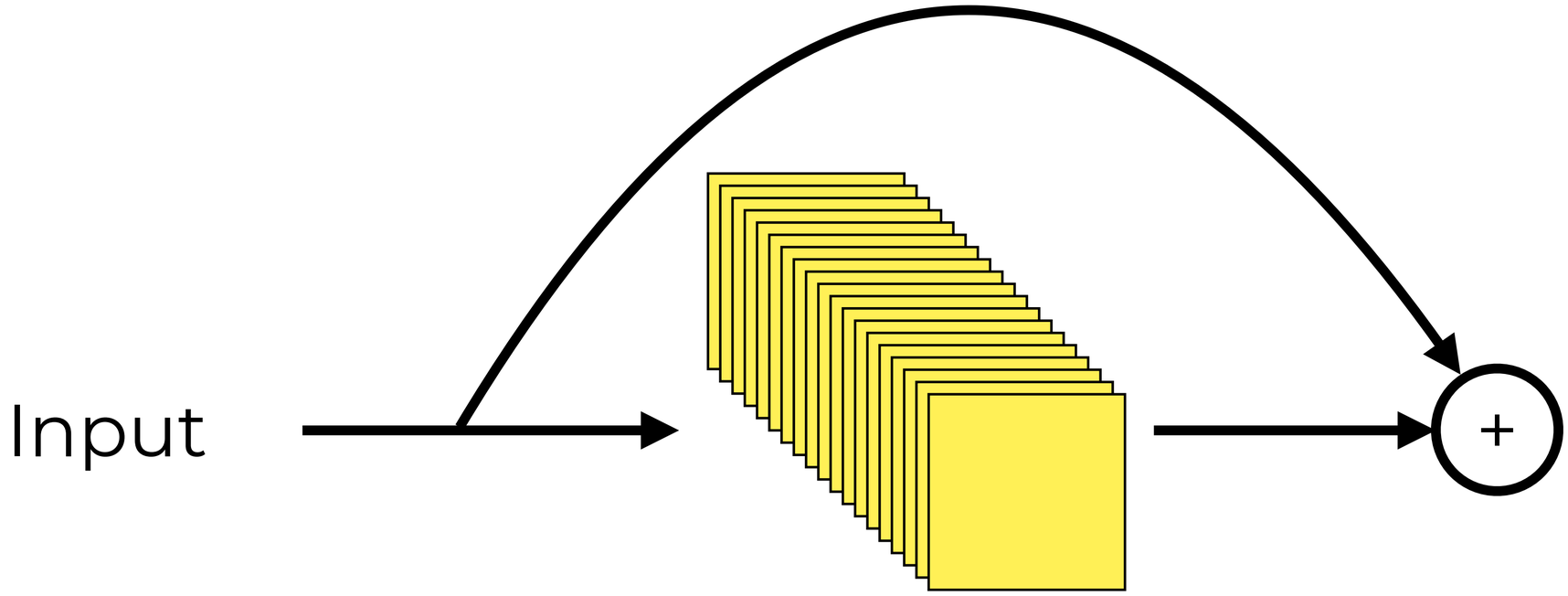
Binary cross-entropy loss:

$$-(y^* \log(y) + (1 - y^*)\log(1 - y))$$

Multi-class cross-entropy loss:

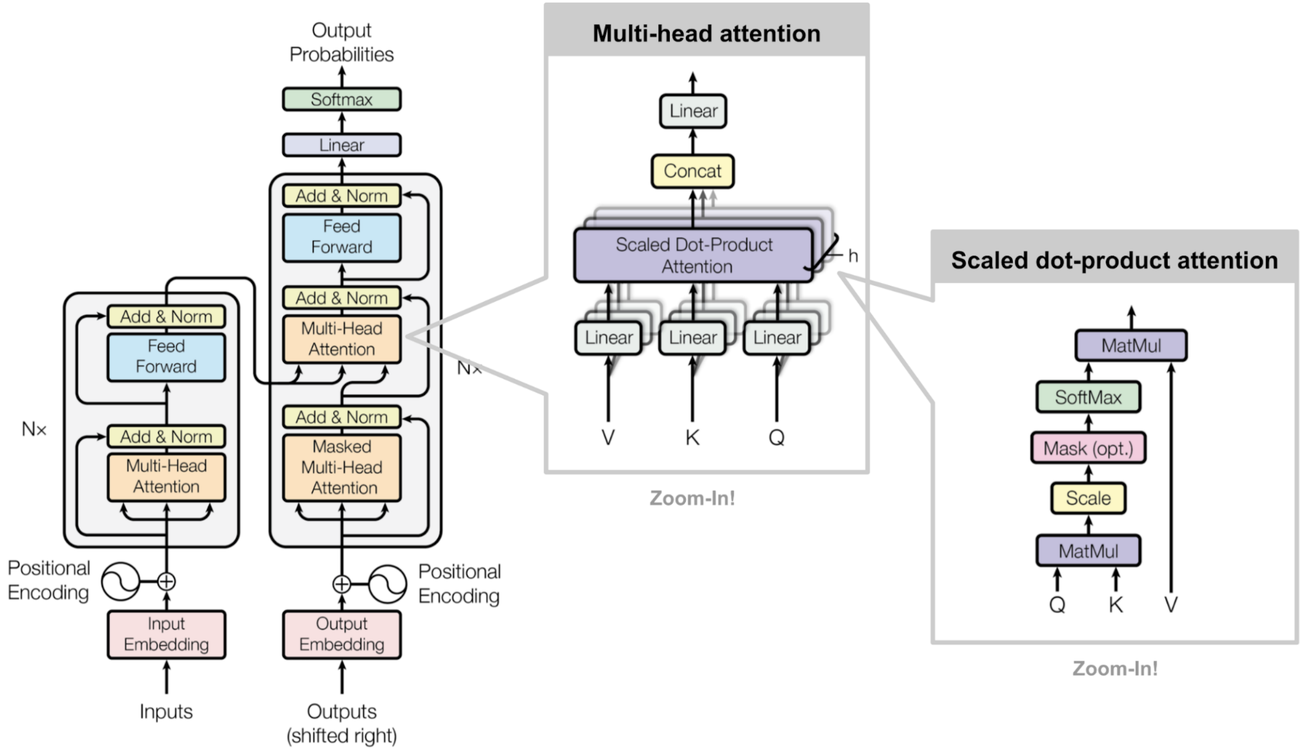
$$-\sum_{c=1}^{M} p(y_c^*)\log p(y_c)$$

Residual layers



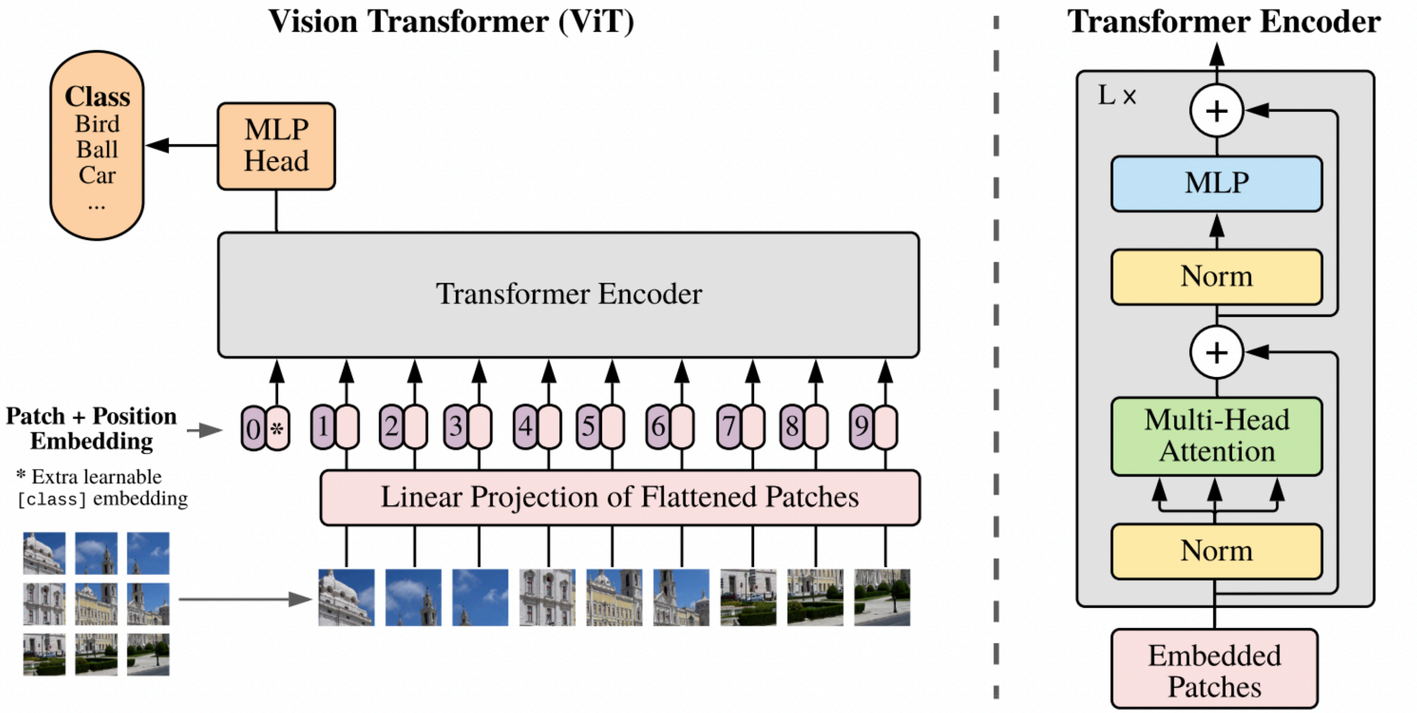
Transformer

- Origins in NLP
- “Tokenize” and “encode” the input
- Use attention to allow the relationships between tokens to be learned
- Decode the output



Vision Transformer

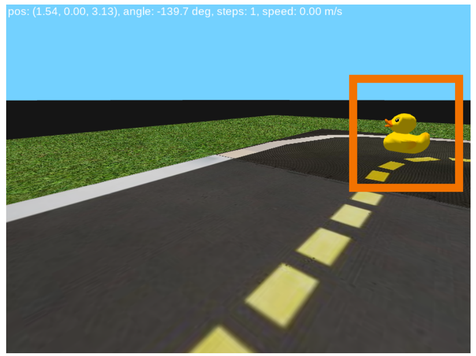
- Similar idea but tokenize the image using image patches
- instead of predicting next word we add a new [CLS] token at the end
- Supposedly less inductive bias than a CNN (allows contextual information to flow further more easily)

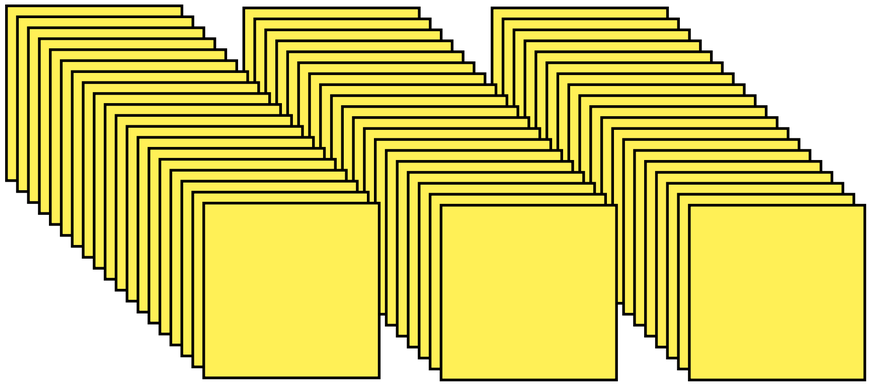


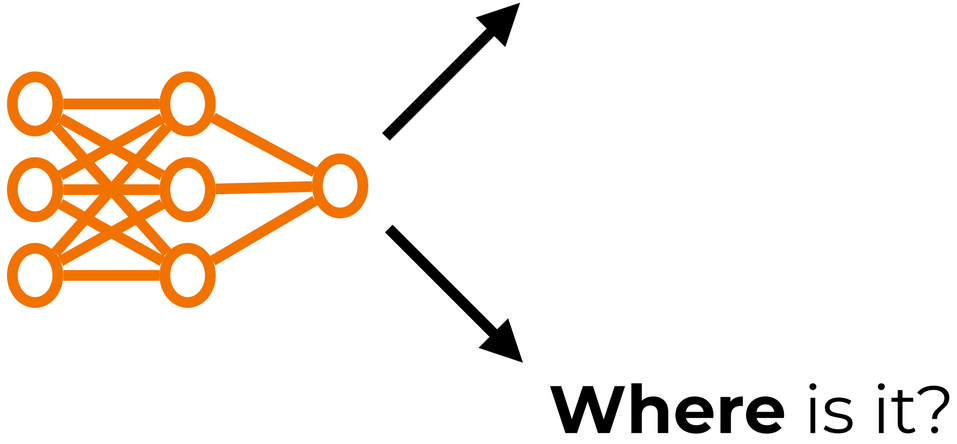
Advanced Visual Perception

Table of Contents

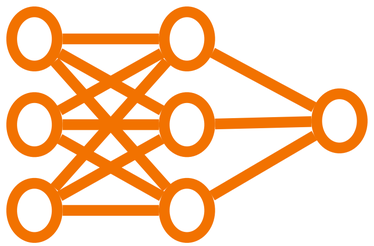
CNNs for object detection







Loss Function



What is it?

Softmax Layer

class probabilities

$\mathscr{L}_{what} = \text{CE}(y)$

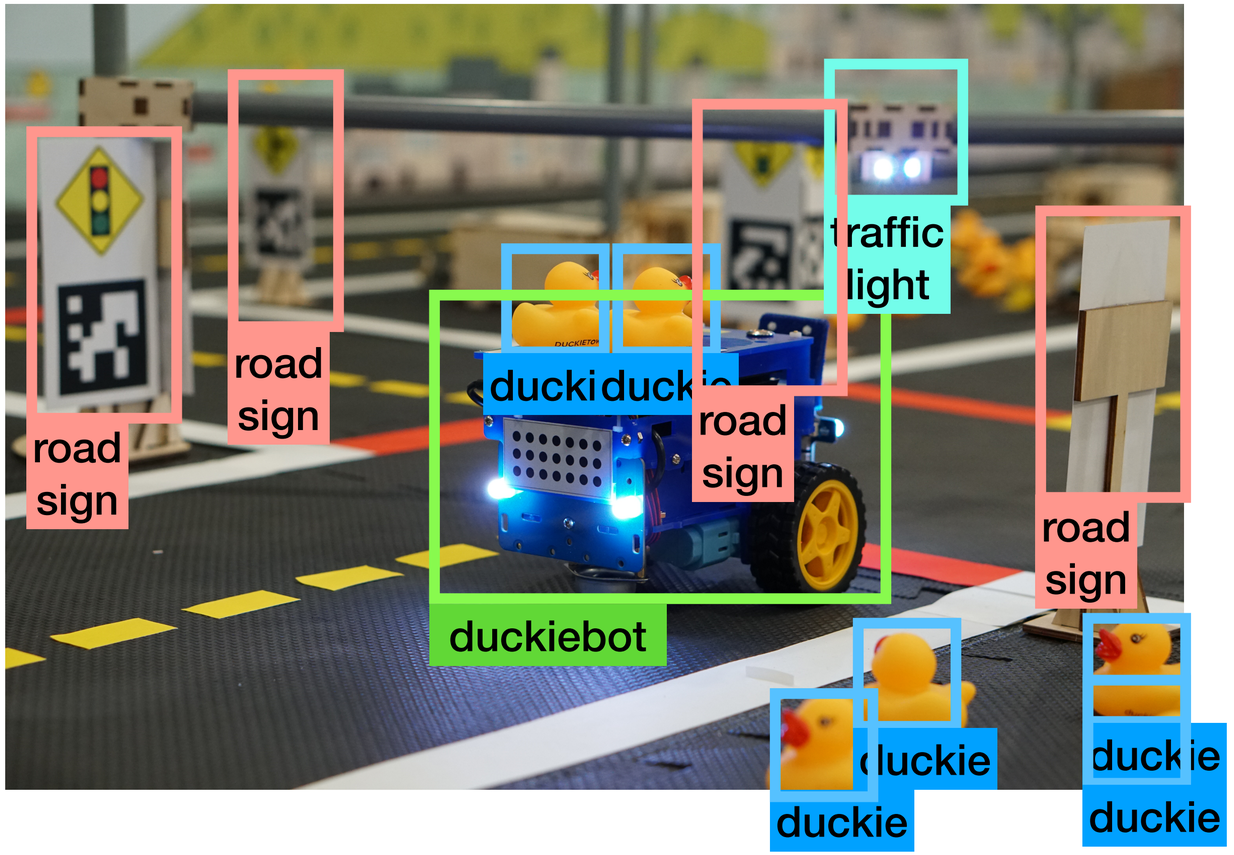
Where is it?

bounding box position and size

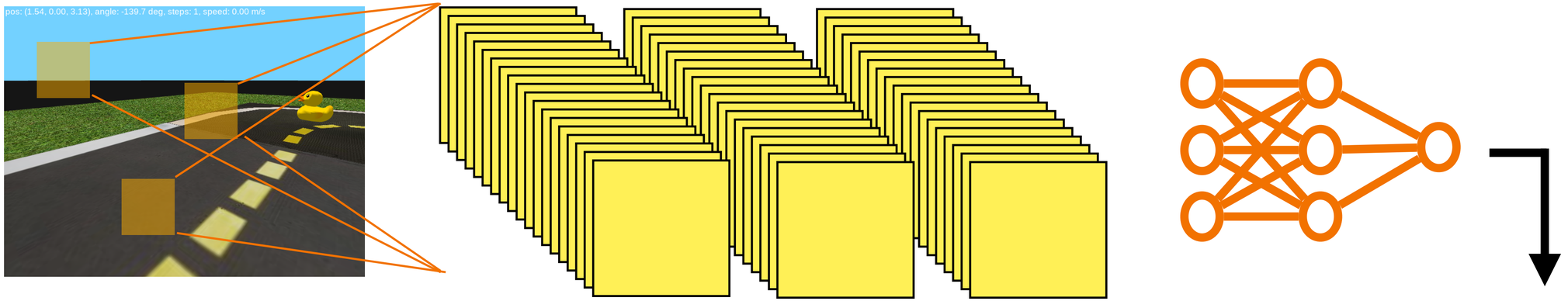
(x, y, w, h)

$\mathscr{L}_{where} = ||\text{predicted\_box} - \text{actual\_box}||_2$

$\mathscr{L}_{obj\_det} = \alpha\mathscr{L}_{what} + \beta\mathscr{L}_{where}$


road sign
road sign
traffic light
road sign
road sign
duckiebot
duckie
duckie
duckie
duckie

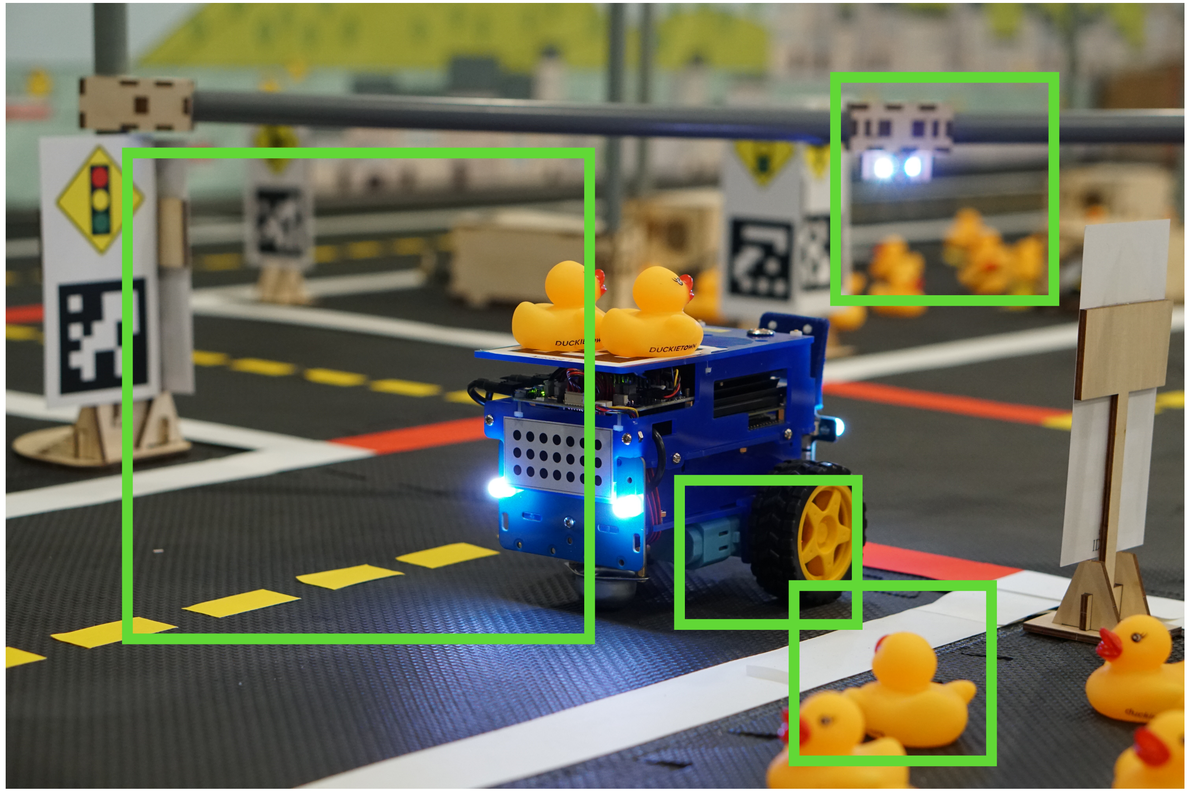
CNNs for object detection



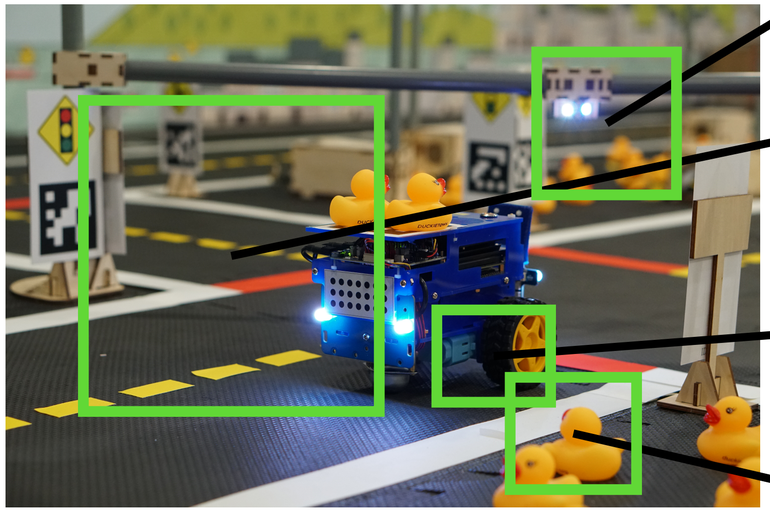
$$\text{class} \in \{\text{duckie}, \text{duckiebot}, \ldots, \text{background}\}$$

Too many possible boxes to do this exhaustively

Region proposals



R-CNN: Region-Based CNN



Warp

Warp

Warp

Warp

CNN

CNN

CNN

CNN

Class

Δ bounding box

Class

Δ bounding box

Class

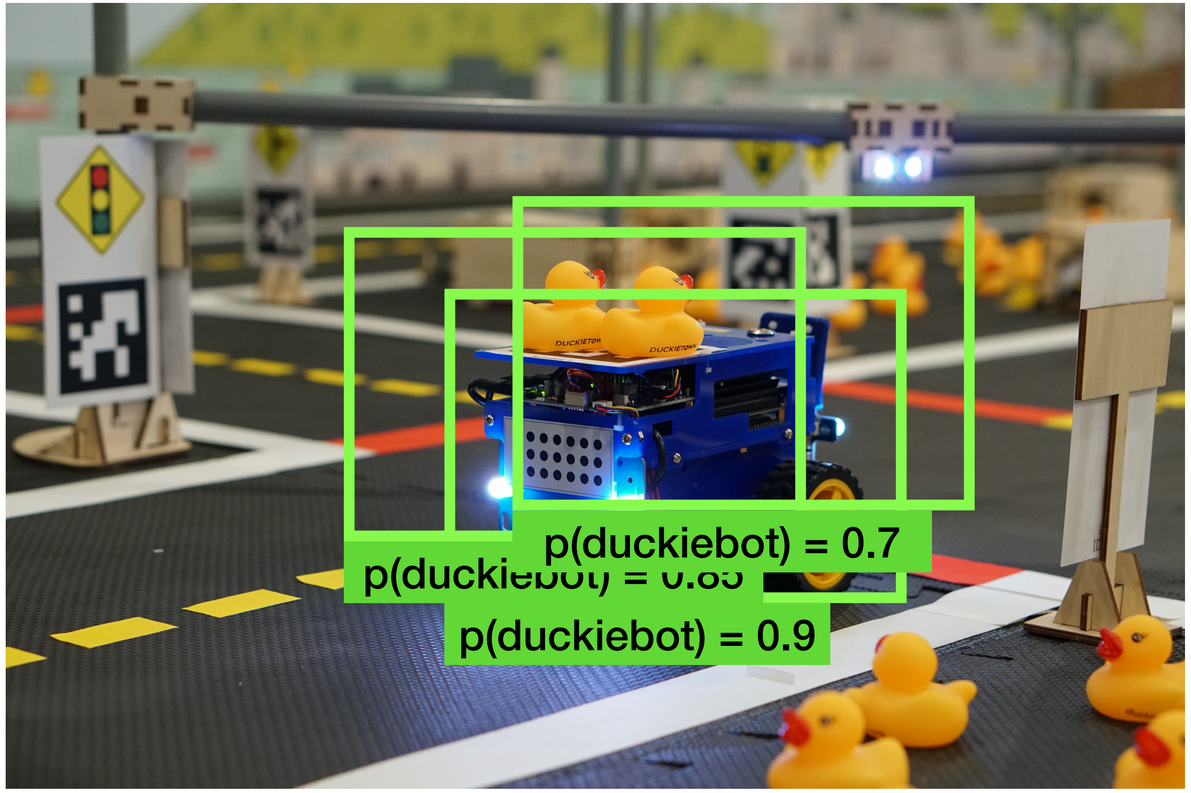
Δ bounding box

Class

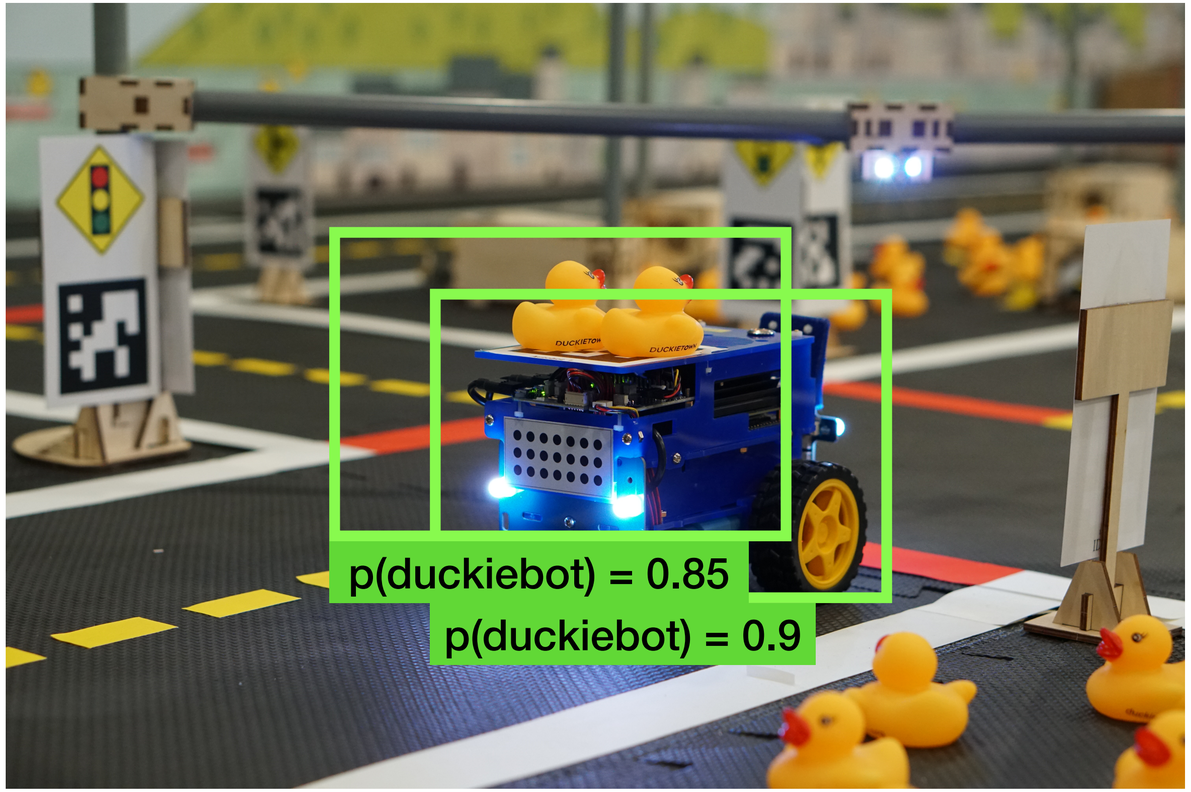
Δ bounding box

~2000 region proposals
picked with selective search

Non-maximum suppression

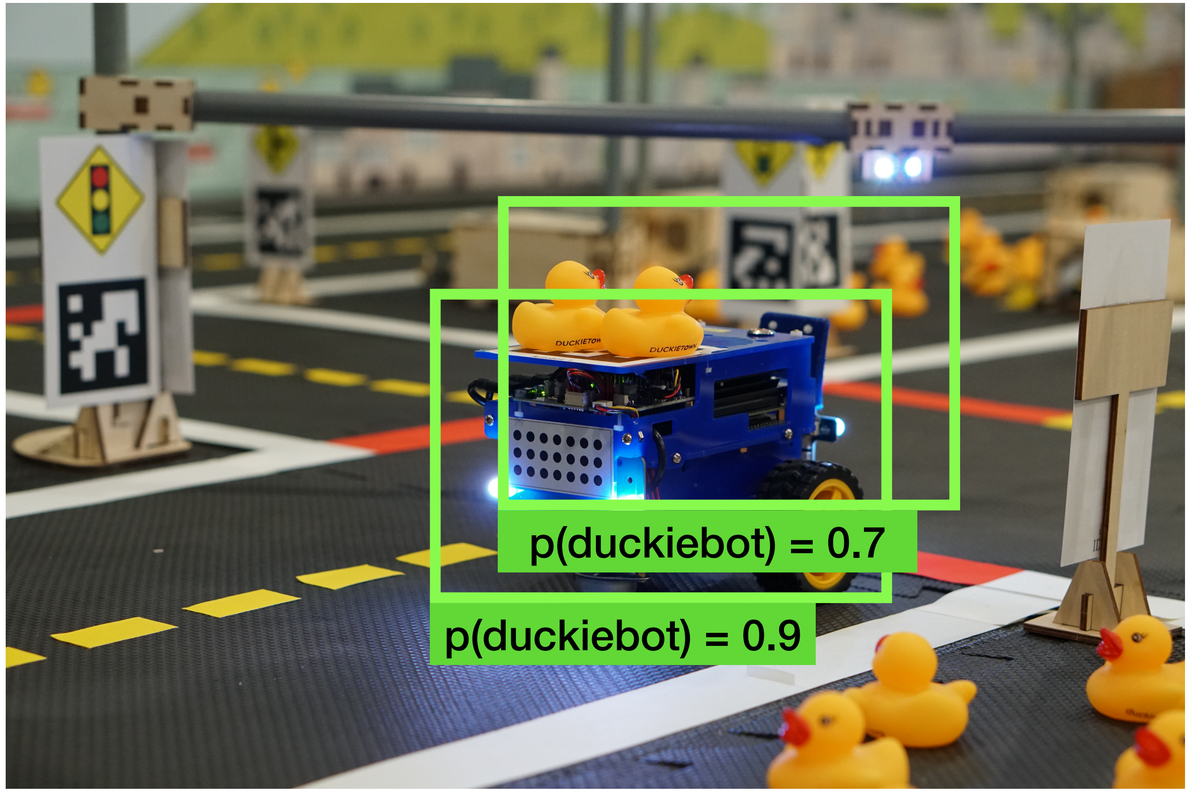


Non-maximum suppression



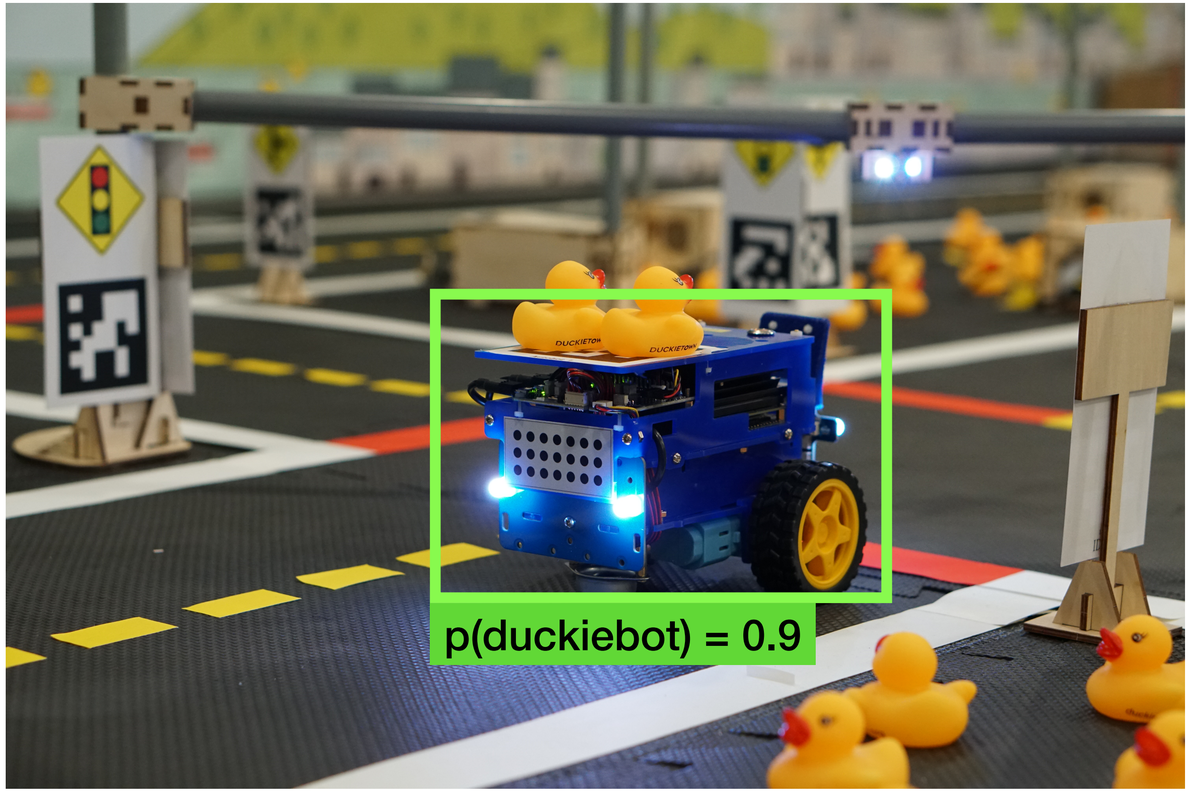
IOU > threshold

Non-maximum suppression



IOU > threshold

Non-maximum suppression



Fast R-CNN

CNN

Backbone

Image features

Crop & Resize

Crop & Resize

Crop & Resize

Crop & Resize

Class

Δ bounding box

Class

Δ bounding box

Class

Δ bounding box

Class

Δ bounding box

Region of Interest (RoI) pooling

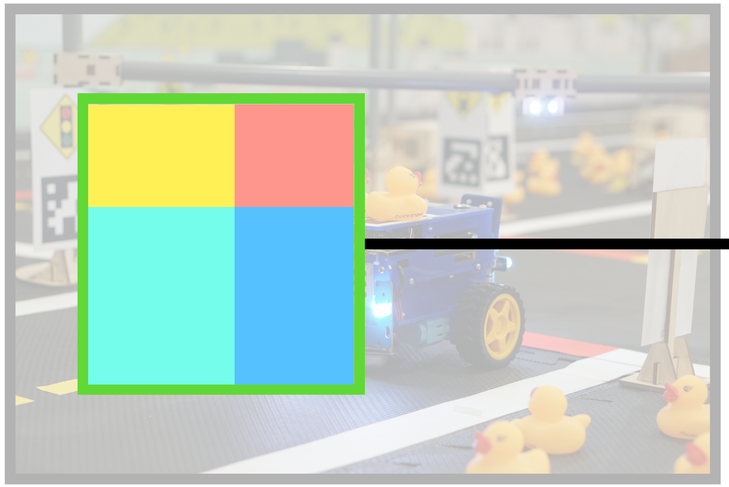
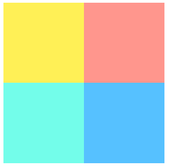


Image features

Max Pooling



Region features
(fixed size)

Faster R-CNN

CNN

Backbone

Image features

RPN

RoI Pool

Class

Δ bounding box

Class

Δ bounding box

Class

Δ bounding box

Class

Δ bounding box

Region Proposal Network

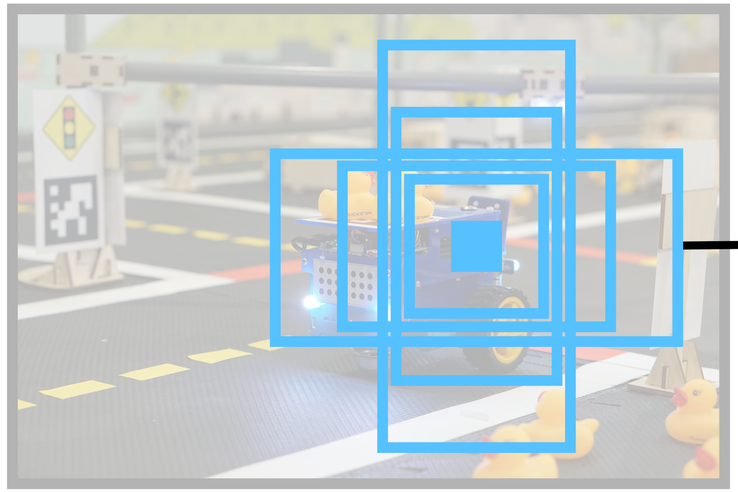
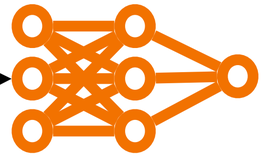


Image features

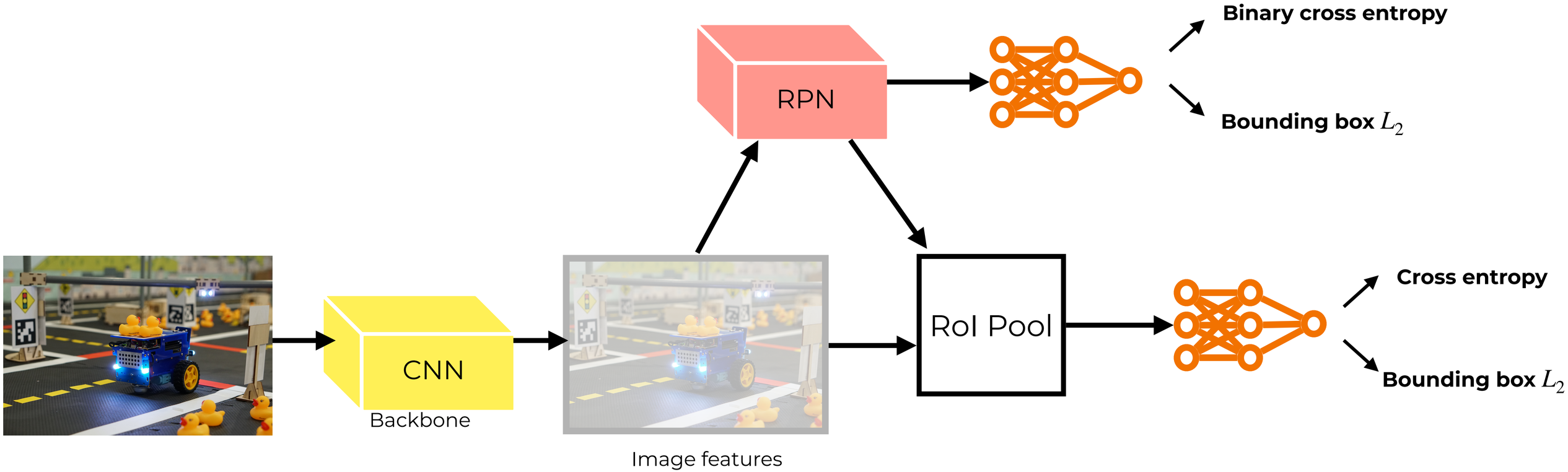


Object?

Δ bounding box

: Anchor boxes

Faster R-CNN

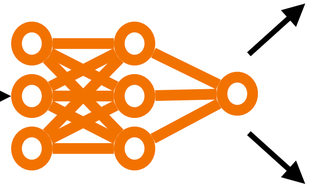

RPN
Binary cross entropy
Bounding box L_2
CNN
Backbone
Image features
RoI Pool
Cross entropy
Bounding box L_2

You Only Look Once (YOLO)

CNN

Backbone

Image features



Class

Δ bounding box

: Anchor boxes

No need for RPN

Issues with single-stage object detectors

Problem 1: Difficult to detect objects at different scales

Solution 1: Use features from different layers of backbone

Method 1: Single shot multibox (SSD)

Issues with single-stage object detectors

Problem 1: Difficult to detect objects at different scales

Solution 1: Use features from different layers of backbone

Method 1: Single shot multibox (SSD)

Problem 2: Data imbalance due to background

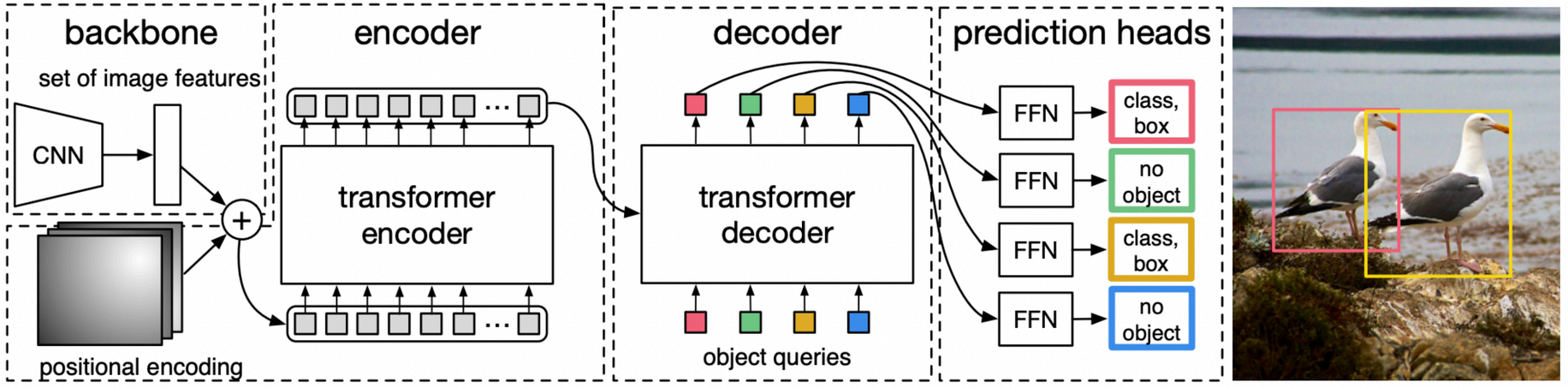
Solution 2: Focal loss that prioritizes objects

Method 2: RetinaNet

DETR

Application of vision transformers to the problem of object detection:

$$L = \sum_{i=1}^{N} \left[-\log \hat{p}_{\sigma(i)}(c_i) + \mathbf{1}_{c_i \neq \emptyset} \left(\lambda_{L1} \| b_i - \hat{b}_{\sigma(i)} \|_1 + \lambda_{giou} L_{giou}(b_i, \hat{b}_{\sigma(i)}) \right) \right]$$



Datasets

- Kitti, CityScapes Nuscenes

- Mostly annotated by hand

Advanced Visual Perception

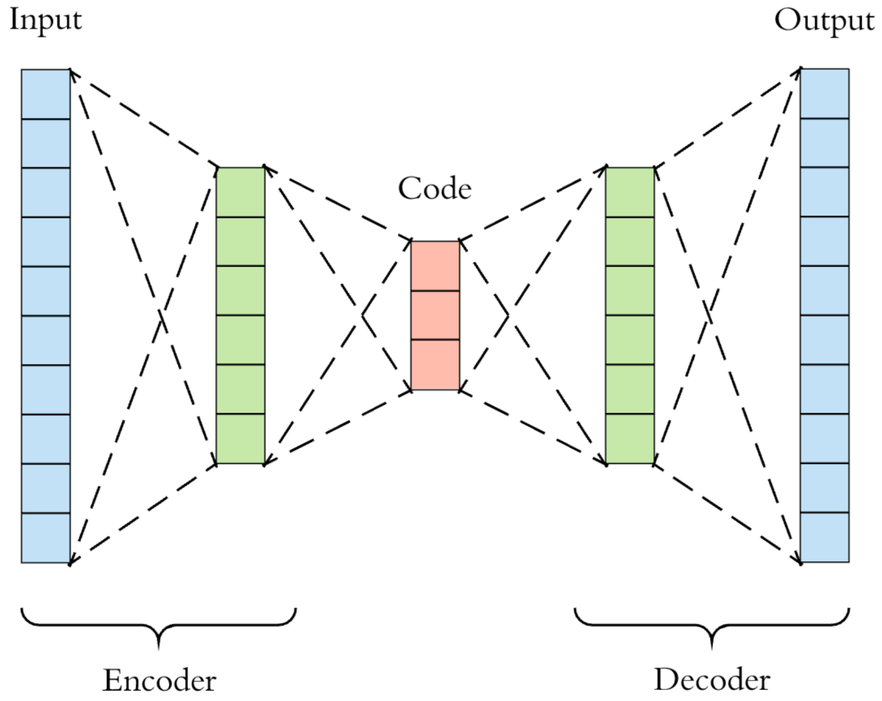
Table of Contents

Autoencoder

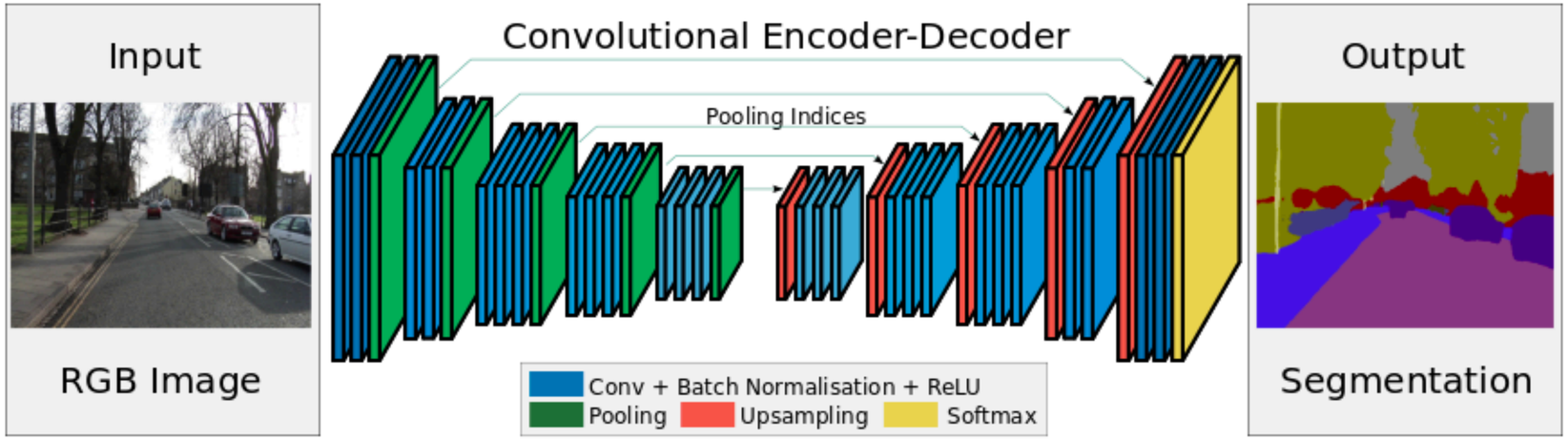
Objective: learn a latent representation of your input

No need for data

Usually: minimize the reconstruction loss (maybe regularized by something to enforce the kind of latent representation you want)



Segnet


Input
RGB Image
Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax
Output
Segmentation


Sky
Building
Pole
Road Marking
Road
Pavement
Tree
Sign Symbol
Fence
Vehicle
Pedestrian
Bike

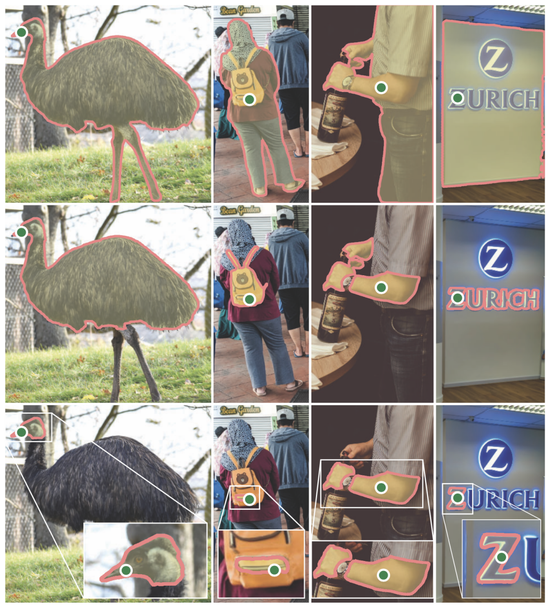
Segment Anything Model

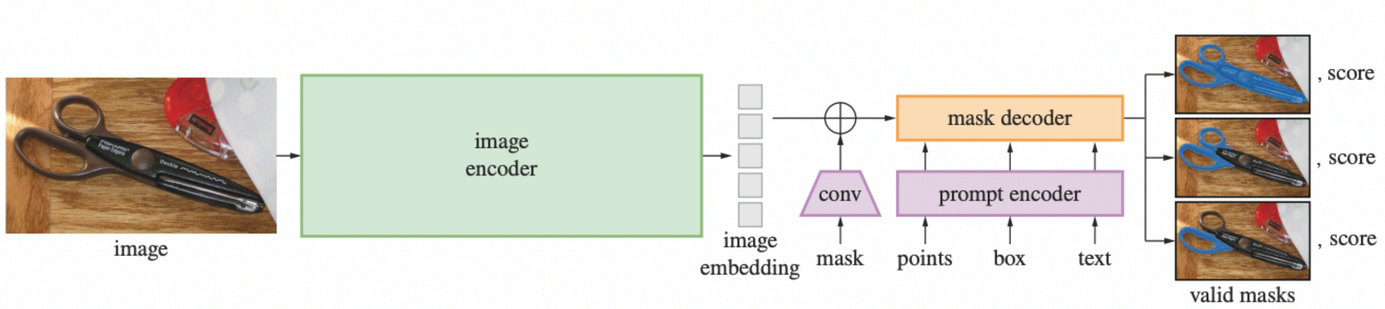
1. Image Encoder (ViT)

2. Prompt encoder (encode points / masks)

3. Mask decoder

Dataset of 11 million images and 1B masks (but most not generated by humans)





Advanced Visual Perception

Table of Contents

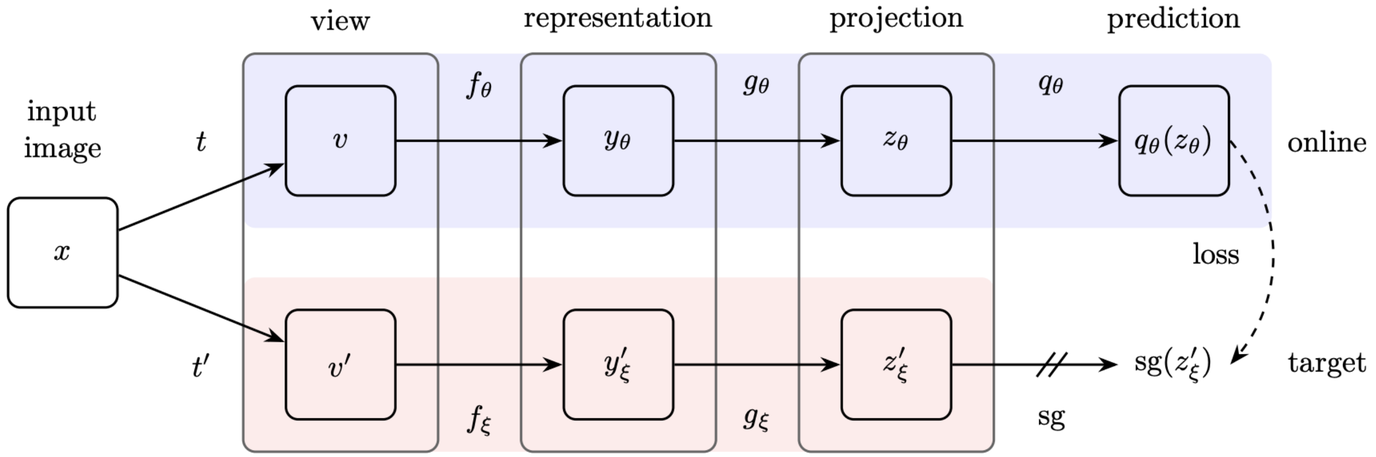
Contrastive Learning

- Use some “pretext task” to learn a representation
- Often works well in conjunction with a **contrastive loss** (e.g. in SimCLR)

$$l_{i,j} = -\log \frac{\exp\left(\text{sim}\left(\mathbf{z_i}, \mathbf{z_j}\right)/\tau\right)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp\left(\text{sim}\left(z_i, \mathbf{z_k}\right)/\tau\right)}$$

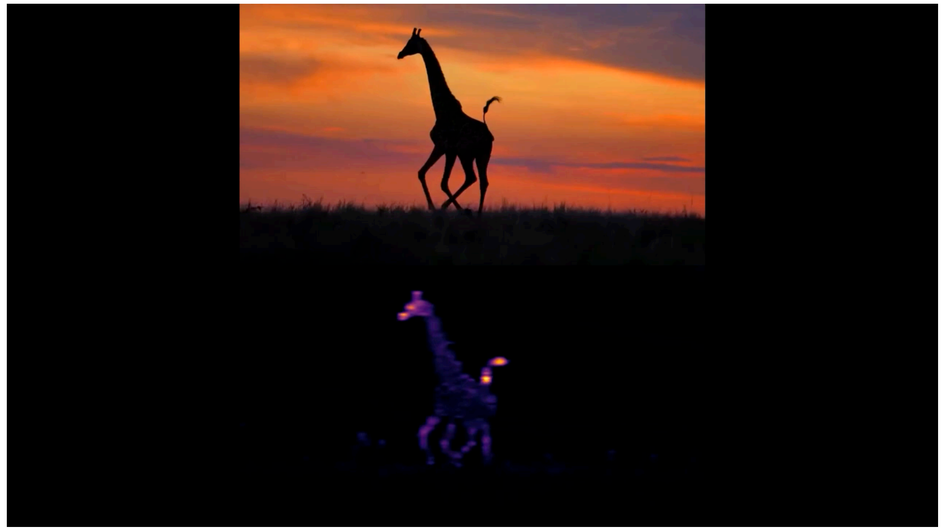
Self-distillation

- Use some “pretext task” to learn a representation
- Or through the process of knowledge distillation (e.g. in BYOL)

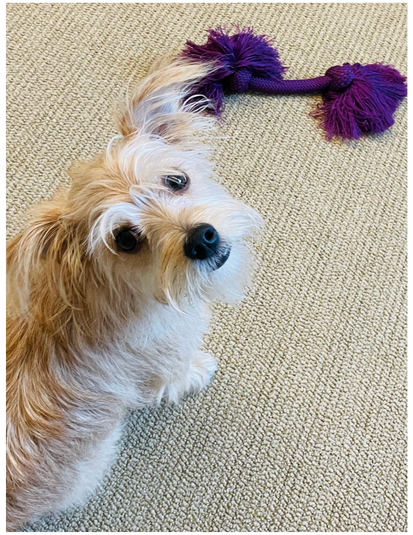


Self-Supervision + Transformer

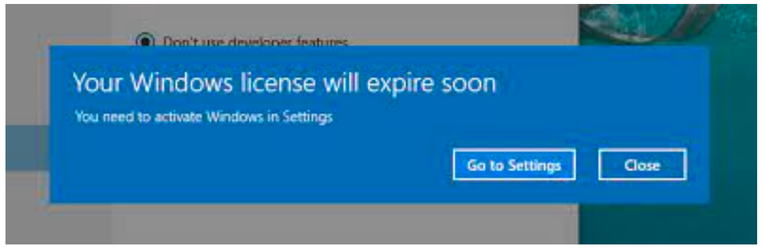
DINO



Multi-modal embeddings - CLIP



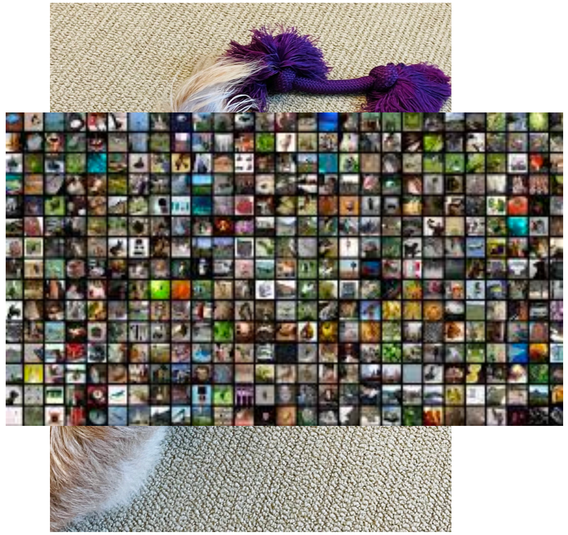
Tucker, the pup, looking up at the camera momentarily



A computer screen with a Windows license message about expiry

A digital thermometer showing 160 F

Multi-modal embeddings - CLIP



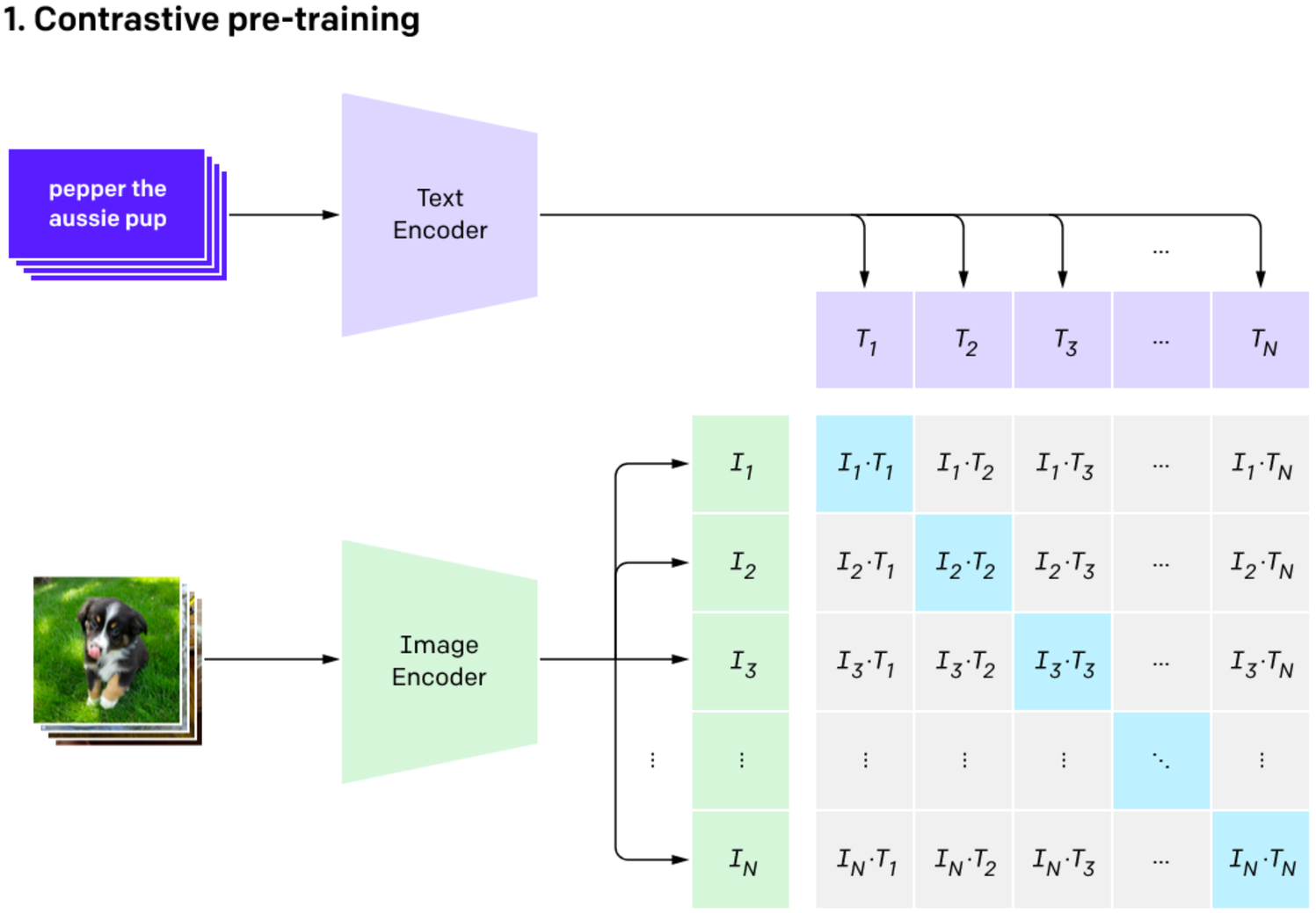
Vision transformer

I_1 I_2 I_3 I_N

Language transformer

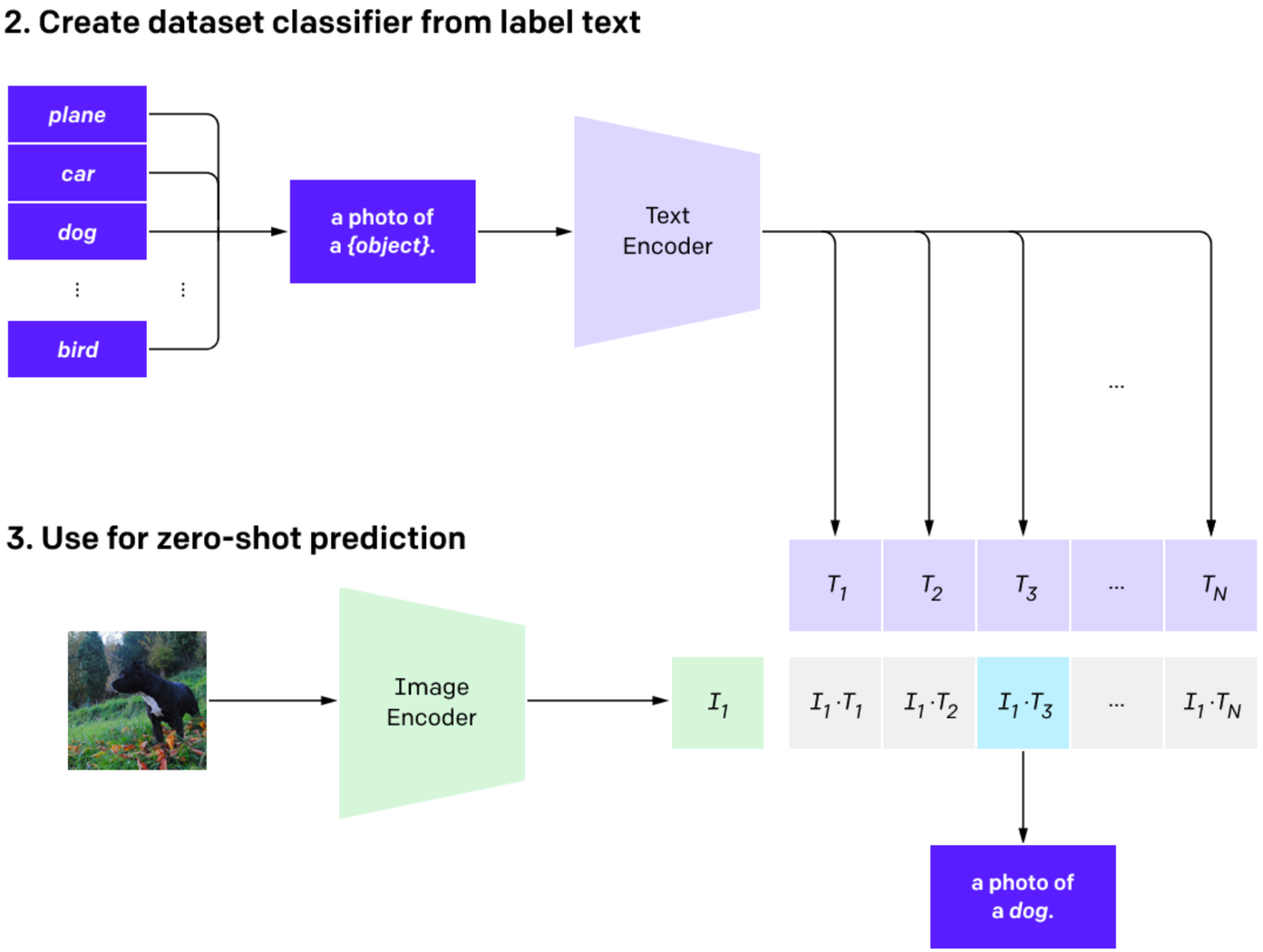
T_1 T_2 T_3 T_N

Multi-modal embeddings - CLIP



https://openai.com/blog/clip/

Multi-modal embeddings - CLIP



https://openai.com/blog/clip/