University of Zurich UZH | ETH zürich

Institute of Informatics – Institute of Neuroinformatics

ROBOTICS & PERCEPTION GROUP

Deep Learning

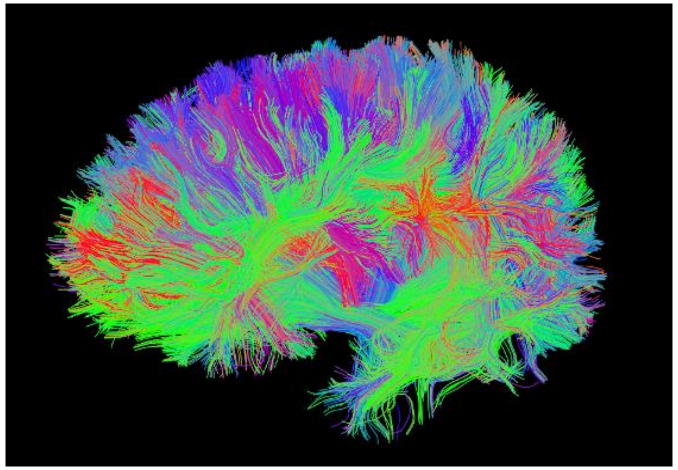
Daniel Gehrig

Outline

$\}$ Relevant for the exam

The Deep Learning Revolution

Medicine

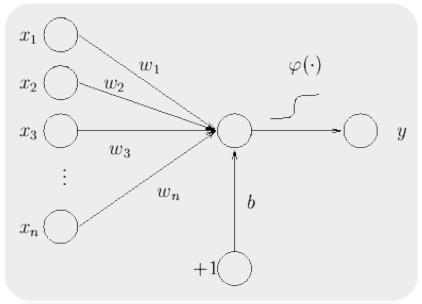


Media & Entertainment

Surveillance & Security

Autonomous Driving

Some History

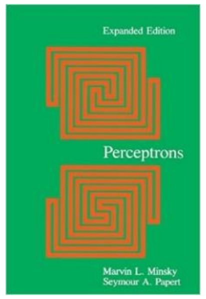


Perceptron

1958

1969

Critics


Expanded Edition

Perceptrons

Marvin L. Minsky
Seymour A. Papert

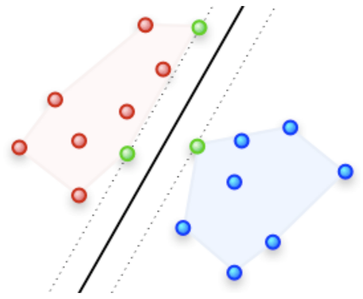
Back-Propagation

1974

AI Winter

1995

Convolutional Neural Networs for Handwritten Digits Recognition

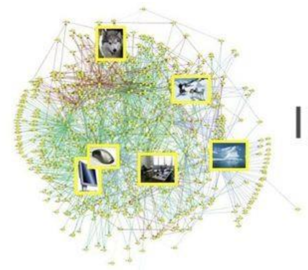


SVM

1998

2006

Restricted Boltzman Machines



IMAGENET

AlexNet

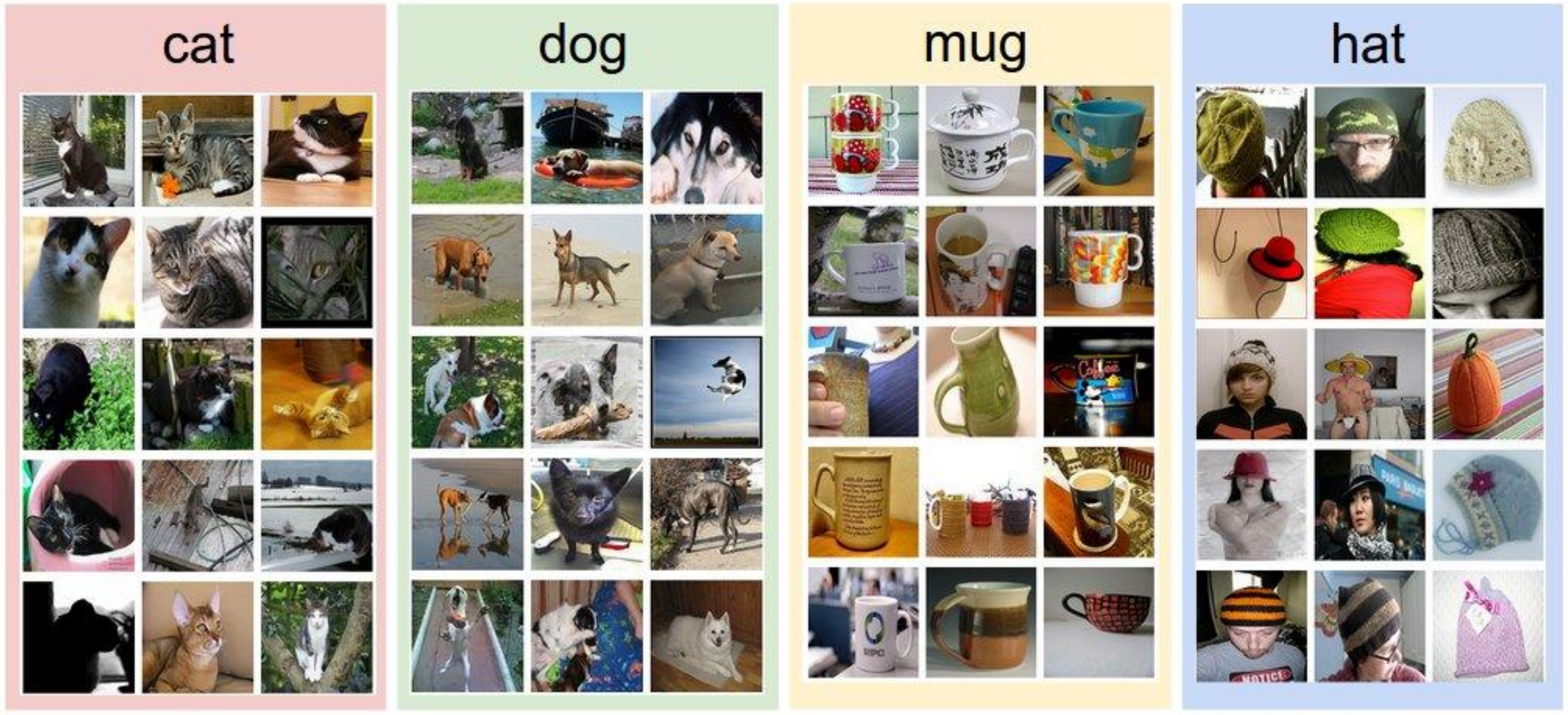
2012

What changed?

- Hardware Improvements
- Big Data Available
- Algorithmic Progress

Image Classification

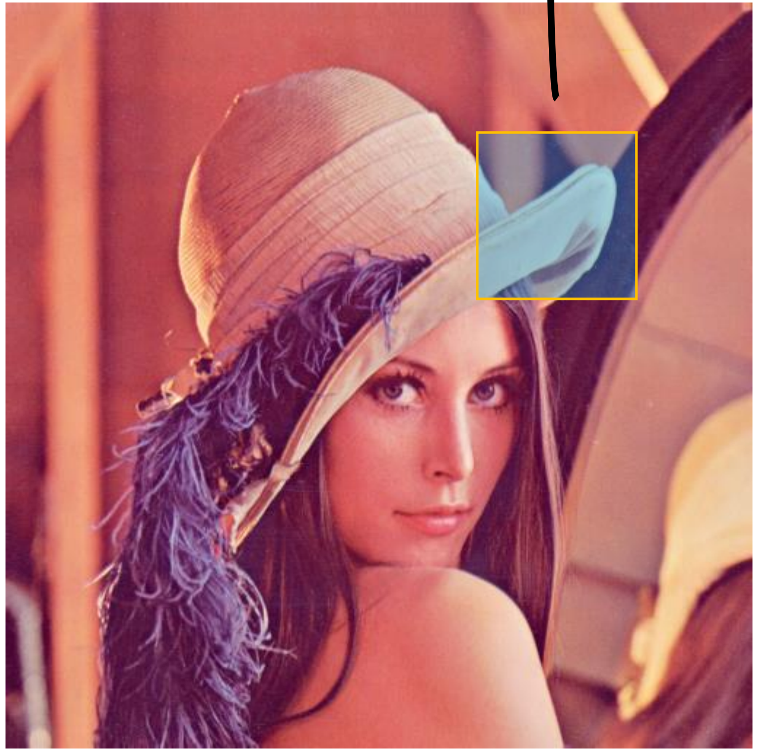
Task of assigning an input image a label from a fixed set of categories.

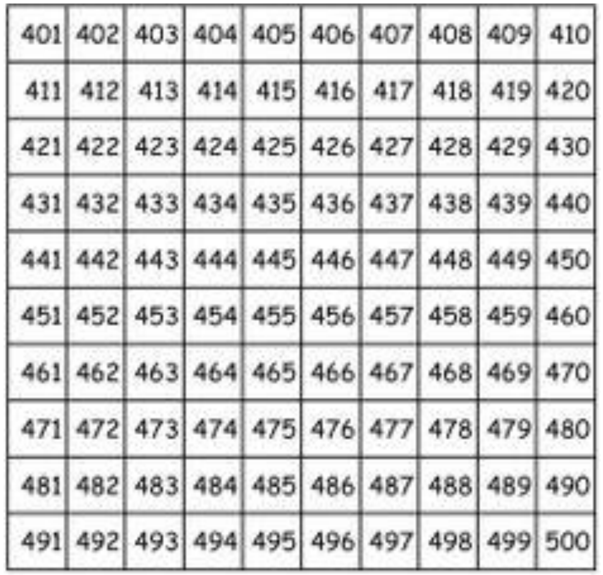


Slide adapted from CNNs for Visual Recognition (Stanford)

The semantic gap

- What computers see against what we see

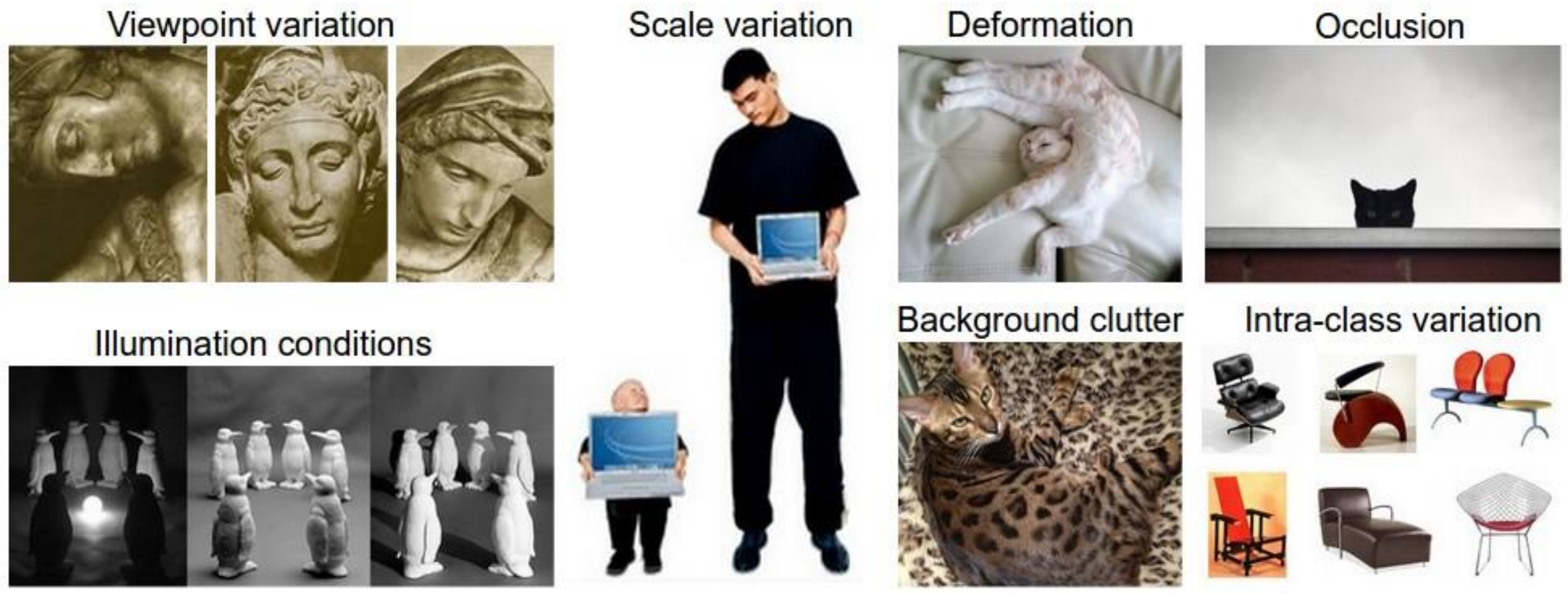




401	402	403	404	405	406	407	408	409	410
411	412	413	414	415	416	417	418	419	420
421	422	423	424	425	426	427	428	429	430
431	432	433	434	435	436	437	438	439	440
441	442	443	444	445	446	447	448	449	450
451	452	453	454	455	456	457	458	459	460
461	462	463	464	465	466	467	468	469	470
471	472	473	474	475	476	477	478	479	480
481	482	483	484	485	486	487	488	489	490
491	492	493	494	495	496	497	498	499	500

Classification Challenges

Directly specifying how a category looks like is impossible.



We need use a **Data Driven Approach**

Slide adapted from CNNs for Visual Recognition (Stanford)

Supervised Learning

Find function $f(x, \theta)$ that imitates a ground truth signal

$$f(x, \theta) \longrightarrow$$

N numbers representing class scores

Predicted

$$\begin{pmatrix} 0.1 \\ 0.7 \\ \cdots \\ 0.0 \end{pmatrix}$$

Ground truth, y_i

$$\begin{pmatrix} 1.0 \\ 0.0 \\ \cdots \\ 0.0 \end{pmatrix}$$

Function parameters or weights

Update

$$Loss(f(x_i, \theta), y_i)$$

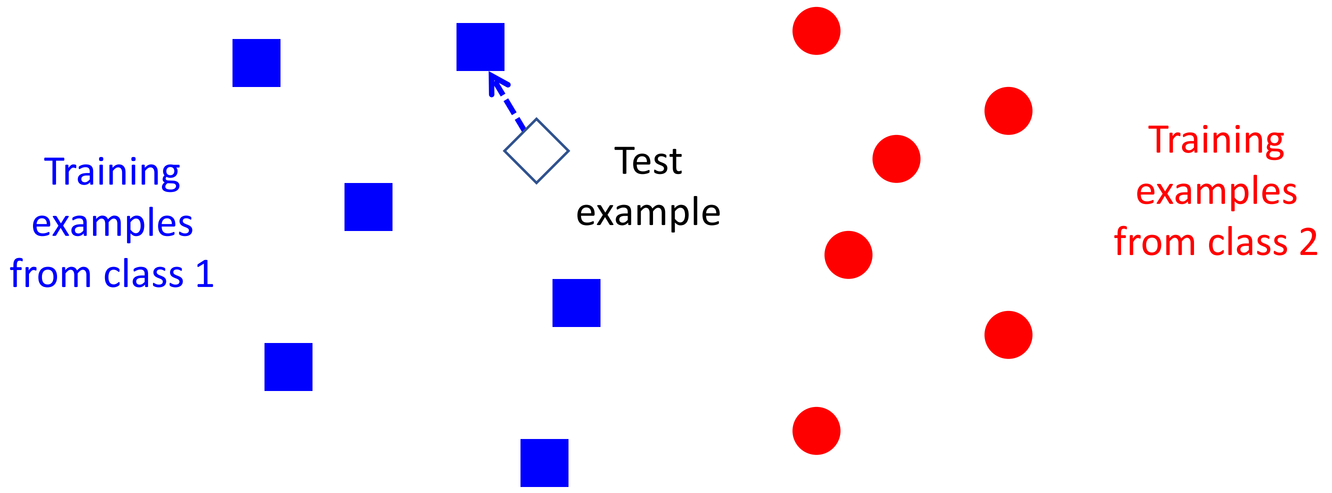
Machine Learning Keywords

- Loss: Quantify how good θ are
- Optimization: The process of finding θ that minimize the loss
- Function: Problem modelling -> Deep networks are highly non-linear $f(x, \theta)$

Slide adapted from CNNs for Visual Recognition (Stanford)

Classifiers: K-Nearest neighbor

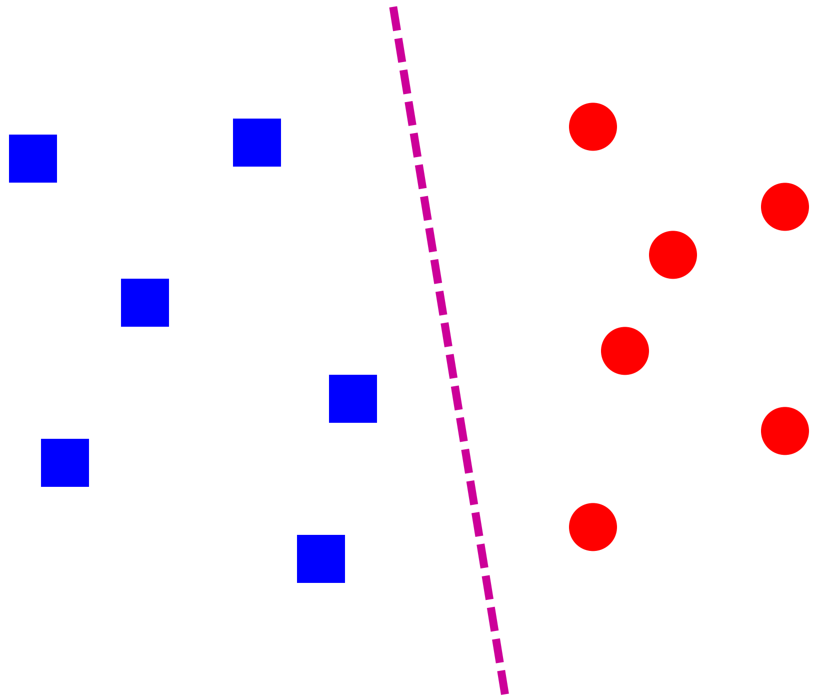
Features are represented in the descriptor space



$f(\boldsymbol{x}, \theta)$ = label of the K training examples nearest to $\boldsymbol{x}$

- How fast is training? How fast is testing?
  - O(1), O(n)
- What is a good distance metric ? What K should be used? ☹

Classifiers: Linear



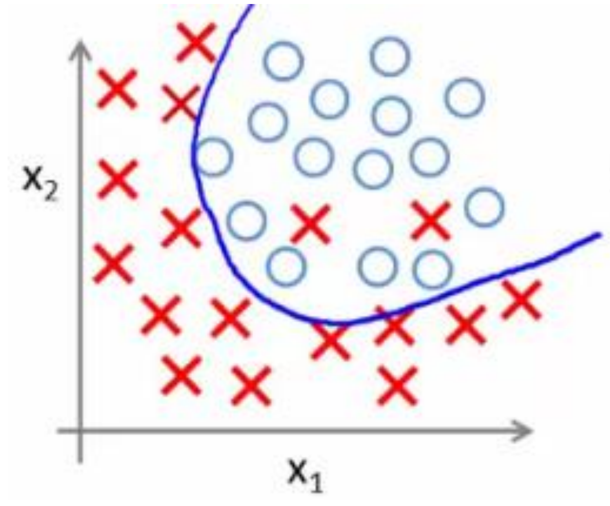
- Find a *linear function* to separate the classes:

$$f(\boldsymbol{x}, \theta) = sgn(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

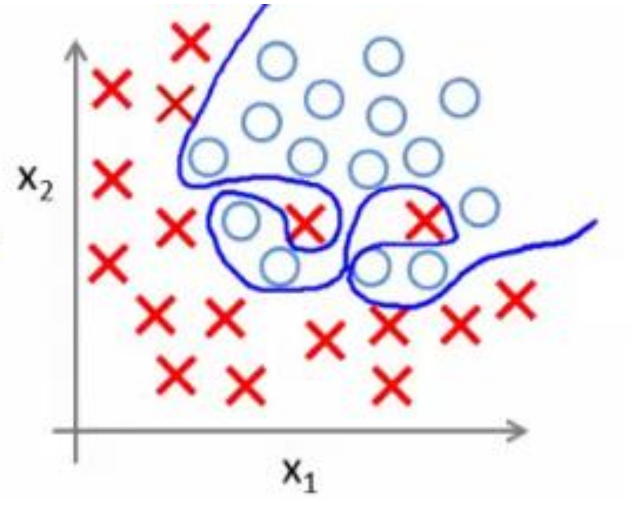
- What is now $\boldsymbol{\theta}$? What is the dimensionality of images?

Classifiers: non-linear

Good classifier

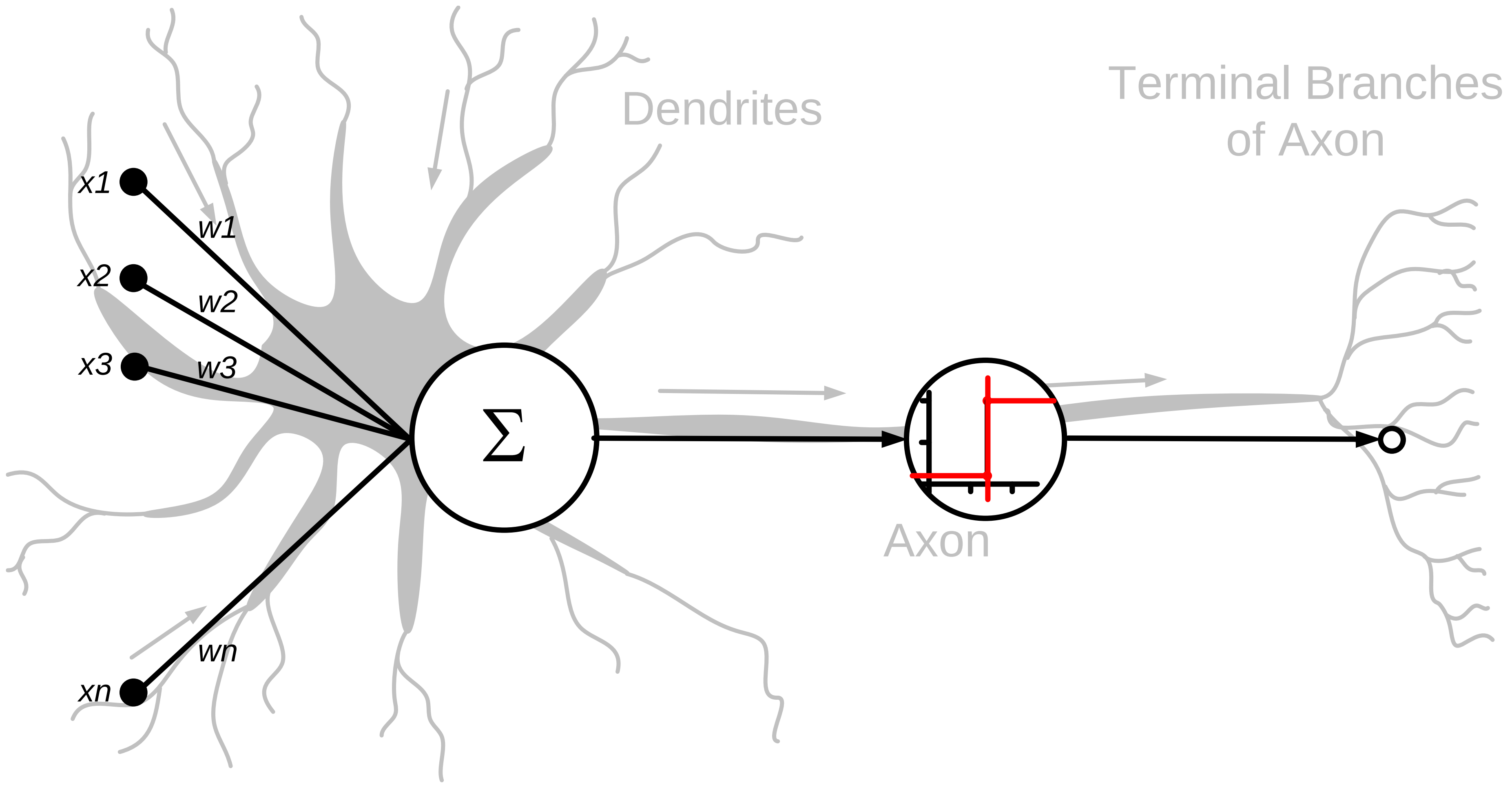


Bad classifier (over fitting)



- What is $f(\boldsymbol{x}, \theta)$?

Biological Inspiration

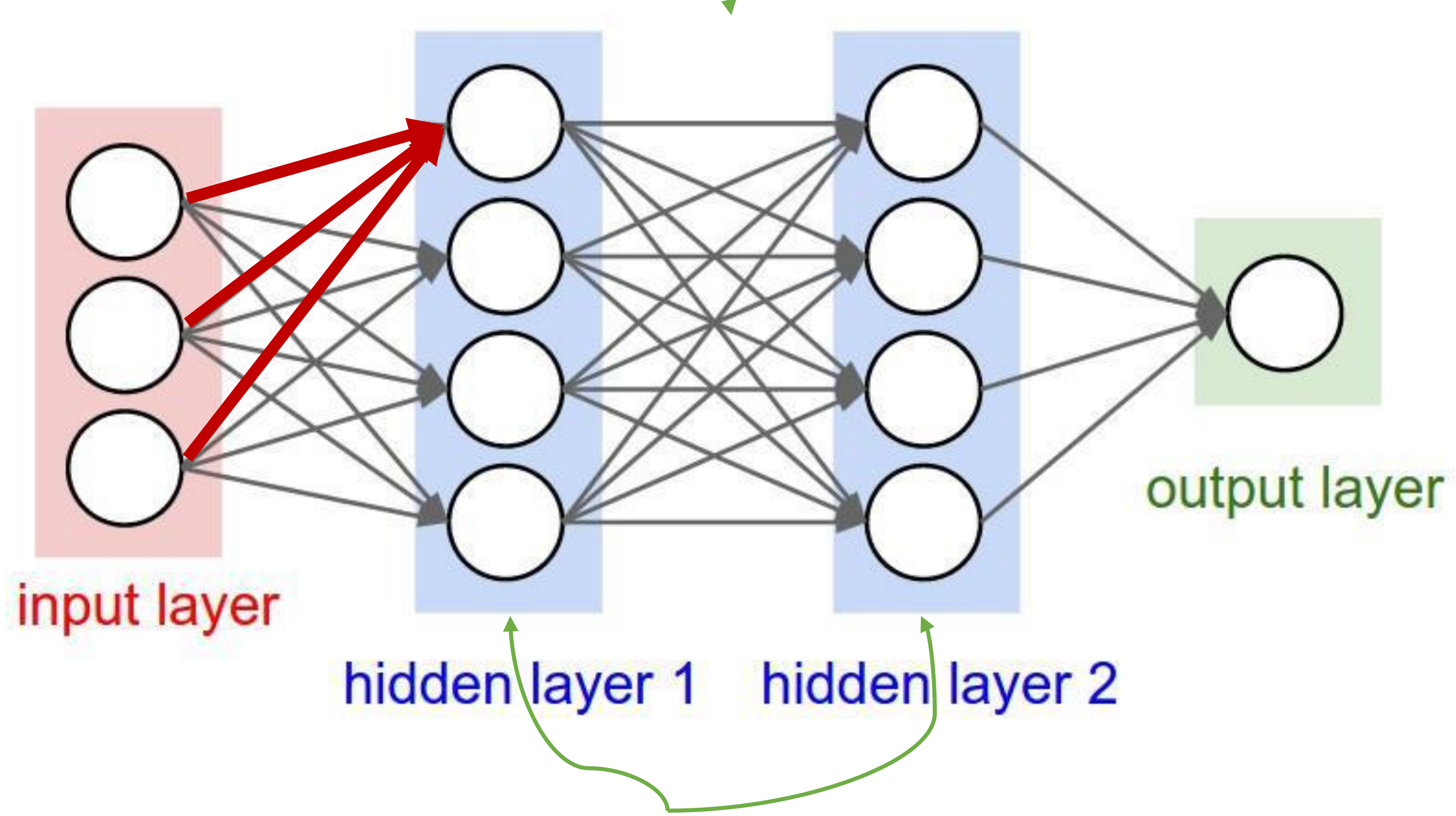


f$(x, \theta) = F(Wx)$, F is a non-linear activation function (Step, ReLU, Sigmoid)

The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Frank Rosenblatt (1958)

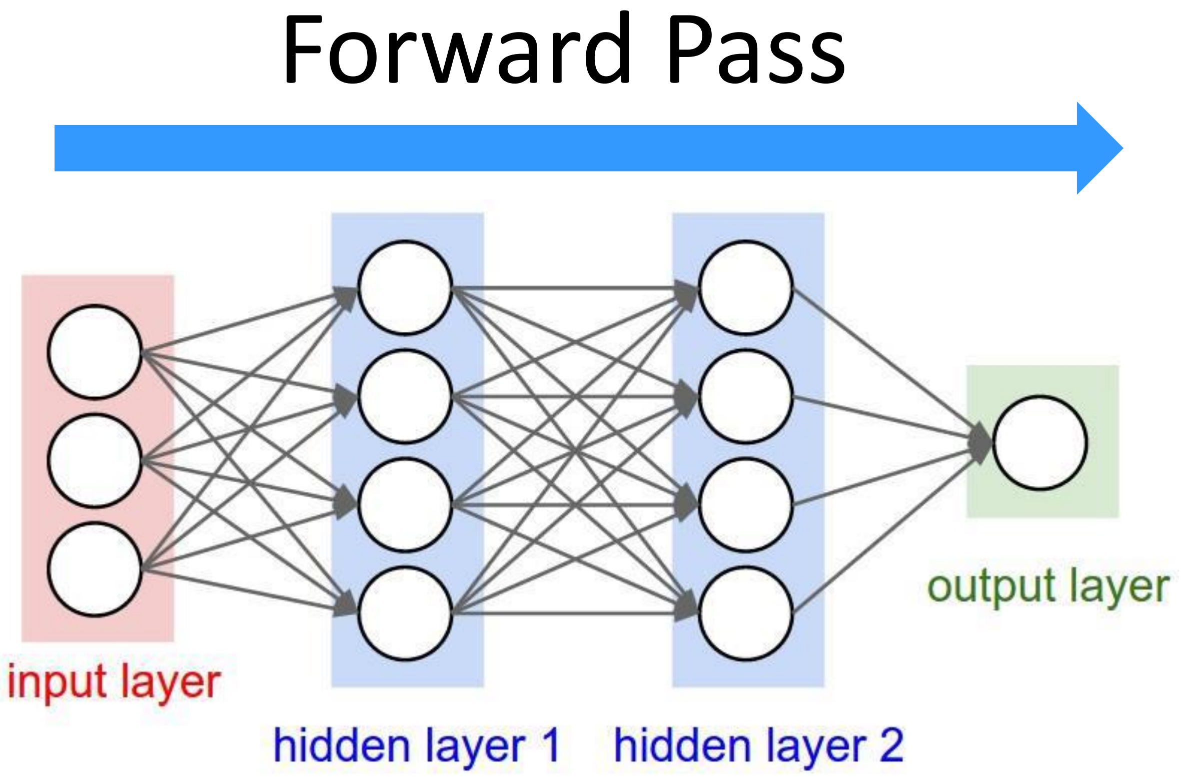
Multi Layer Perceptron

$\boldsymbol{f}\,(\boldsymbol{x}, \boldsymbol{\theta})$



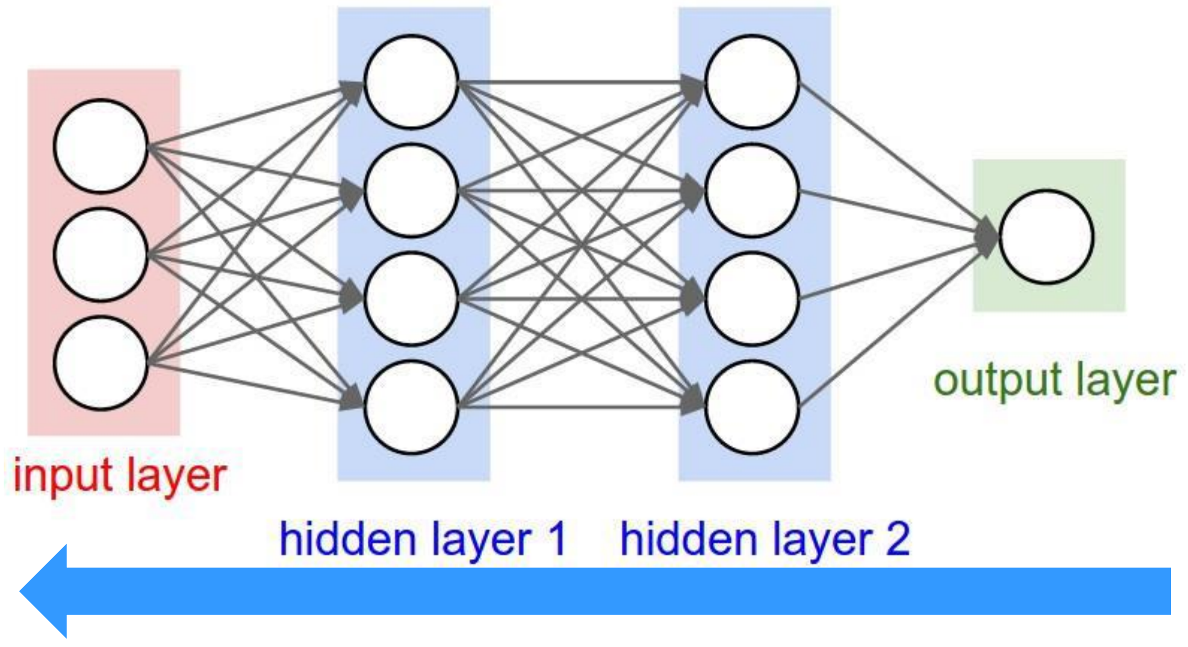
Non-linear Activation functions (ReLU, sigmoid, etc.)

Forward propagation

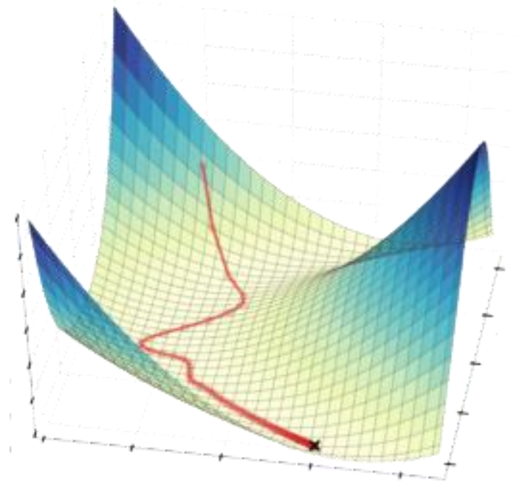


$Loss(f(x_i, \theta), y_i)$

Optimization: Back-propagation



$$Loss(f(x_i, \theta), y_i)$$



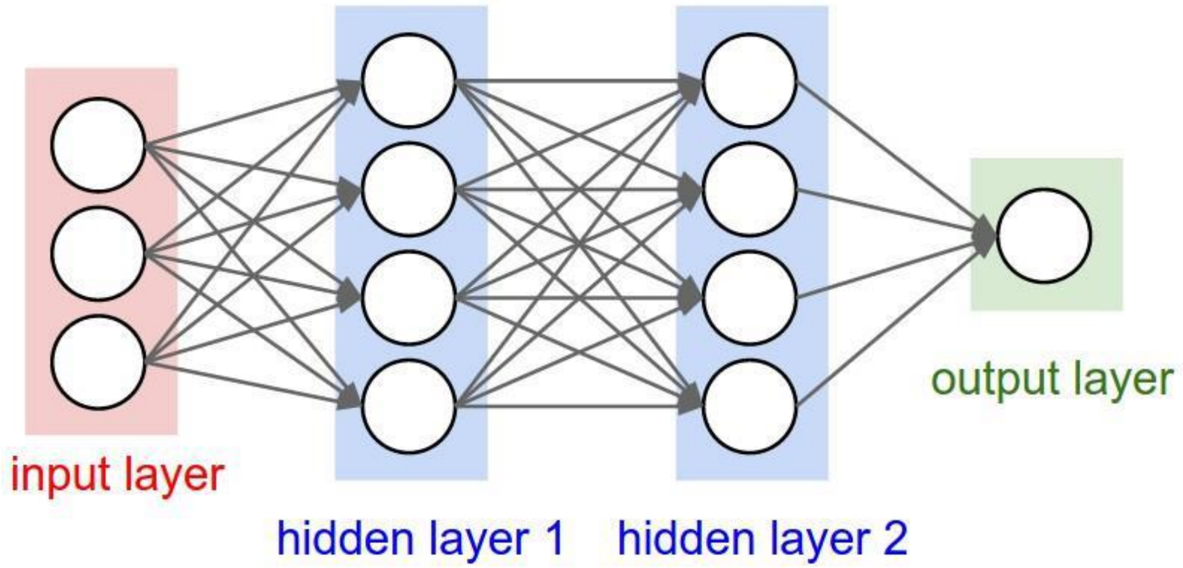
Backward Pass

Compute gradients with respect to all parameters and perform **gradient descent**

$$\theta_{new} = \theta_{old} - \mu \nabla Loss$$

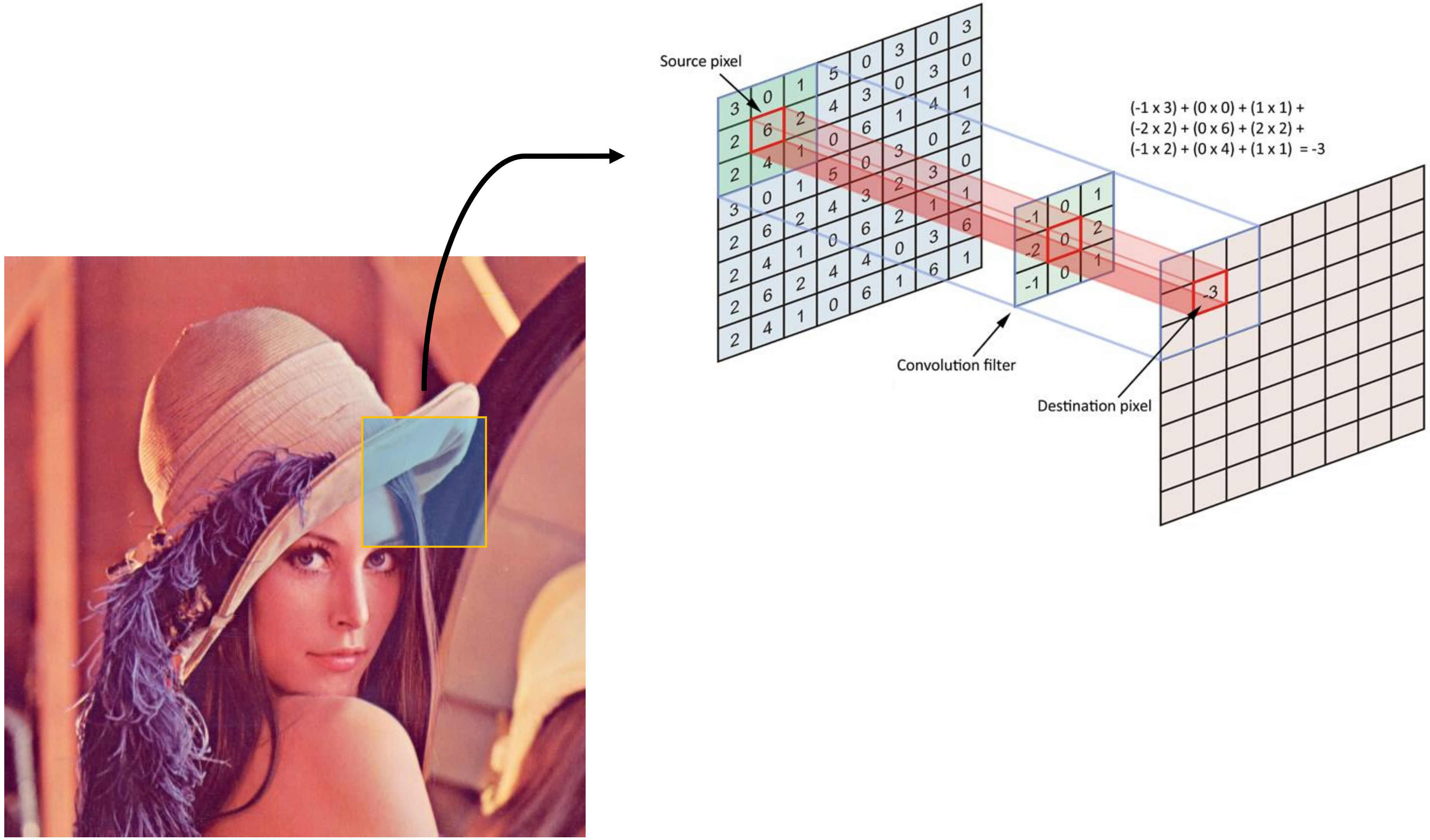
Artificial Neural Networks, Back Propagation and the Kelley-Bryson Gradient Procedure – Stuart E .Dreyfus

Problems of fully connected network



- Too many parameters -> possible overfit
- We are not using the fact that inputs are images!

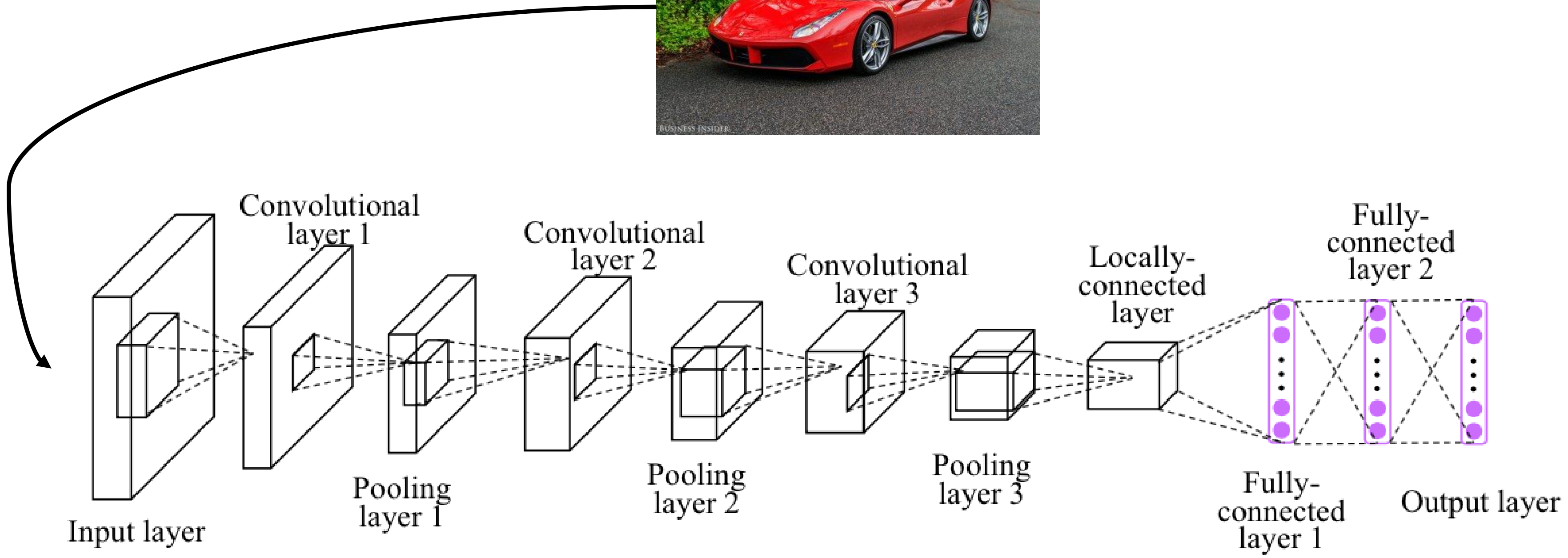
Convolutional Neural Networks

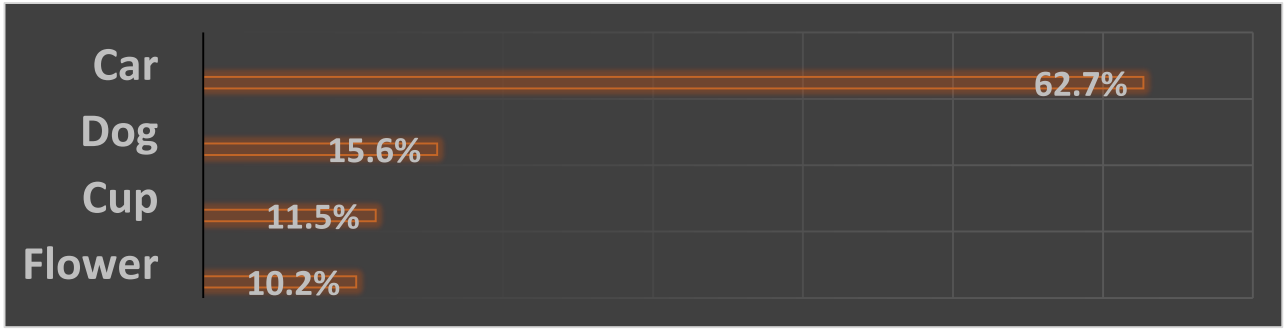

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel

Gradient-based learning applied to document recognition, Y. LeCun et al. (1998)

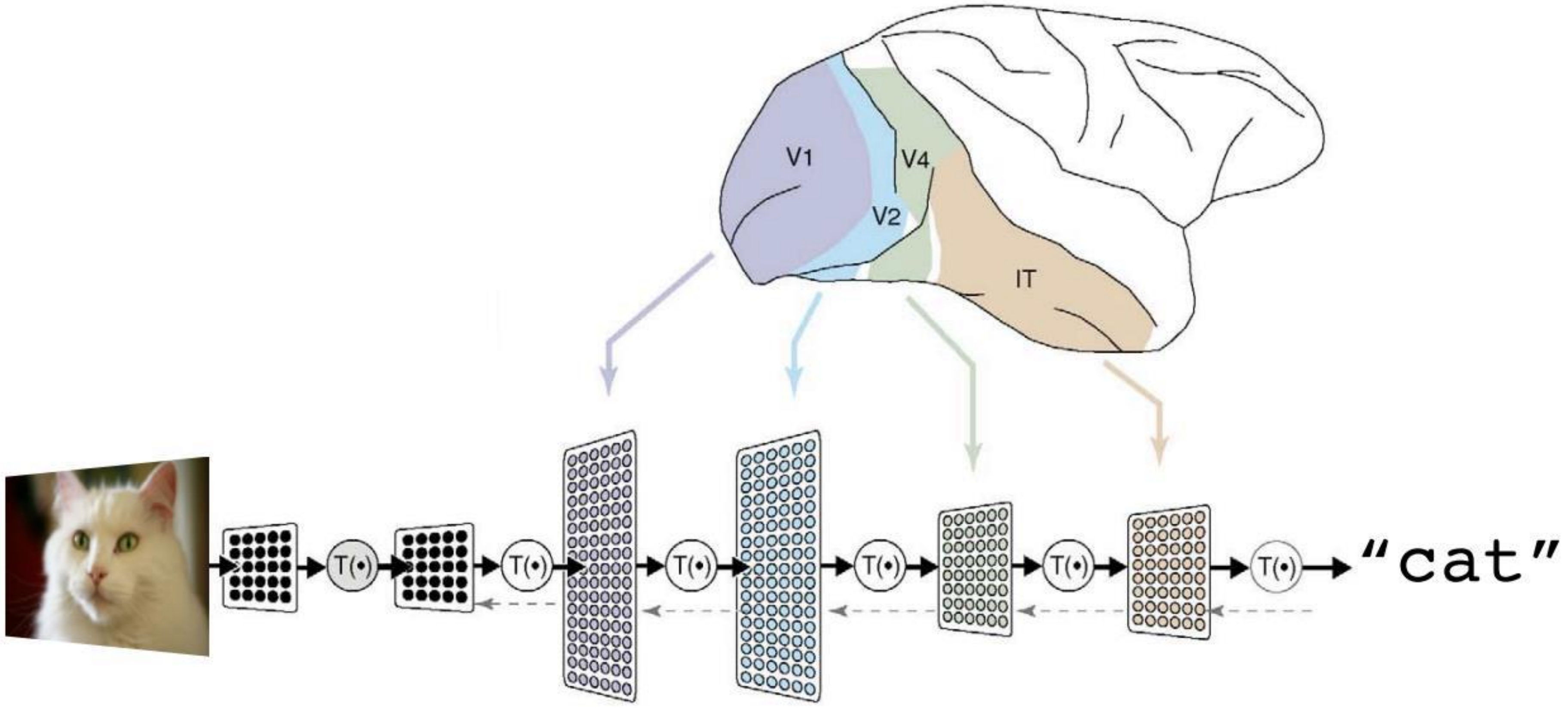
Going Deep

BUSINESS INSIDER


Input layer
Convolutional layer 1
Pooling layer 1
Convolutional layer 2
Pooling layer 2
Convolutional layer 3
Pooling layer 3
Locally-connected layer
Fully-connected layer 1
Fully-connected layer 2
Output layer


Car
62.7%
Dog
15.6%
Cup
11.5%
Flower
10.2%

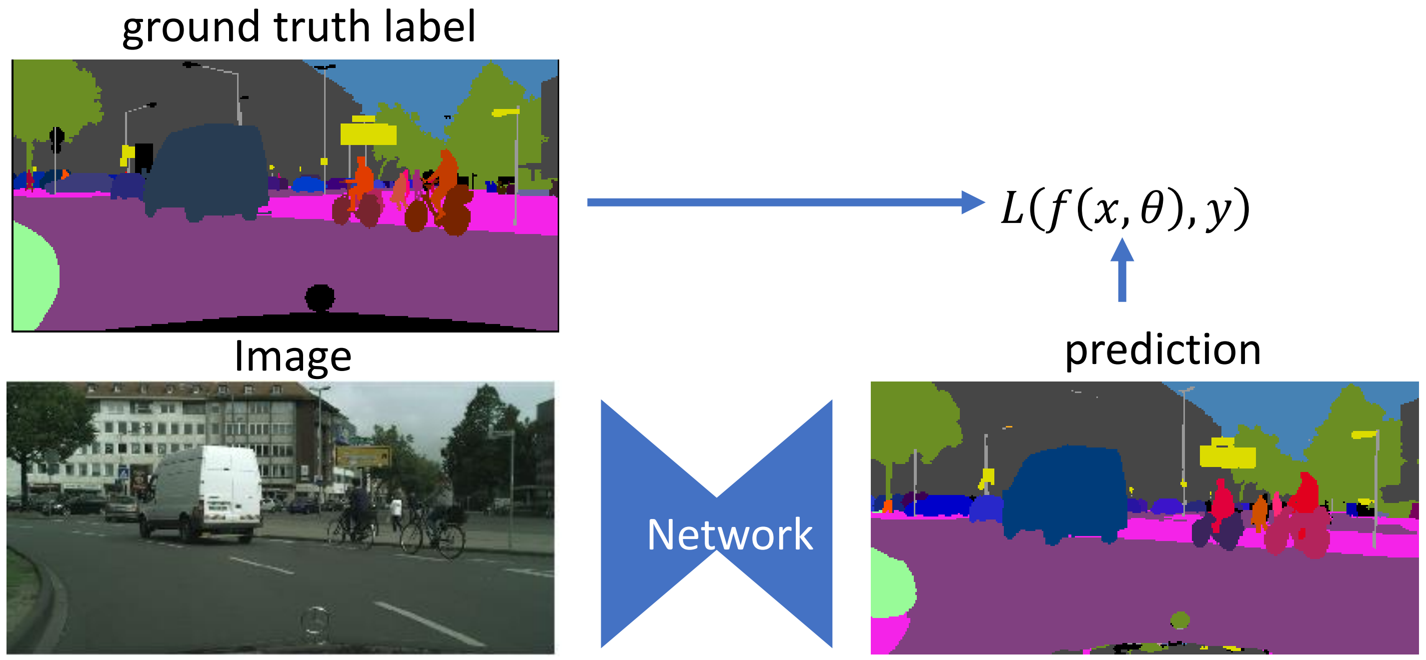
Why Deep?



- Inspired by the **human visual system**
- Learn **multiple layers** of transformations of input
- Extract progressively more **sophisticated representations**

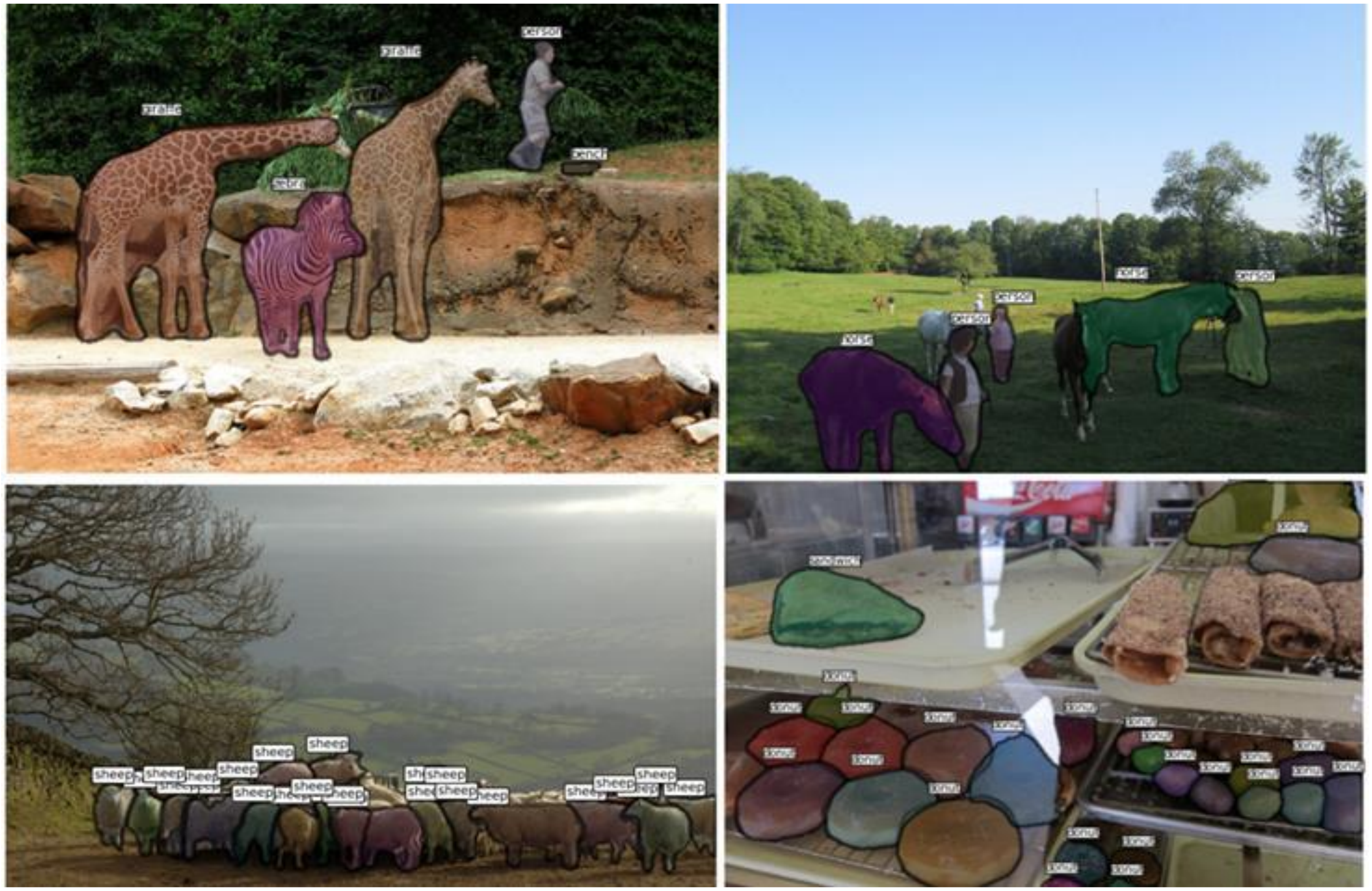
Supervised Learning

- In supervised learning it is assumed that we have access to both input data or **images** and **ground truth labels**.
- Networks trained with supervision usually perform best
- However, usually it is hard to generate this data since it often has to be **hand-labelled**



Supervised Learning

Image Segmentation



Fully Convolutional Networks for Semantic Segmentation – J. Long, E. Shelhamer, 2015

Supervised Learning

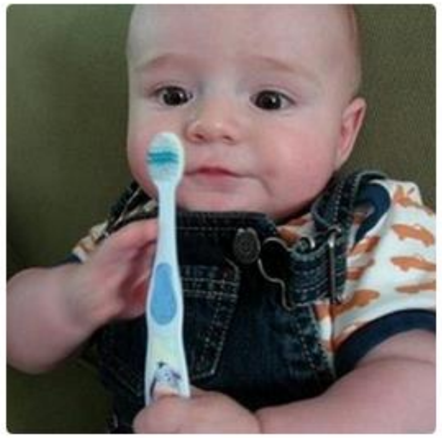
Image Captioning

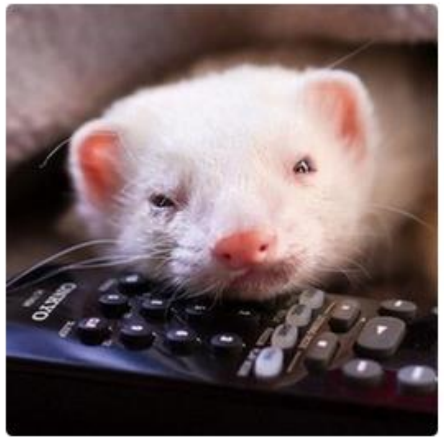
"little girl is eating piece of cake."

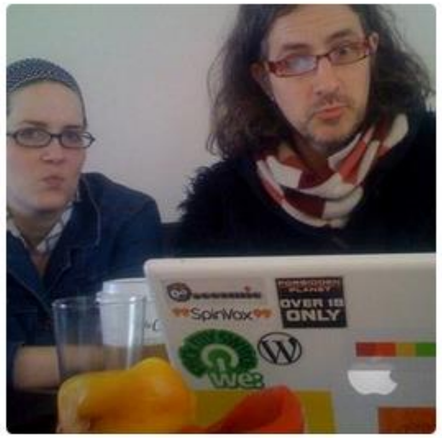
"baseball player is throwing ball in game."

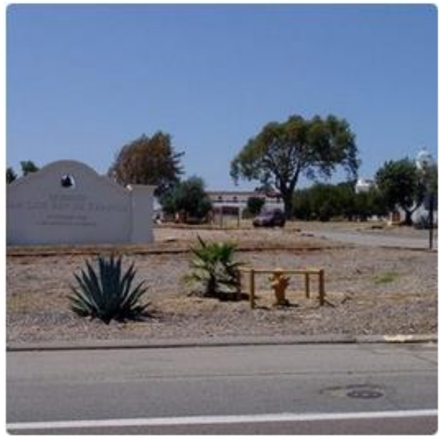
"woman is holding bunch of bananas."

"black cat is sitting on top of suitcase."


"a young boy is holding a baseball bat."


"a cat is sitting on a couch with a remote control."


"a woman holding a teddy bear in front of a mirror."


"a horse is standing in the middle of a road."

Deep Visual-Semantic Alignments for Generating Image Descriptions – Karpathy et al., 2015

Supervised Learning

Image Localization

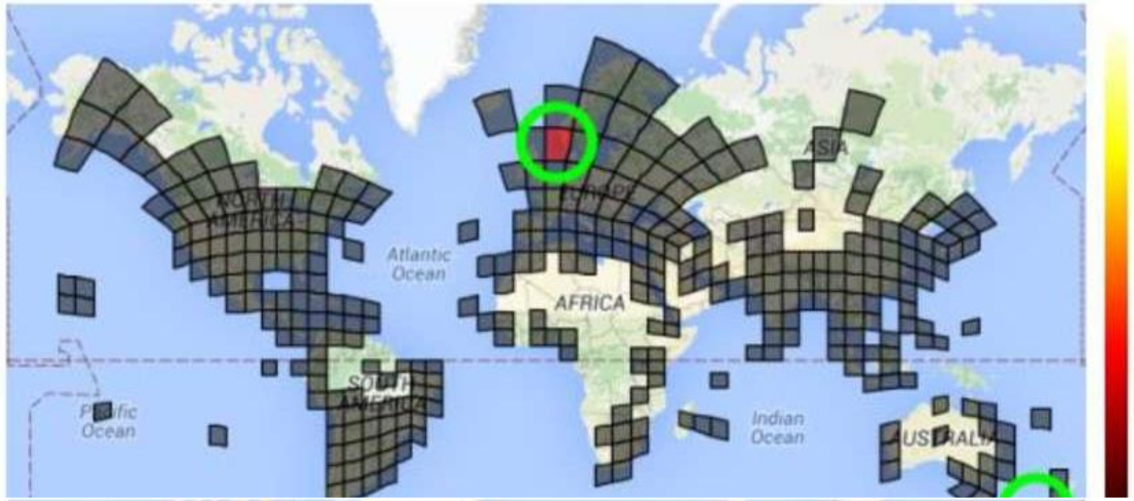
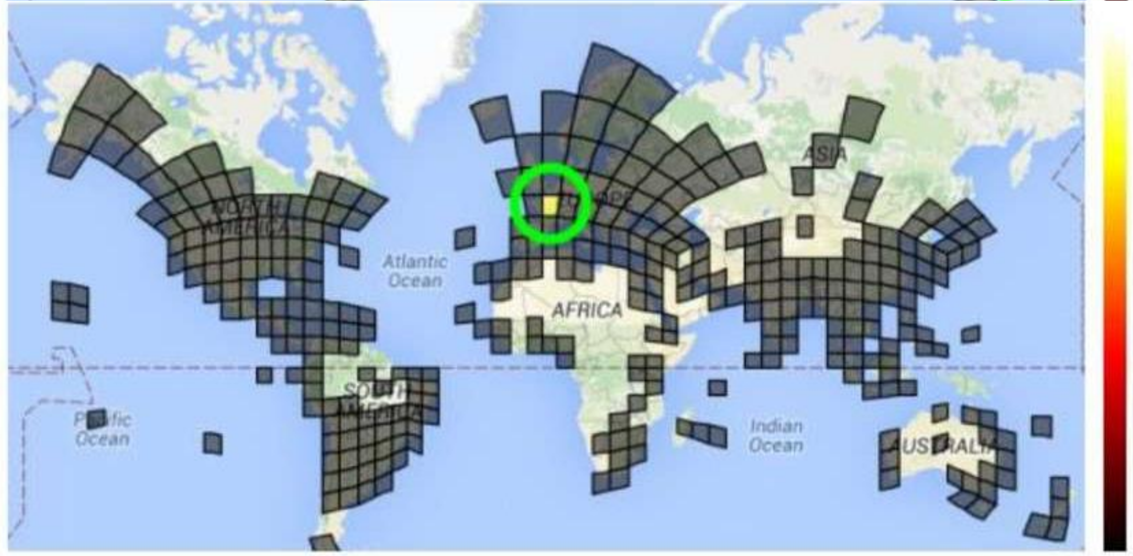


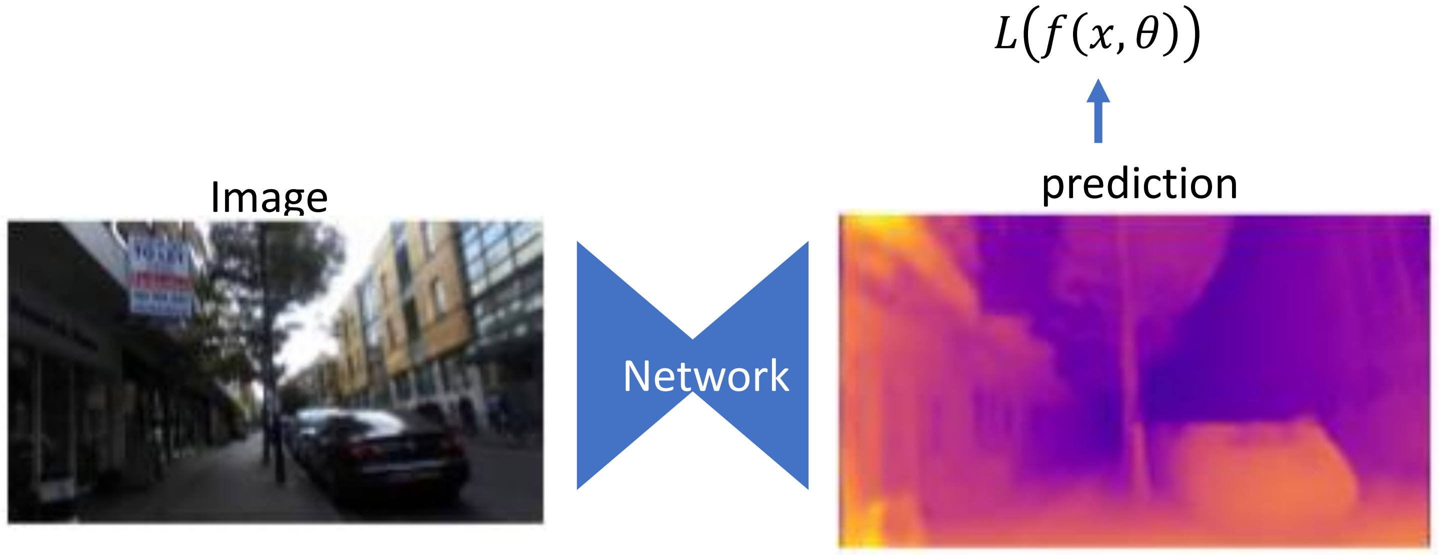
Photo CC-BY-NC by stevekc



PlaNet - Photo Geolocation with Convolutional Neural Networks - Weyand et al. 2016

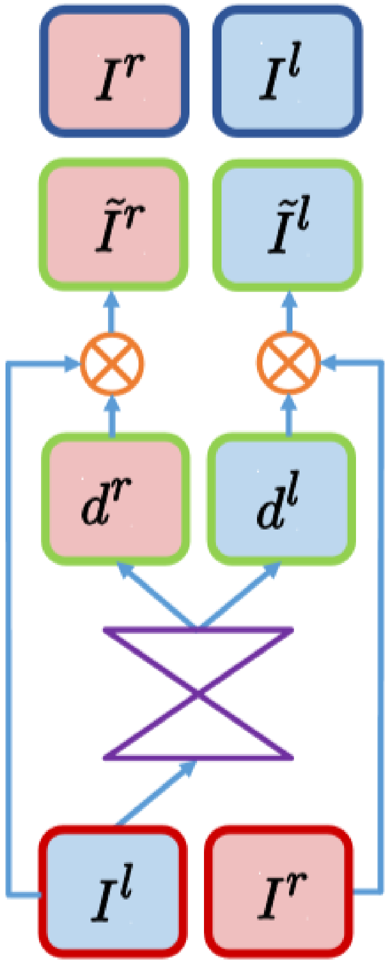
Unsupervised Learning

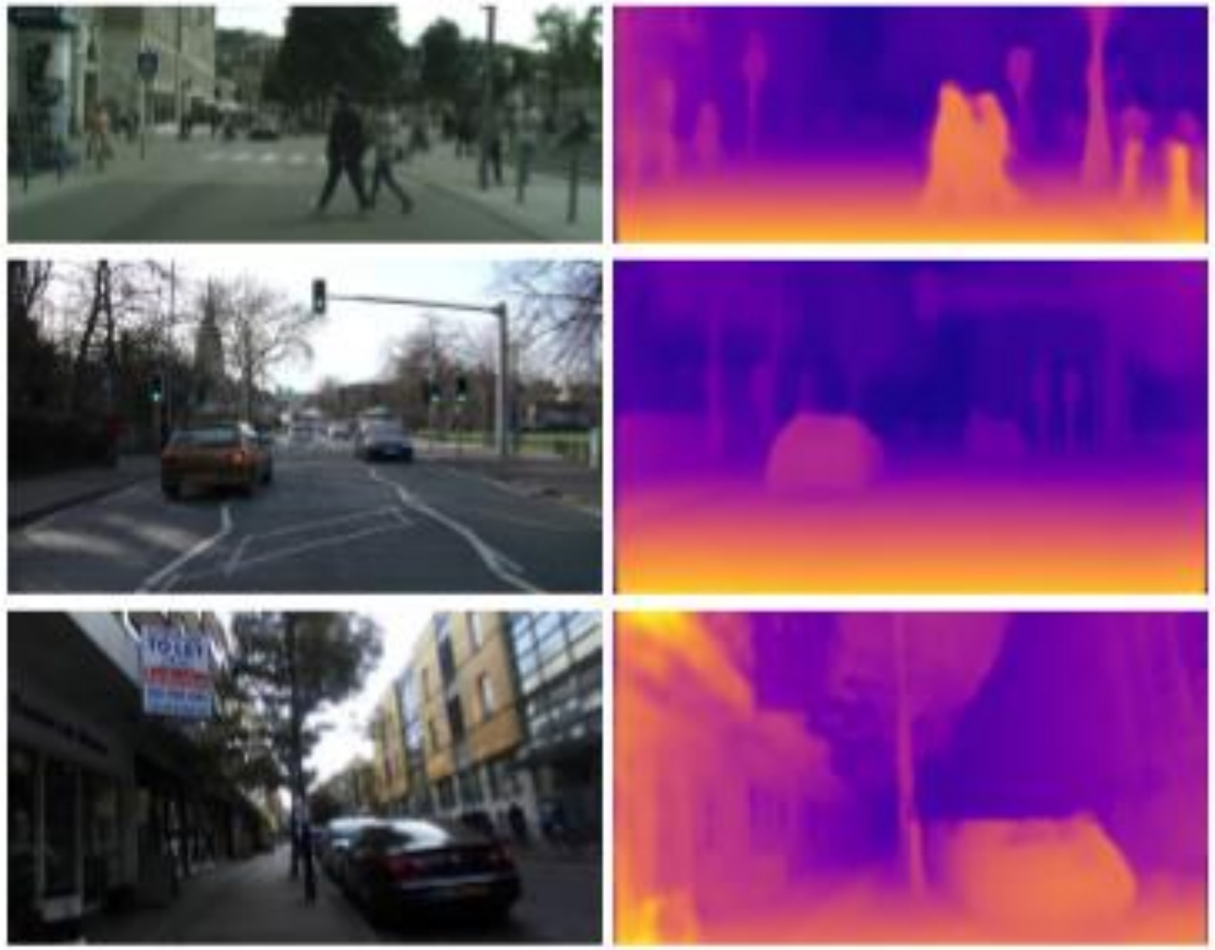
- In unsupervised learning we only have access to input data or **images**.
- Usually, these methods are more popular because they can use much larger datasets that do not need to be manually labelled.



Unsupervised Learning

Monocular Depth Estimation



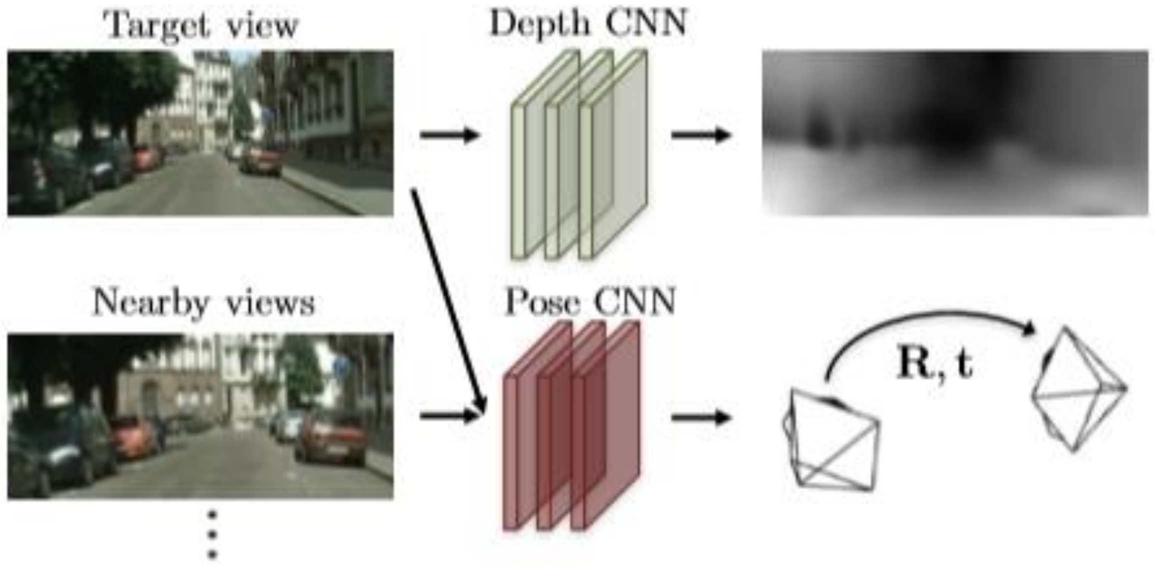


Unsupervised Monocular Depth Estimation with Left-Right Consistency, Godard et al. 2017

Unsupervised Learning

Structure from Motion

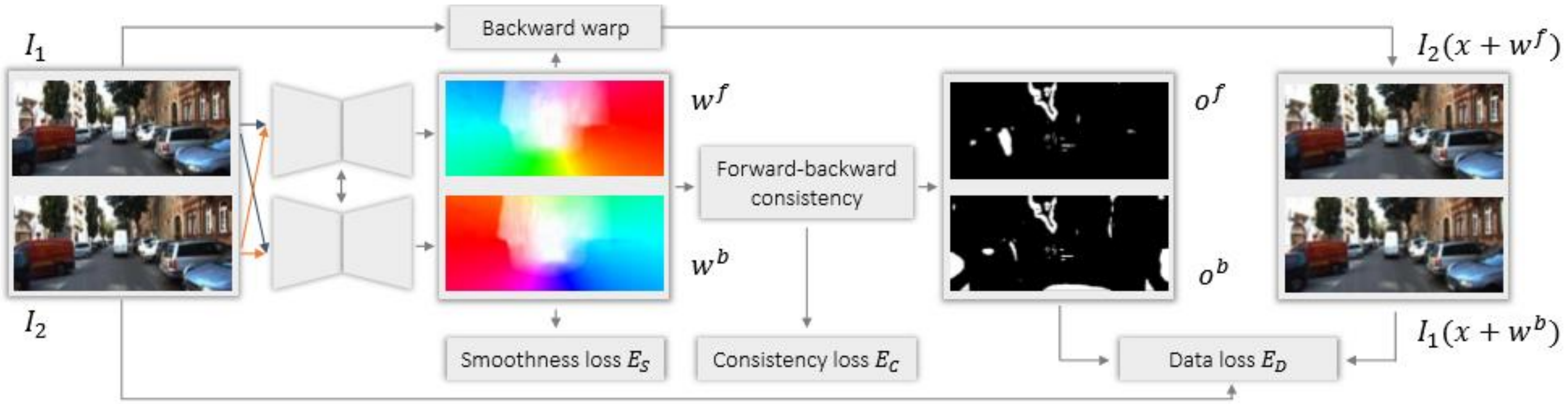
(a) Training: unlabeled video clips.



Unsupervised Learning of Depth and Ego-Motion from Video, Zhou et al. 2017

Unsupervised Learning

Dense Optical Flow



Characteristic of the learned flow:

- Robustness against light changes (Census Transform)
- Occlusion handling (Bi-directional Flow)
- Smooth flow

UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss, Meister et al, 2018

Unsupervised vs. Supervised learning

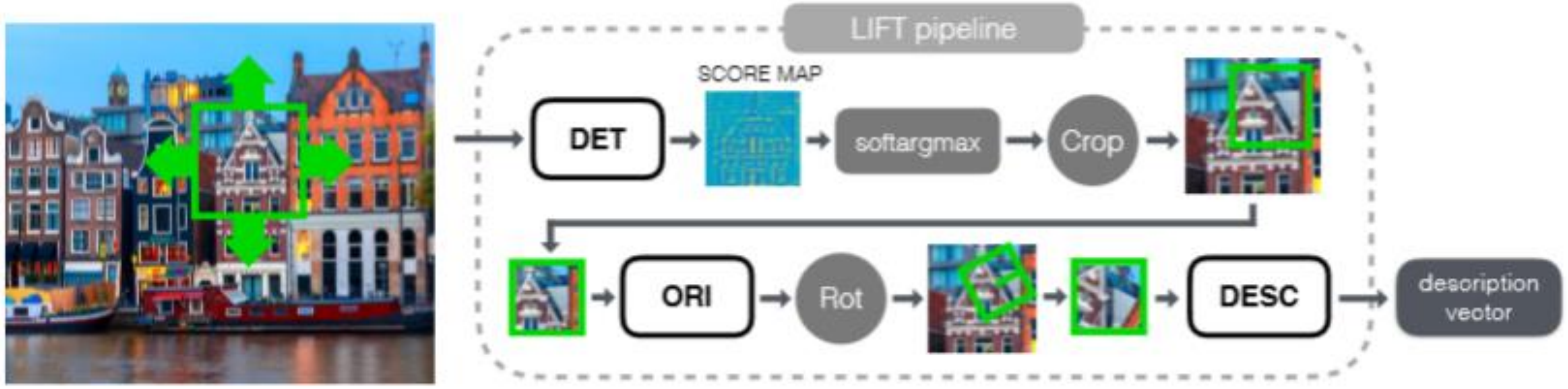
	Supervised	Unsupervised
Performance	Usually better for the same dataset size.	Usually worse, but can outperform supervised methods due to larger data availability.
Data availability	Low, due to manual labelling.	High, no labelling required.
Training	Simple, ground truth gives a strong supervision signal.	Sometimes difficult, loss functions have to be engineered to get good results.
Generalizability	Good, although sometimes the network learns to blindly copy the labels provided, leading to poor generalizability.	Better, since unsupervised losses often encode the task in a more fundamental way.

Applications to Computer Vision

Applications to Computer Vision

Keypoint Detection and Description

Deep Descriptors: LIFT

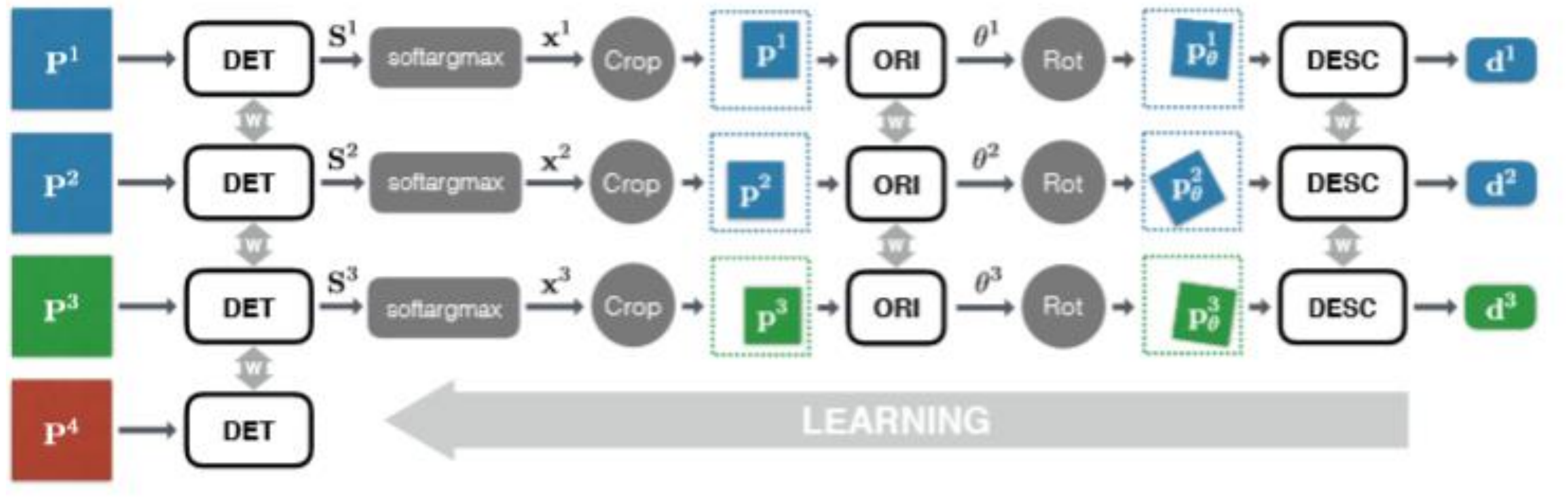


LIFT Pipeline consists of 3 neural networks:

- A keypoint detector
- An orientation detector
- A descriptor generator

LIFT: Learned Invariant Feature Transform, Kwang Moo Yi et al., 2016

LIFT Loss

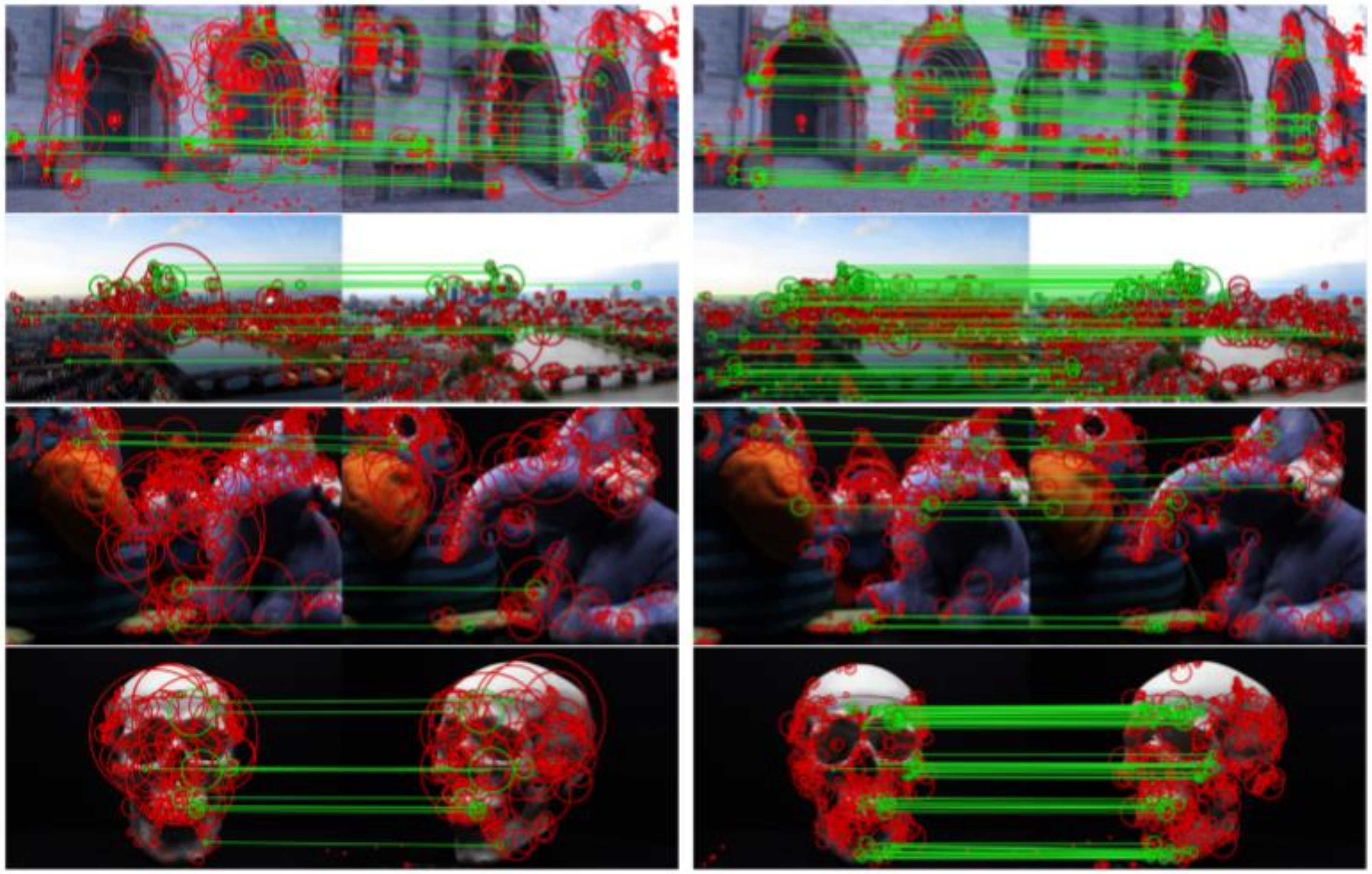


LIFT Loss has 3 components:

- Distance between descriptors of correponding patches, d^1 d^2, that should be *small*
- Distance between descriptors of different patches, d^1 d^3, that should be *large*
- Keypoints should not be located in homogeneous regions: P4 should not be detected as a keypoint

LIFT Results

- Works better than SIFT! (well, in some datasets)

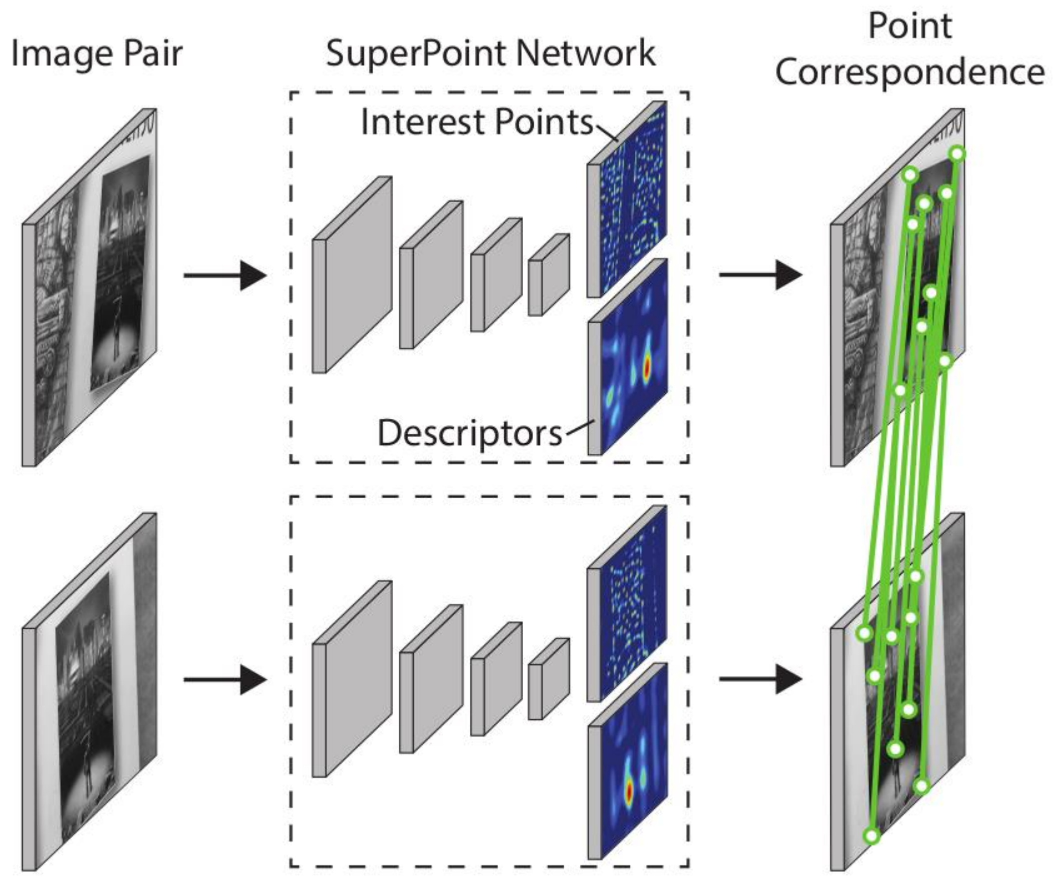


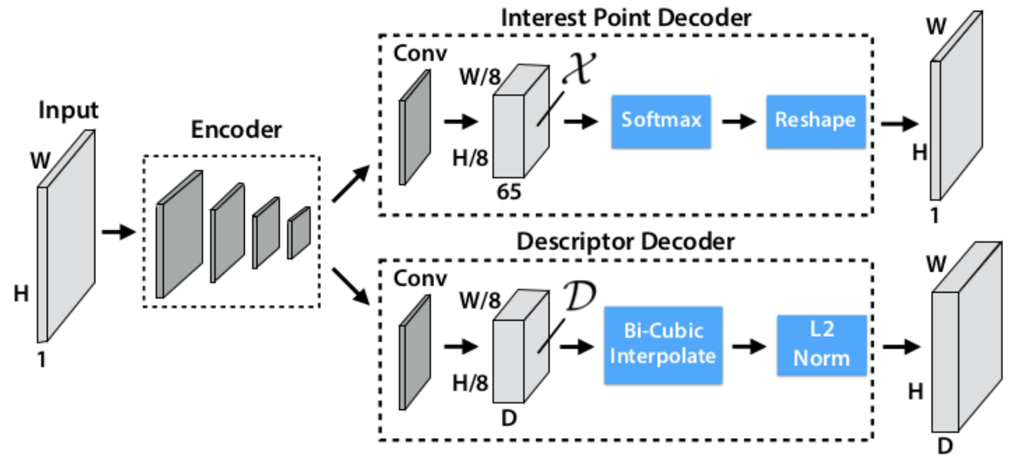
Applications to Computer Vision

Keypoint Detection and Description

SuperPoint: Self-Supervised Interest Point Detection and Description

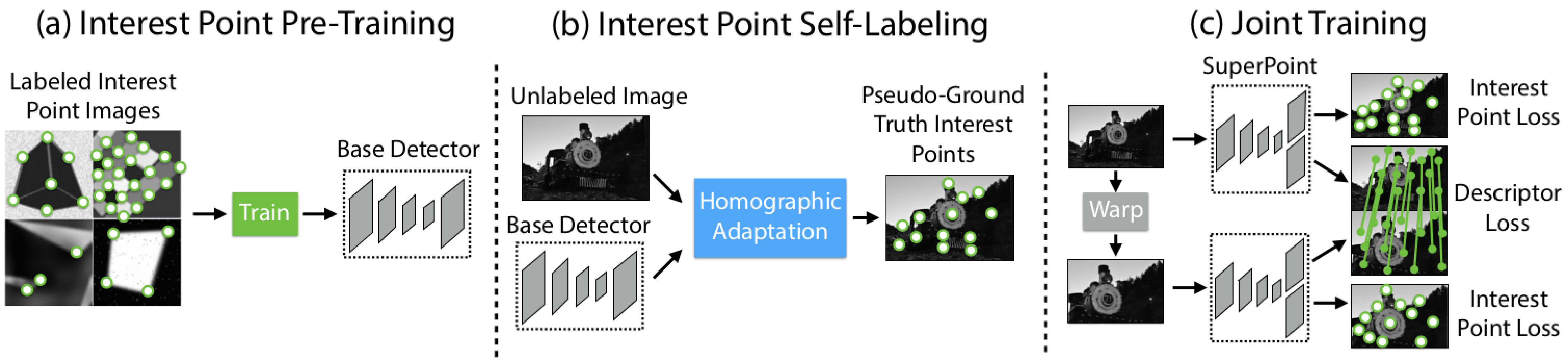
- SIFT and friends are complicated heuristic algorithms
- Still, SIFT is a hard to beat baseline for new methods
- Can we do better with a data driven approach?
- SuperPoint shows how a convolutional neural network can be trained to **predict keypoints and descriptors simultaneously**.
- Detector less accurate than SIFT, but descriptor shown to outperform SIFT in some scenarios.





SuperPoint: Self-Supervised Interest Point Detection and Description – D. Detone et al, CVPRW 2018

SuperPoint - Training

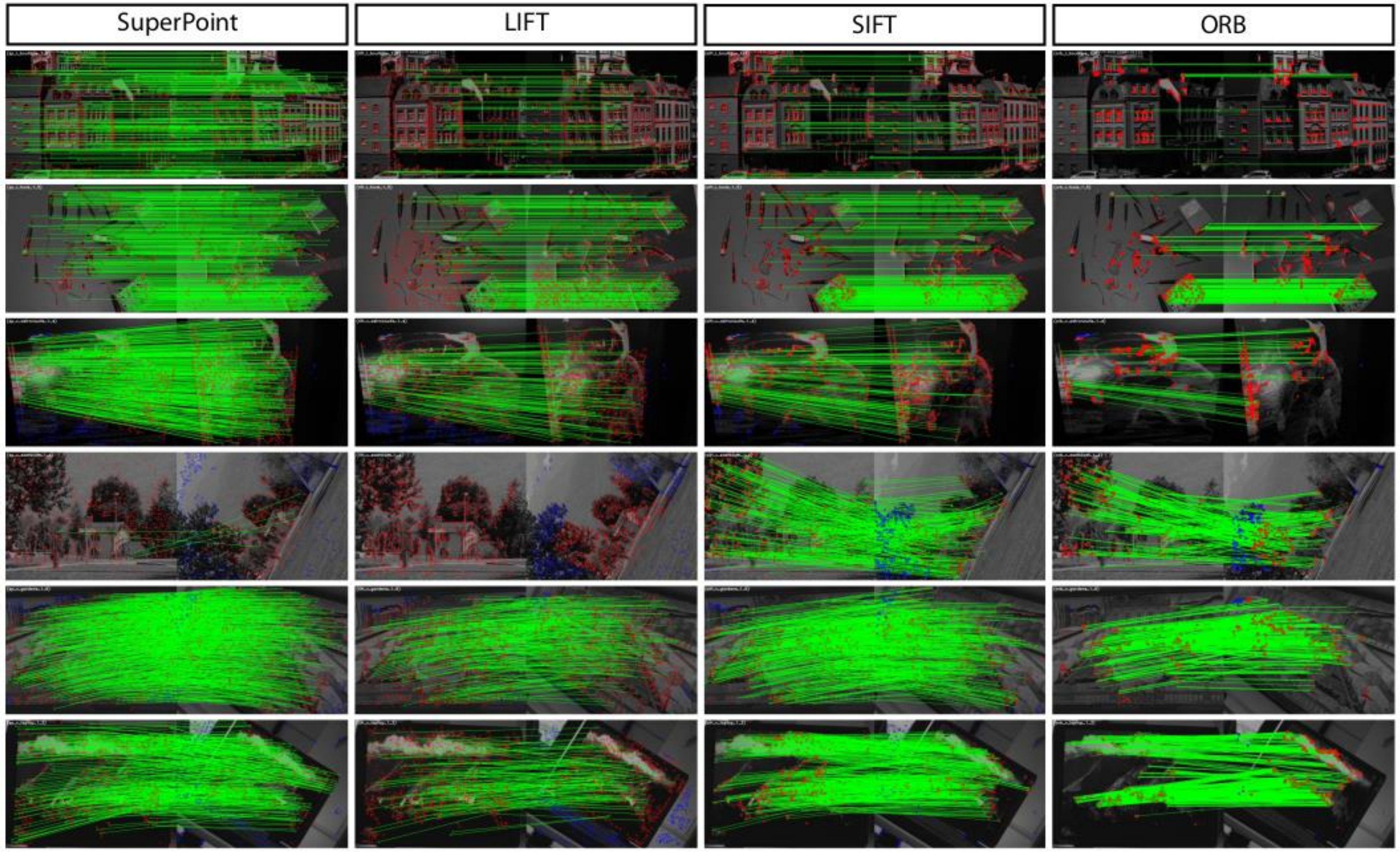


a) Training on synthetic dataset to bootstrap the detector

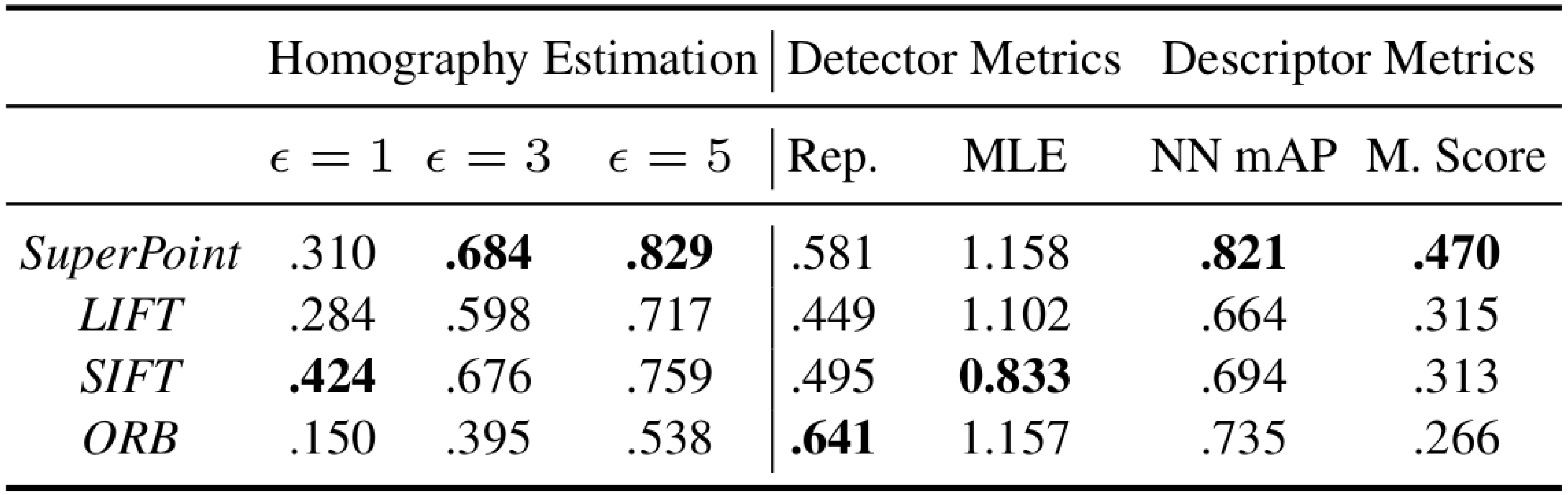
a) Use this detector on real images and generate groundtruth correspondences by sampling perspective transformations

a) Jointly training both detector and descriptor

SuperPoint - Qualitative (Cherry Picked) Results



SuperPoint - Quantitative Results



	Homography Estimation			Detector Metrics		Descriptor Metrics	
	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	Rep.	MLE	NN mAP	M. Score
SuperPoint	.310	**.684**	**.829**	.581	1.158	**.821**	**.470**
LIFT	.284	.598	.717	.449	1.102	.664	.315
SIFT	**.424**	.676	.759	.495	**0.833**	.694	.313
ORB	.150	.395	.538	**.641**	1.157	.735	.266

- **ε**: Pixel distance to count keypoint as correct. SIFT is very accurate due to sub-pixel refinement.
- **MLE** (Mean Localization Error): Lower is better. Metric for determining detector accuracy
- **NN mAP** (nearest-neighbor mean average precision): Measures discriminativeness of descriptors. Higher is better.
- **M. Score** (Matching Score): Evaluates detector and descriptor jointly on groundtruth correspondences.

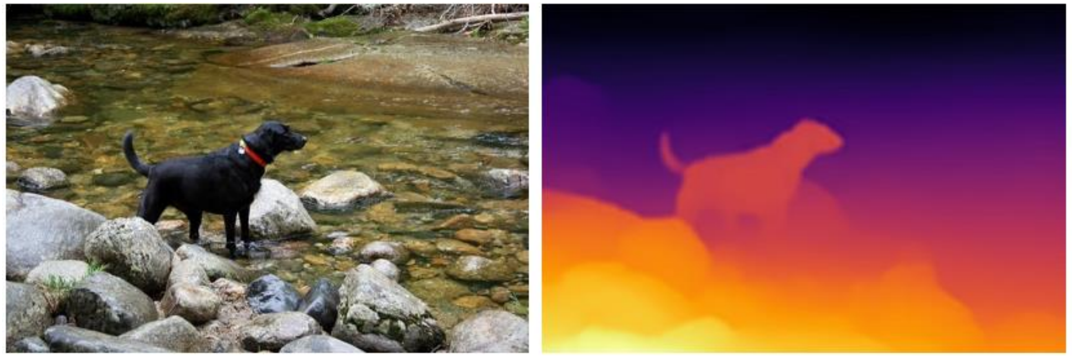
Applications to Computer Vision

Monocular Depth Estimation

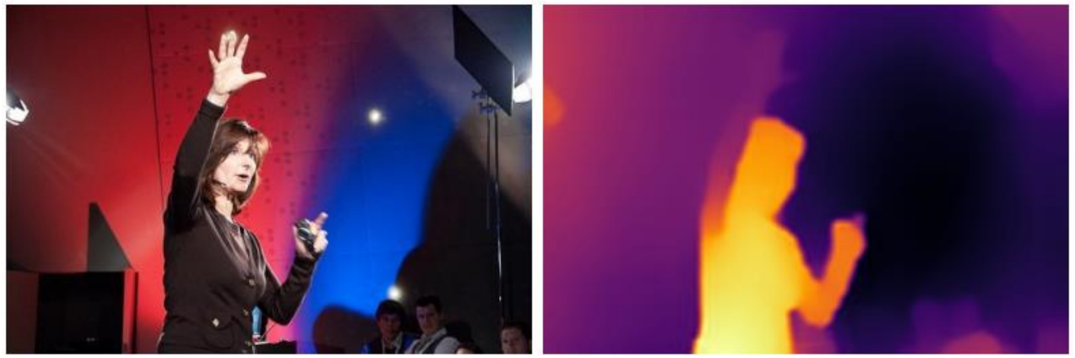
Supervised Learning of Monocular Depth (single image)

- Depth cannot be inferred from a single view by only considering geometric information
- Neural networks can learn priors (e.g. object sizes, scene regularity) to predict depth
- But learned priors do also lead to failure cases

good prediction



failure

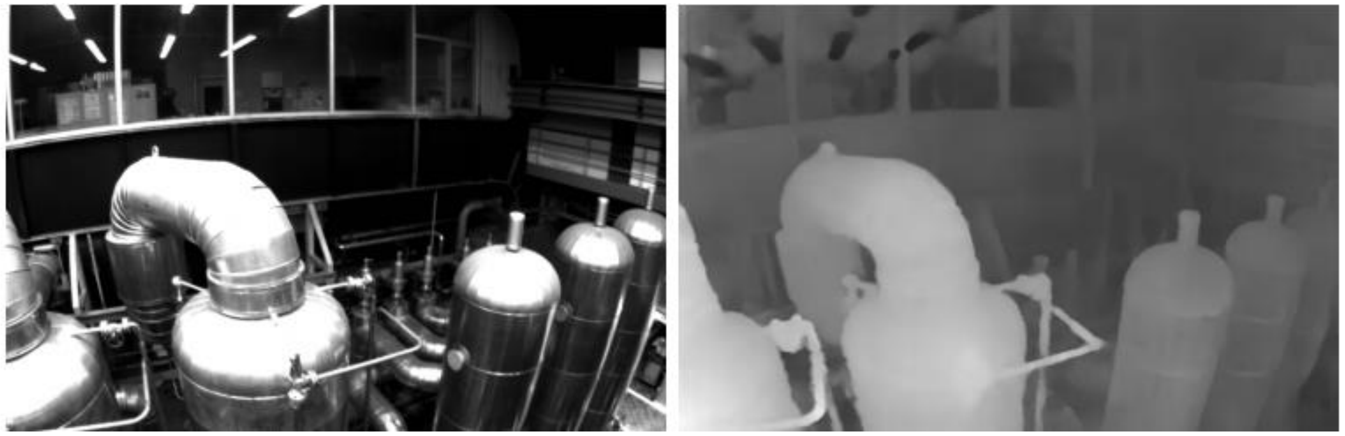


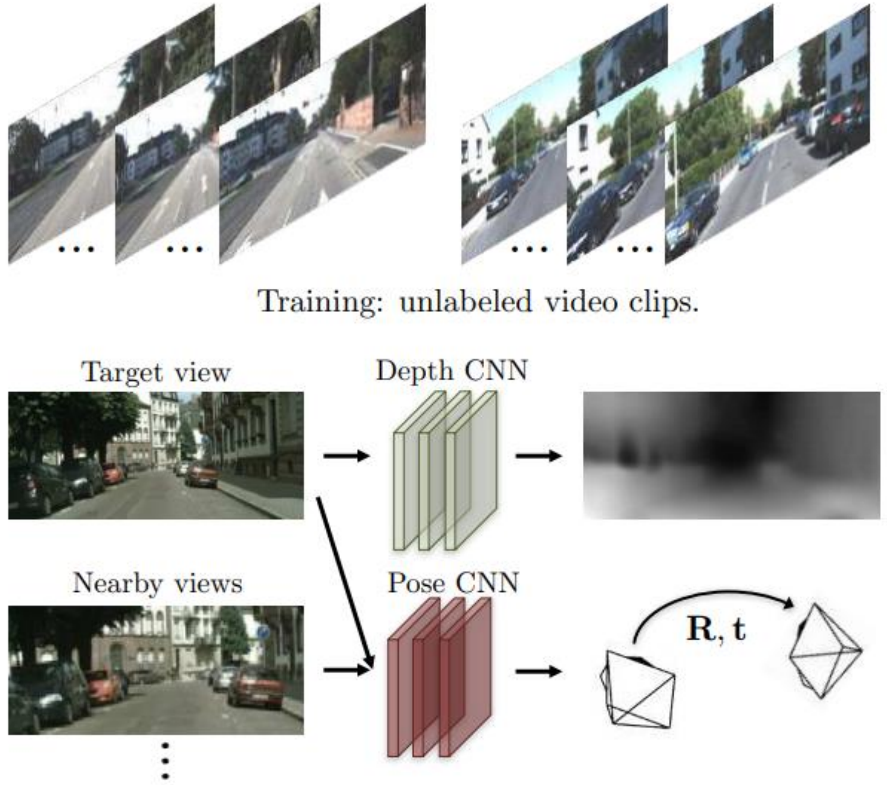
- **Main challenge is ground-truth data** -> Research on supervised monocular depth estimation has cooled down.

Depth map prediction from a single image using a multi-scale deep network – D. Eigen et al, NIPS 2014

Self-Supervised Learning of Monocular Depth (video)

- Video sequences can be used to **jointly learn depth and ego-motion without groundtruth**.
- This way, unlimited amounts of data can be used for training.
- Requires the definition of a **proxy loss**. Training with this proxy loss ideally leads to accurate prediction of depth and ego-motion.





Depth from Video in the Wild – A. Gordon et al, ICCV 2019
Unsupervised learning of depth and ego-motion from video, T. Zhou et al. CVPR 2017

Self-Supervised Learning of Monocular Depth (video) - Training

- The depth network predicts depth D of a target image t.
- The pose network takes multiple images other than t and predicts the transformations between those image wrt. the target image.
- The differentiable mapping between the homogeneous pixel coordinates p_t in image t and p_s in image s:

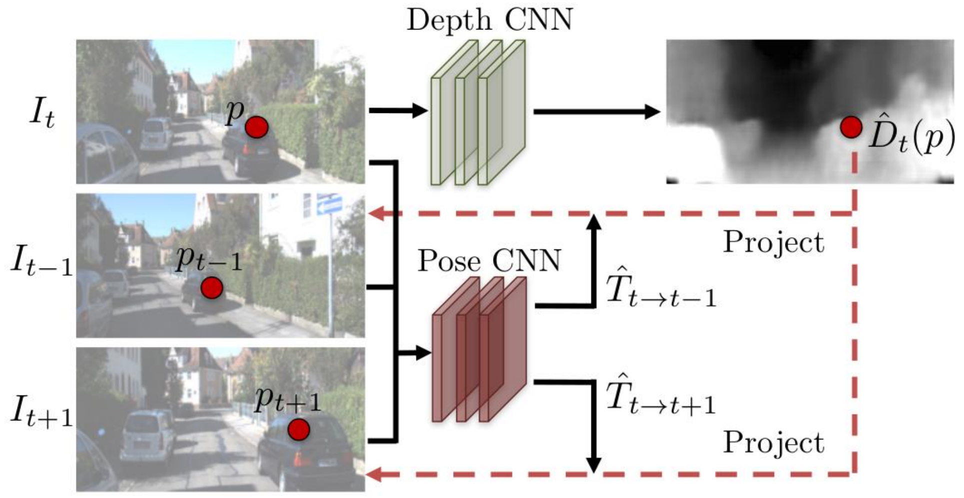
$$p_s \sim K\hat{T}_{t\rightarrow s}\hat{D}_t(p_t)K^{-1}p_t$$

- Minimal loss function:

$$\mathcal{L}_{final} = \sum_l \mathcal{L}_{vs}^l + \lambda_s \mathcal{L}_{smooth}^l$$

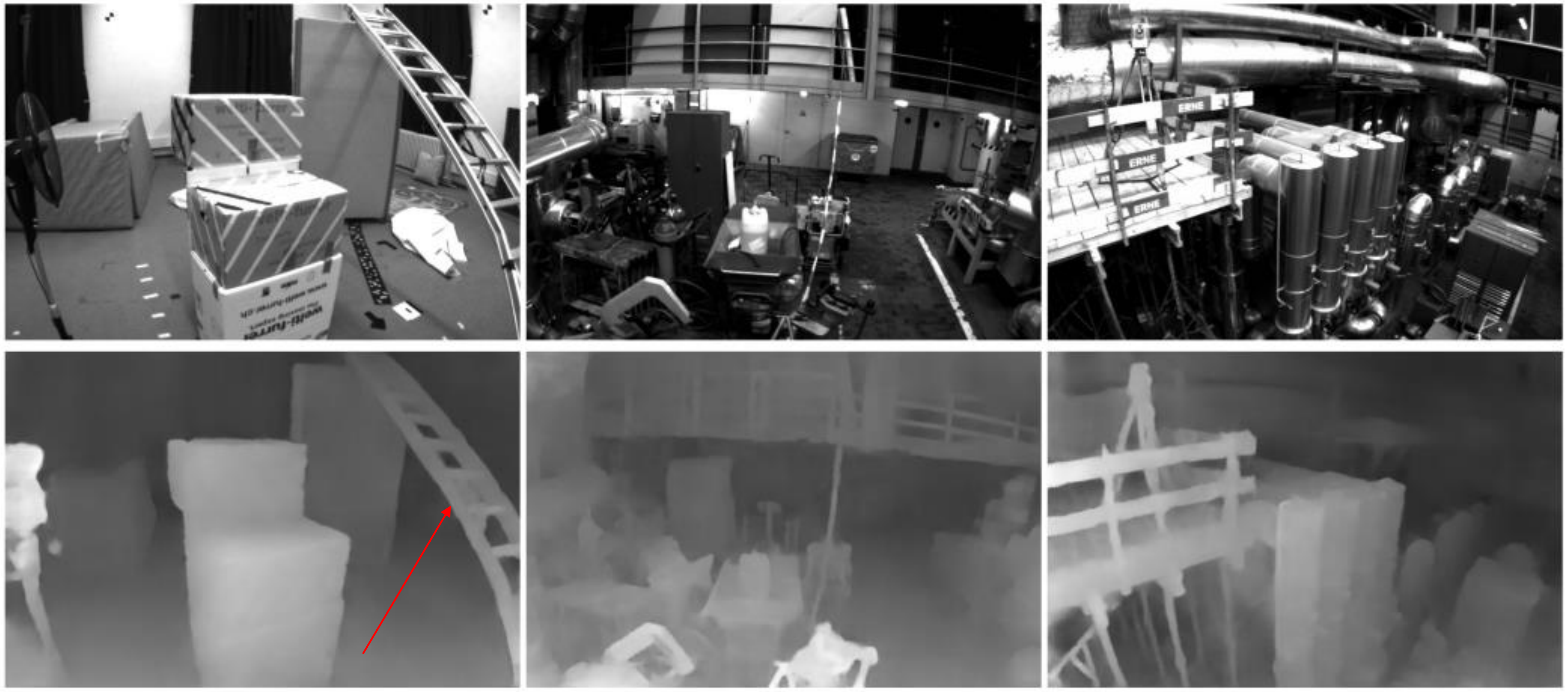
$$\mathcal{L}_{vs} = \sum_s \sum_p |I_t(p) - \hat{I}_s(p)|$$

  - Smoothness: Penalize depth map discontinuities if image intensity is discontinuous *
  - Levels l: The losses are applied on multiple image levels for gradient flow.



*Unsupervised monocular depth estimation with left-right consistency – A. Gordon et al, CVPR 2017
Unsupervised learning of depth and ego-motion from video, T. Zhou et al. CVPR 2017

Self-Supervised Learning of Monocular Depth (video) - Results

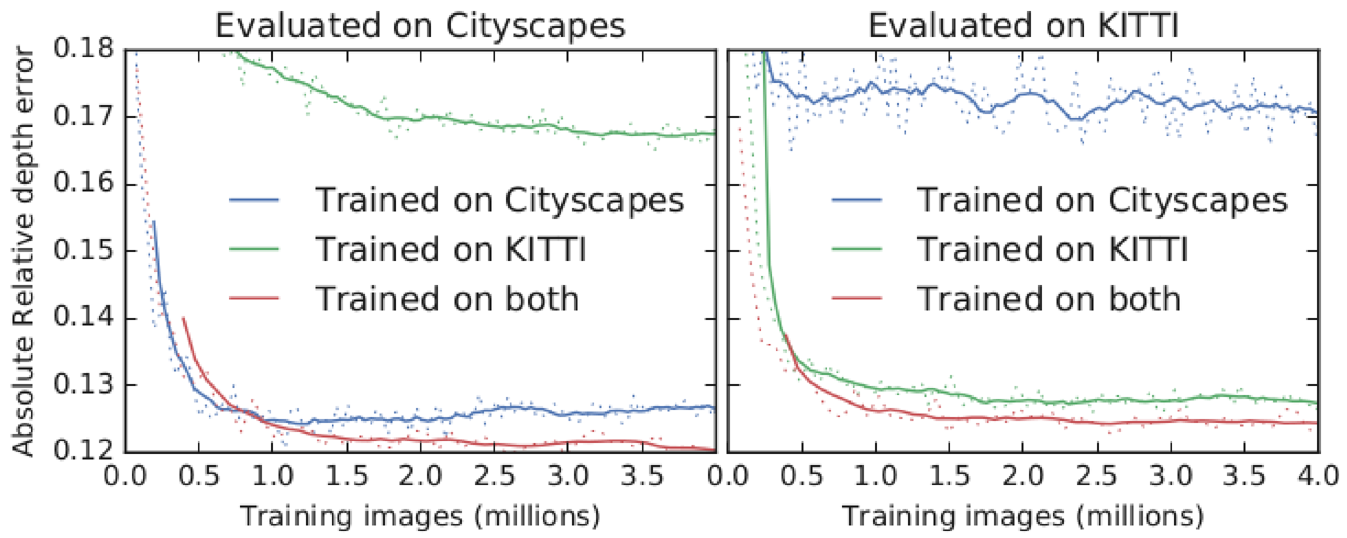


Depth from Video in the Wild – A. Gordon et al, ICCV 2019

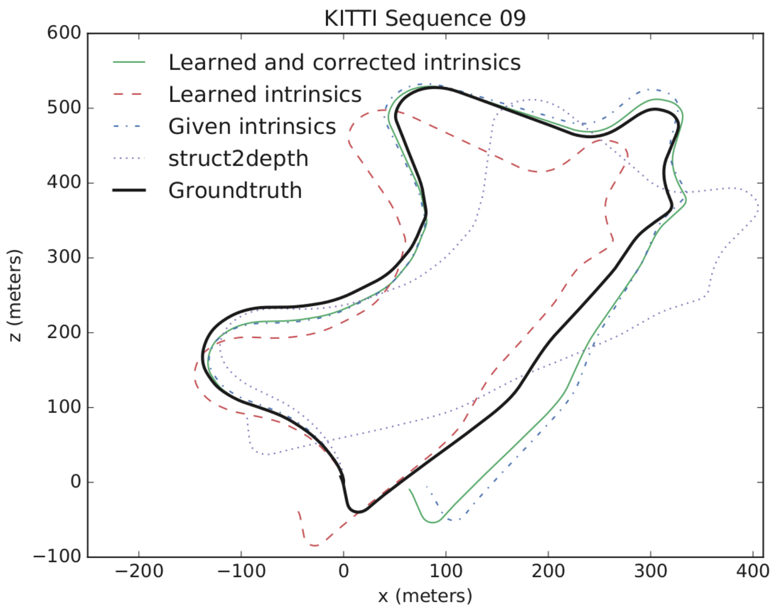
Self-Supervised Learning of Monocular Depth (video) - Results

- Ego-motion is usually seen as by-product of the learning process. Modern feature-based V(I)O algorithms are more accurate than ego-motion from learning approaches.
- Depth prediction has reached remarkable accuracy. However, depth prediction error still depends on matching training and testing data distribution.

Cross-dataset generalization is still a challenge:



Learning intrinsics can also lead to better results:

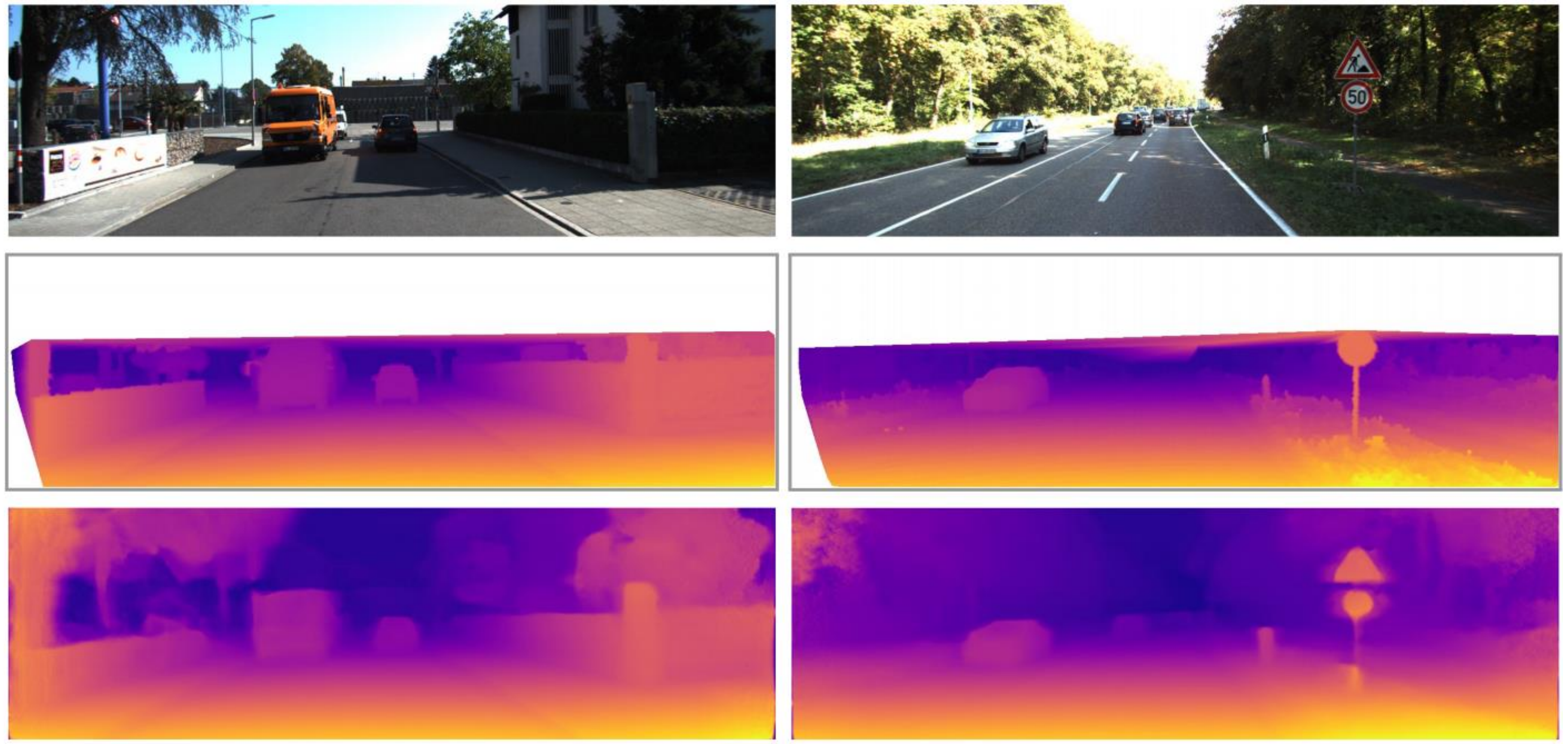


Depth from Video in the Wild – A. Gordon et al, ICCV 2019

Applications to Computer Vision

Monocular Disparity Estimation

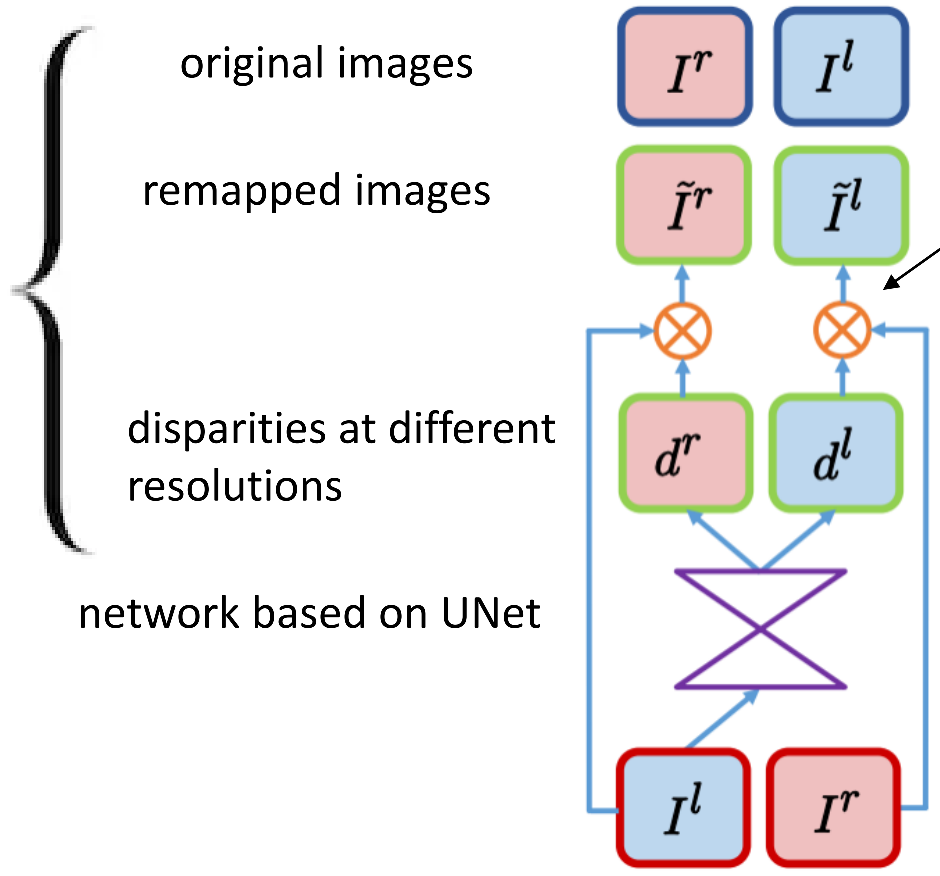
- The task is to predict the per-pixel disparity from a monocular image. The disparity is with respect to a **virtual camera** with a given baseline. With this disparity and baseline full depth can be recovered.
- As we saw, monocular depth works because the network learns the absolute size of objects and rescales the depth accordingly.



Unsupervised Monocular Disparity

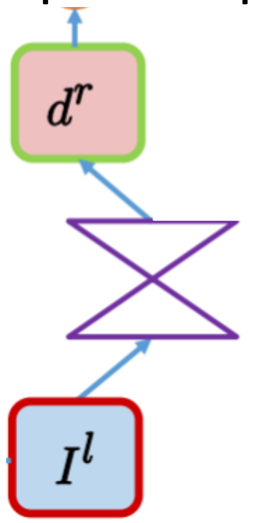
- What is this virtual camera view? The network is **trained using stereo images.** Once trained the network **can be used to predict depth from single images**.

used to define the training loss



The images are warped using the disparity map

can be used to compute depth



Training

Testing

Unsupervised Monocular Disparity - Training

- The network is trained in an unsupervised way. This means no ground truth depth maps are required.
- For each view (left and right) and for each resolution for disparities (four in total) three losses are computed:

1. **Appearance matching loss**

$$C_{ap}^{l} = \frac{1}{N} \sum_{i,j} \alpha \frac{1 - \mathrm{SSIM}(I_{ij}^{l}, \tilde{I}_{ij}^{l})}{2} + (1-\alpha) \left\| I_{ij}^{l} - \tilde{I}_{ij}^{l} \right\|$$

SSIM: structural similarity, can be at most 1

2. **Disparity Smoothness loss**

$$C_{ds}^{l} = \frac{1}{N} \sum_{i,j} \left| \partial_x d_{ij}^{l} \right| e^{-\left\| \partial_x I_{ij}^{l} \right\|} + \left| \partial_y d_{ij}^{l} \right| e^{-\left\| \partial_y I_{ij}^{l} \right\|}$$

∂: derivative, it is weighted by the image derivative

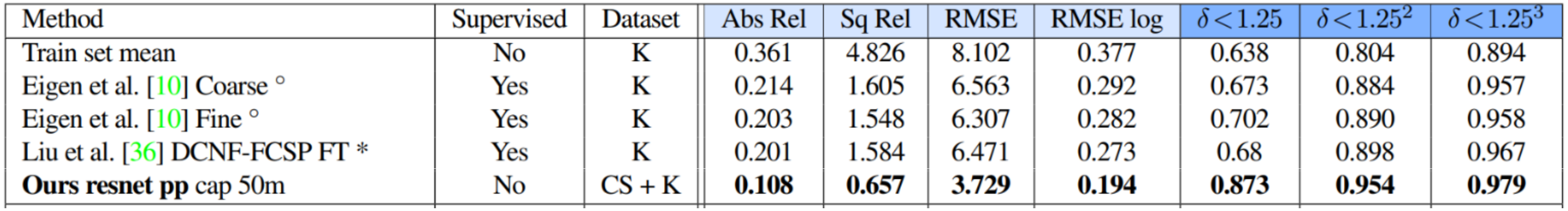
3. **Left-Right Disparity Consistency Loss**

$$C_{lr}^{l} = \frac{1}{N} \sum_{i,j} \left| d_{ij}^{l} - d_{ij+d_{ij}^{l}}^{r} \right|$$

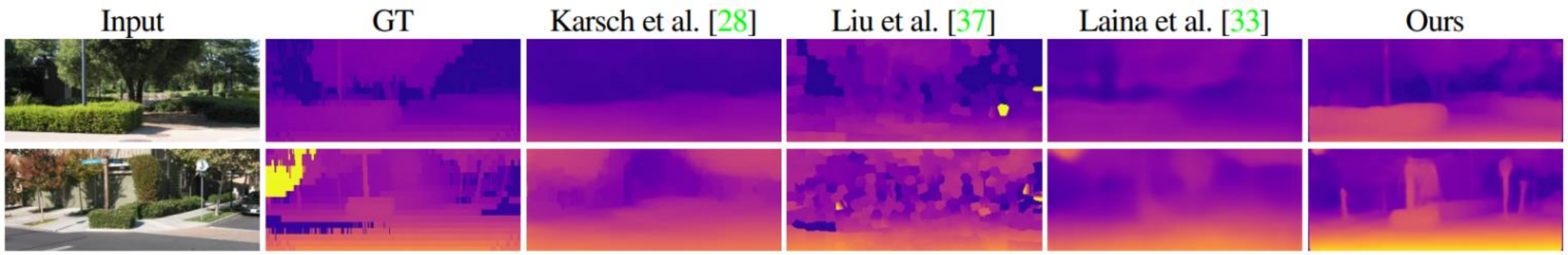
Remapped disparity maps must be the same

Unsupervised Monocular Disparity - Results

- The network is trained on the KITTI dataset using 29,000 stereo images and evaluate on depth maps provided in the dataset. Surprisingly, it outperforms supervised methods.
- Again this method works well in regions that have **low texture** and **repetitive structure**



Method	Supervised	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Train set mean	No	K	0.361	4.826	8.102	0.377	0.638	0.804	0.894
Eigen et al. [10] Coarse °	Yes	K	0.214	1.605	6.563	0.292	0.673	0.884	0.957
Eigen et al. [10] Fine °	Yes	K	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Liu et al. [36] DCNF-FCSP FT *	Yes	K	0.201	1.584	6.471	0.273	0.68	0.898	0.967
Ours resnet pp cap 50m	No	CS + K	**0.108**	**0.657**	**3.729**	**0.194**	**0.873**	**0.954**	**0.979**



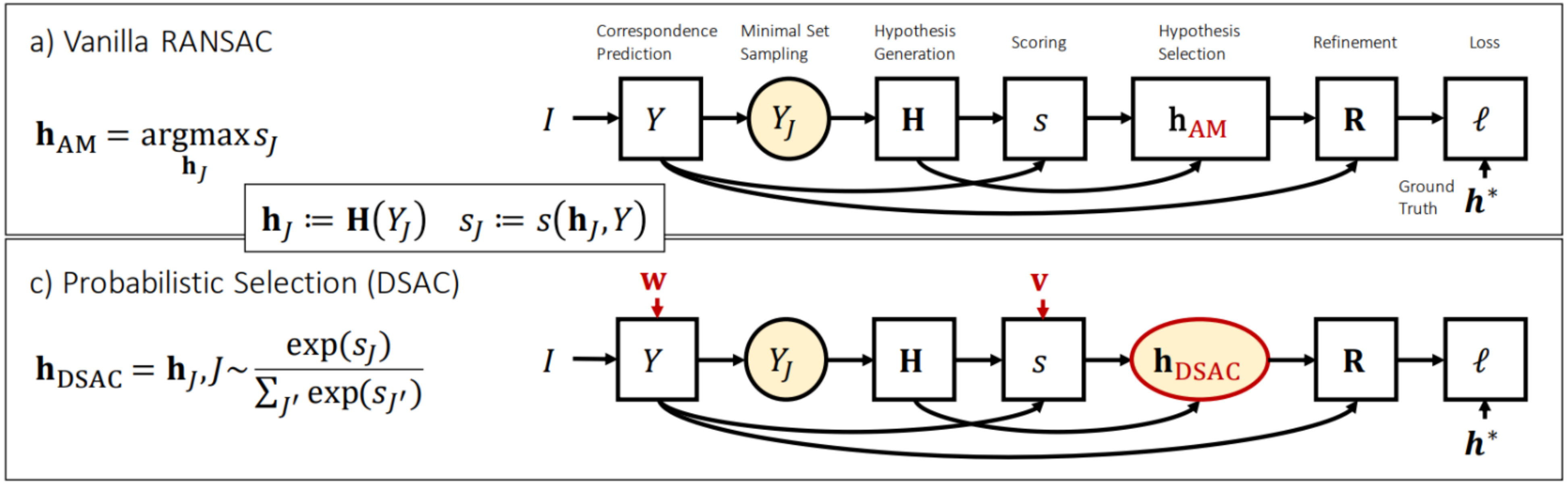
Applications to Computer Vision

Differential Sample Consensus (DSAC)

- Random Sample Consensus (RANSAC) is not differentiable since it relies on selecting a hypothesis based on maximizing the number of inliers.
- DSAC shows how sample consensus can be used in a differentiable way
- This enables the use of sample consensus in a variety of learning tasks.
- Here we cover its application to localization: given a single 2D image and a **global** 3D map of a scene, compute the pose of the camera

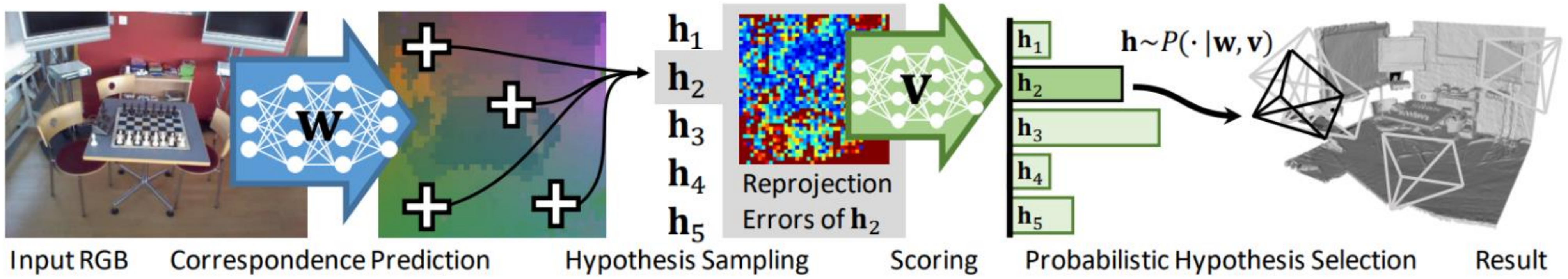
Choose the best based on score

Randomly sample based on score



a) Vanilla RANSAC
$\mathbf{h}_{\mathrm{AM}} = \operatorname{argmax}_{\mathbf{h}_J} s_J$
$\mathbf{h}_J := \mathbf{H}(Y_J) \quad s_J := s(\mathbf{h}_J, Y)$
Correspondence Prediction
Minimal Set Sampling
Hypothesis Generation
Scoring
Hypothesis Selection
Refinement
Loss
I
Y
Y_J
$\mathbf{H}$
s
$\mathbf{h}_{\mathrm{AM}}$
$\mathbf{R}$
ℓ
Ground Truth
$\boldsymbol{h}^*$
c) Probabilistic Selection (DSAC)
$\mathbf{h}_{\mathrm{DSAC}} = \mathbf{h}_J, J \sim \frac{\exp(s_J)}{\sum_{J'} \exp(s_{J'})}$
$\mathbf{w}$
$\mathbf{v}$
$\mathbf{h}_{\mathrm{DSAC}}$

Differential Sample Consensus (DSAC) - Method


W
h_1
h_2
h_3
h_4
h_5
Reprojection Errors of h_2
V
$h \sim P(\cdot \mid w, v)$
Input RGB
Correspondence Prediction
Hypothesis Sampling
Scoring
Probabilistic Hypothesis Selection
Result

1. For each patch (42 x 42) in the image predict its corresponding global 3D point using a CNN (VGG) with parameters **W**. This gives you dense 2D-3D correspondence map that can now be sampled by DSAC.
2. Sample 2D-3D correspondences randomly and compute the hypothesis (=camera pose) with P3P. Why do we need four points?
3. Reproject the all predicted 3D points according to each hypothesis and compute their reprojection error. This is done for every 42 x 42 patch resulting in a 40 x 40 error map.
4. Using this error map predict a score for each hypothesis using a DNN (VGG) with parameters **V.**
5. Randomly sample the best hypothesis based on the softmax of the scores. The likelihood of sampling a hypothesis is higher if it has a higher score. This is only done during training. During testing we choose the best hypothesis.
6. With the best hypothesis find all the inliers and compute the pose by minimizing the reprojection error.

Differential Sample Consensus (DSAC) - Training

- DSAC is trained in a supervised fashion in a scene with known 3D environment
- We try to minimize the expected deviation from the ground truth pose after refinement.

$$\underset{\substack{\uparrow \\ \text{Network} \\ \text{parameters}}}{\tilde{\mathbf{w}}, \tilde{\mathbf{v}}} = \arg\min_{\mathbf{w},\mathbf{v}} \sum_{\substack{I\in\mathcal{I} \\ \uparrow \\ \text{Images}}} \mathbb{E}_{J\sim P(J|\mathbf{v},\mathbf{w})}[\underset{\substack{\uparrow \\ \text{L2 norm}}}{l}(\underset{\substack{\uparrow \\ \text{refinement} \\ \text{step}}}{\mathbf{R}}(\mathbf{h}^w_J, \mathbf{Y}^w))))] = \sum_{I\in\mathcal{I}} \sum_{J} \underset{\substack{\uparrow \\ \text{samples}}}{P(J|\mathbf{v},\mathbf{w})}\, l(\mathbf{R}(\underset{\substack{\uparrow \\ \text{hypothesis from} \\ \text{samples J}}}{\mathbf{h}^w_J}, \underset{\substack{\uparrow \\ \text{3D point for} \\ \text{each patch}}}{\mathbf{Y}^w}))))$$

- The probabilities are computed from the softmax of the scores

$$P(J|\mathbf{v},\mathbf{w}) = \frac{\exp(s(\mathbf{h}^w_J, Y^w; \mathbf{v}))}{\sum_{J'} \exp(s(\mathbf{h}^w_{J'}, Y^w; \mathbf{v}))}$$

- During training, random images are selected and 256 hypotheses are generated which allow you to estimate this loss and minimize it.

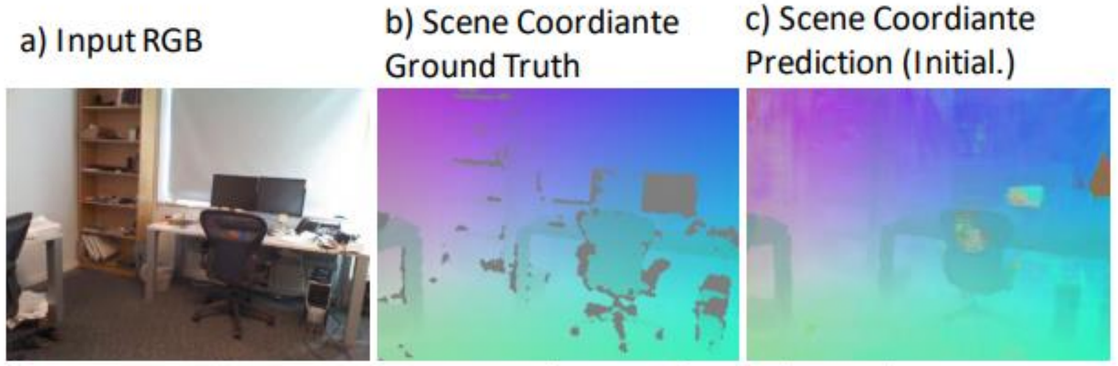
Differential Sample Consensus (DSAC) - Results

- The method was trained and evaluated on the 7-Scenes Dataset [1] which provides 3D models, RGB-D data, and ground truth for several camera paths through the scenes.

- Compared to model-based methods, DSAC based localization does not rely on features. This makes the method more robust in scenes with **low texture** and **repetitive patterns.**

	Sparse Features [36]	Ours: Trained End-To-End SoftAM	Ours: Trained End-To-End DSAC
Chess	70.7%	94.2%	94.6%
Fire	49.9%	**76.9%**	74.3%
Heads	67.6%	74.0%	71.7%
Office	36.6%	56.6%	71.2%
Pumpkin	21.3%	51.9%	53.6%
Kitchen	29.8%	46.2%	**51.2%**
Stairs	9.2%	5.5%	4.5%
Average	40.7%	57.9%	**60.1%**
Complete	38.6%	57.8%	**62.5%**

*fraction of poses with error smaller than 5° and 5 cm


a) Input RGB
b) Scene Coordiante Ground Truth
c) Scene Coordiante Prediction (Initial.)

[1]

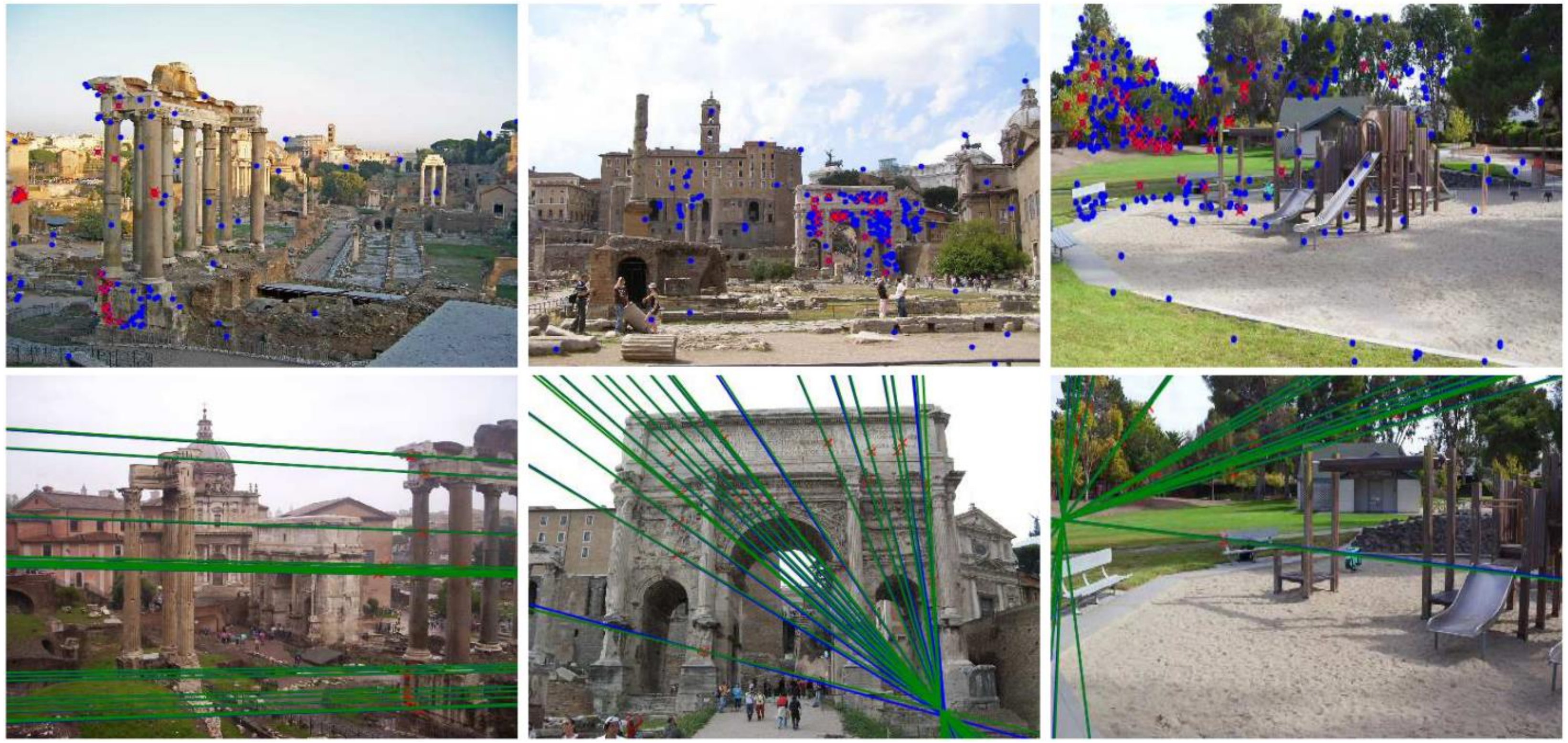
DSAC - Differentiable RANSAC for Camera Localization, E. Brachmann et al. CVPR 2017

[1] Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. J. Shotton et al. CVPR 2013

Applications to Computer Vision

Deep Fundamental Matrix Estimation

- Eight-Point Algorithm involves solving a least-squares (LS) problem together with RANSAC
- Can we estimate the fundamental matrix estimation problem with fewer iterations?
- Yes: Exploit structure in the data and formulate a **weighted** least-squares problem accordingly.



Top-bottom as image-pair

Red: inlier (correspondences)
Blue: outliers

Epipolar lines

Green: estimated
Blue: ground-truth

Deep Fundamental Matrix Estimation - General Method

- Reformulate generic homogeneous LS to iterative weighted homogeneous LS problem

Generic hom. LS:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \sum_{i=1}^{N} \| (\overset{\text{Mapping of points}}{\mathbf{A}}(\underbrace{\mathbf{P}}_{\text{Collection of Points}}))_i \cdot \underbrace{\mathbf{x}}_{\text{Model parameters}} \|^2 \\ \text{subject to} \quad & \|\mathbf{x}\| = 1, \end{aligned}$$

Iterative weighted hom. LS:

$$\mathbf{x}^{j+1} = \underset{\mathbf{x}:\ \|\mathbf{x}\|=1}{\arg\min} \sum_{i=1}^{N} \overset{\text{Weight}}{w}(\overset{\text{Point i}}{\mathbf{p}_i}, \mathbf{x}^j) \, \| (\mathbf{A}(\mathbf{P}))_i \cdot \mathbf{x} \|^2$$

- **Iterative**: instead of solving the LS problem several times in a RANSAC loop. Weights change with each iteration because the estimate x changes too.
- **Weighted**: Learn weights from data. Parameterized by a neural network

$$\mathbf{x}^{j+1} = \underset{\mathbf{x}:\ \|\mathbf{x}\|=1}{\arg\min} \sum_{i=1}^{N} (w(\mathbf{P}, \underbrace{\mathbf{S}}_{\text{Optional side-information for each point.}}, \mathbf{x}^j; \overset{\text{Network parameters}}{\boldsymbol{\theta}}))_i \, \| (\mathbf{A}(\mathbf{P}))_i \cdot \mathbf{x} \|^2$$

Deep Fundamental Matrix Estimation - Training

- Symmetric epipolar distance as residual function:

$$r(\mathbf{p}_i, \mathbf{F}) = \left|\hat{\mathbf{p}}_i^\top \mathbf{F} \hat{\mathbf{p}}_i'\right| \left(\frac{1}{\|\mathbf{F}^\top \hat{\mathbf{p}}_i\|_2} + \frac{1}{\left\|\mathbf{F}\hat{\mathbf{p}}_i'\right\|_2} \right)$$

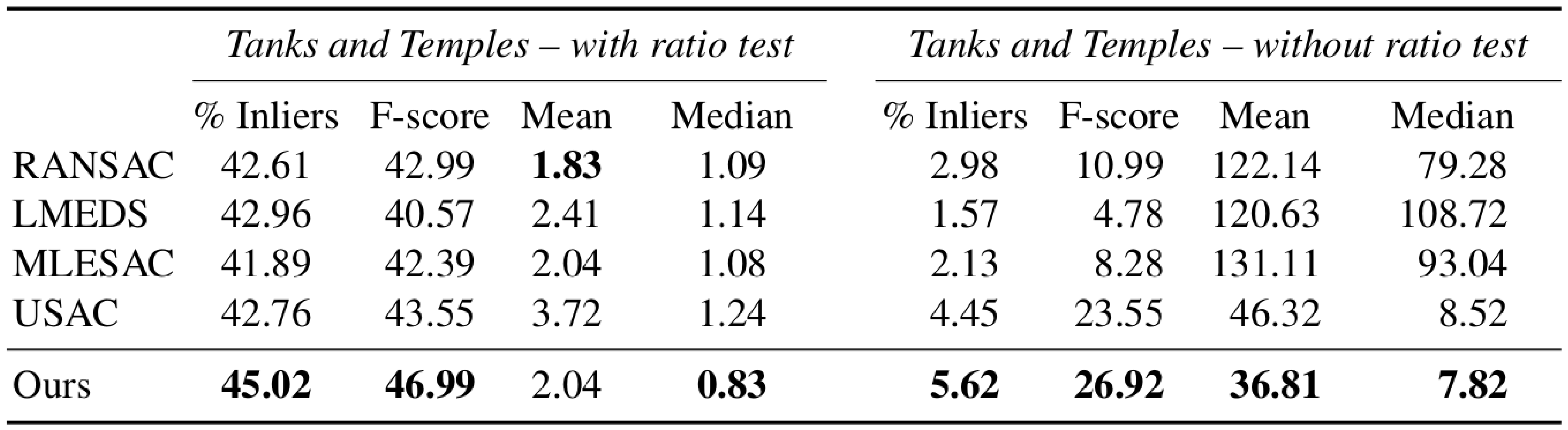
p: homogeneous coordinate of left image
p': homogeneous coordinate of right image

- Training loss:

$$\mathcal{L} = \frac{1}{N_{gt}} \sum_{j=0}^{D} \sum_{i=1}^{N_{gt}} \min\left(r(\mathbf{p}_i^{gt}, g(\mathbf{x}^j)), \gamma \right)$$

g(x):	Solves weighted LS problem (8-pt LS problem) and extracts the estimated fundamental matrix F.
γ:	Stabilize training by clamping the residual
D:	Number of iterations (usually 5)
N:	Number of correspondences

Deep Fundamental Matrix Estimation - Results



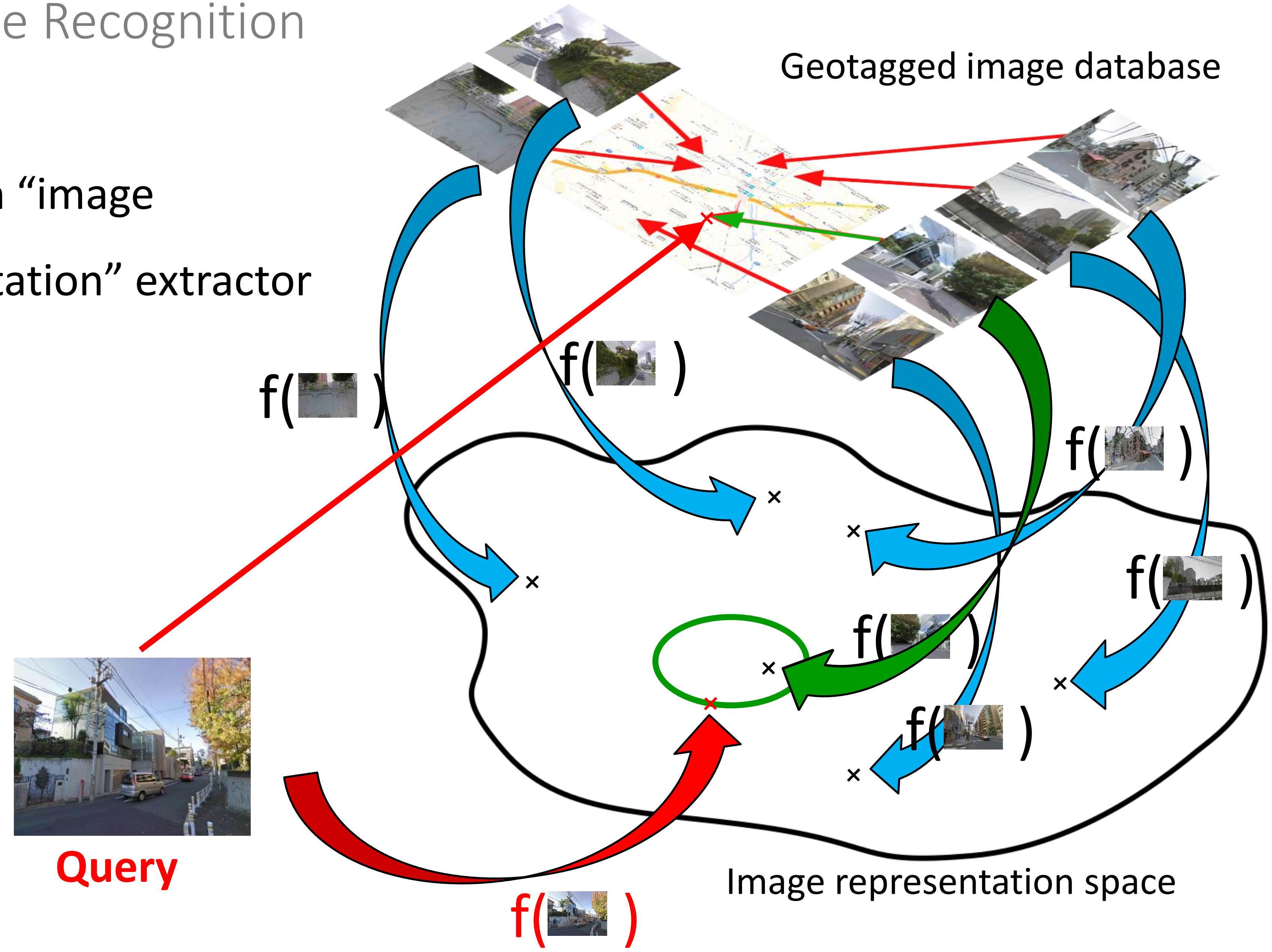
	Tanks and Temples – with ratio test				*Tanks and Temples – without ratio test*			
	% Inliers	F-score	Mean	Median	% Inliers	F-score	Mean	Median
RANSAC	42.61	42.99	**1.83**	1.09	2.98	10.99	122.14	79.28
LMEDS	42.96	40.57	2.41	1.14	1.57	4.78	120.63	108.72
MLESAC	41.89	42.39	2.04	1.08	2.13	8.28	131.11	93.04
USAC	42.76	43.55	3.72	1.24	4.45	23.55	46.32	8.52
Ours	**45.02**	**46.99**	2.04	**0.83**	**5.62**	**26.92**	**36.81**	**7.82**

- Ratio test refers to David Lowe’s ratio test introduces with SIFT
- Better results in noisy case (no ratio test)

Applications to Computer Vision

Place Recognition

Design an "image representation" extractor $f(I, \theta)$



NetVlad

Mimic the classical pipeline with deep learning

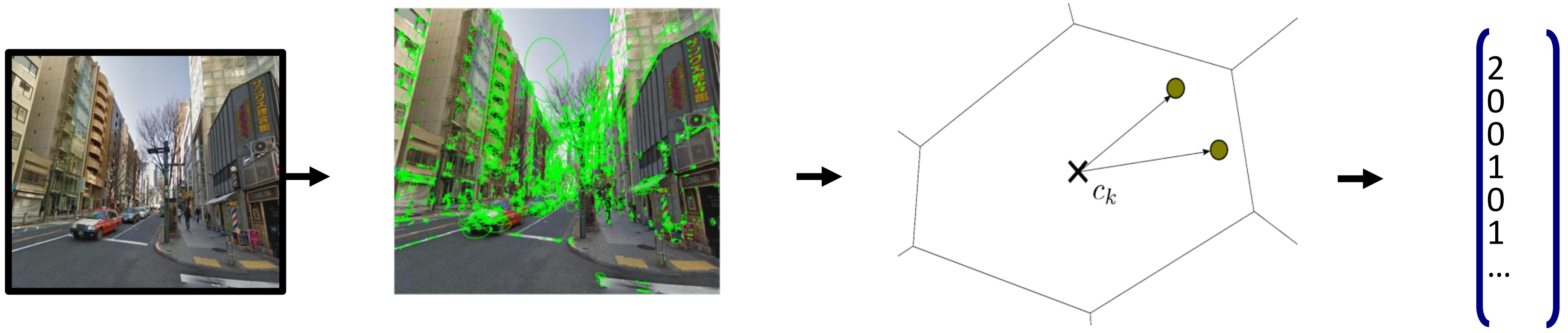
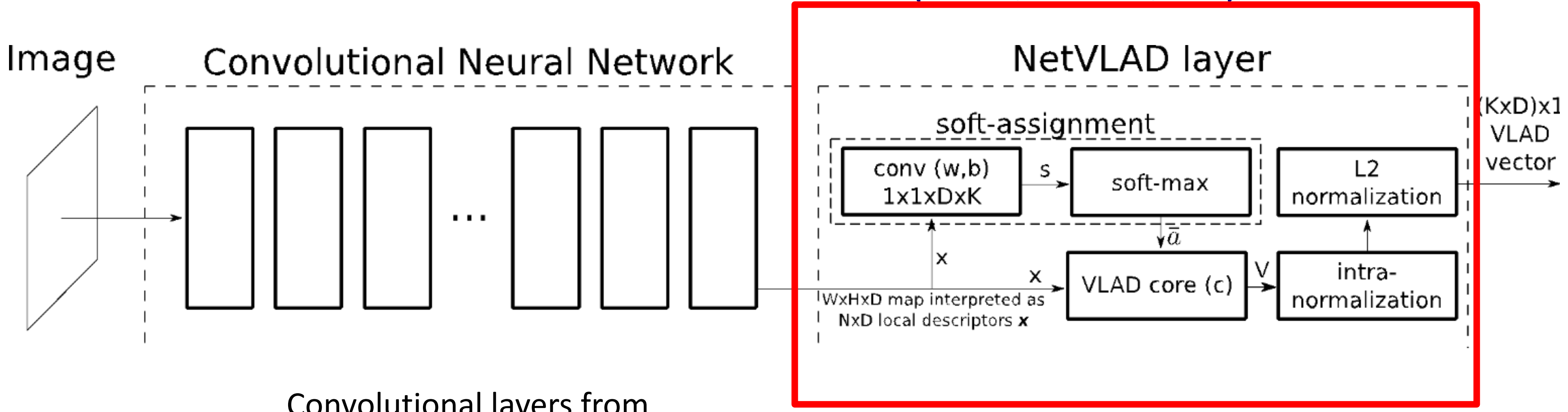


Image I

Extract local features (SIFT)

Aggregate (BoW, **VLAD**, FV)

F(I)



Convolutional layers from AlexNet or VGGNet

Trainable pooling layer

Slide adapted from NetVLAD presentation, CVPR 2017

NetVlad Loss

- Triplet loss formulation

Matching samples

$D_p = ||F_\theta($ $) - F_\theta($ $)||^2$

$D_n = ||F_\theta($ $) - F_\theta($ $)||^2$

Non matching samples

$$L_\theta = \sum_{samples} \max(D_{p(\theta)} + \overbrace{m}^{\text{margin}} - D_{n(\theta)}, 0)$$

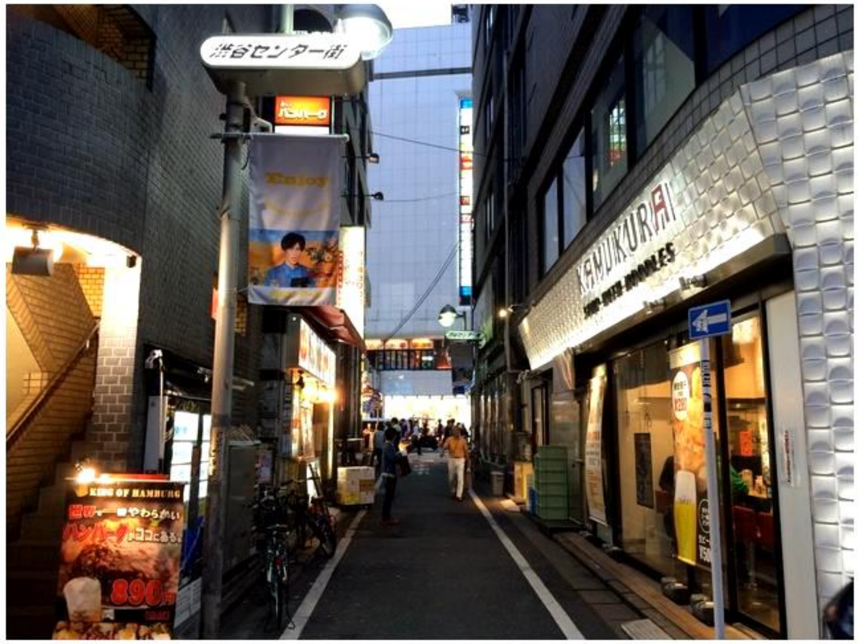
Disclaimer: The actual NetVlad loss is a slightly more complicated version of the one above

NetVlad Results

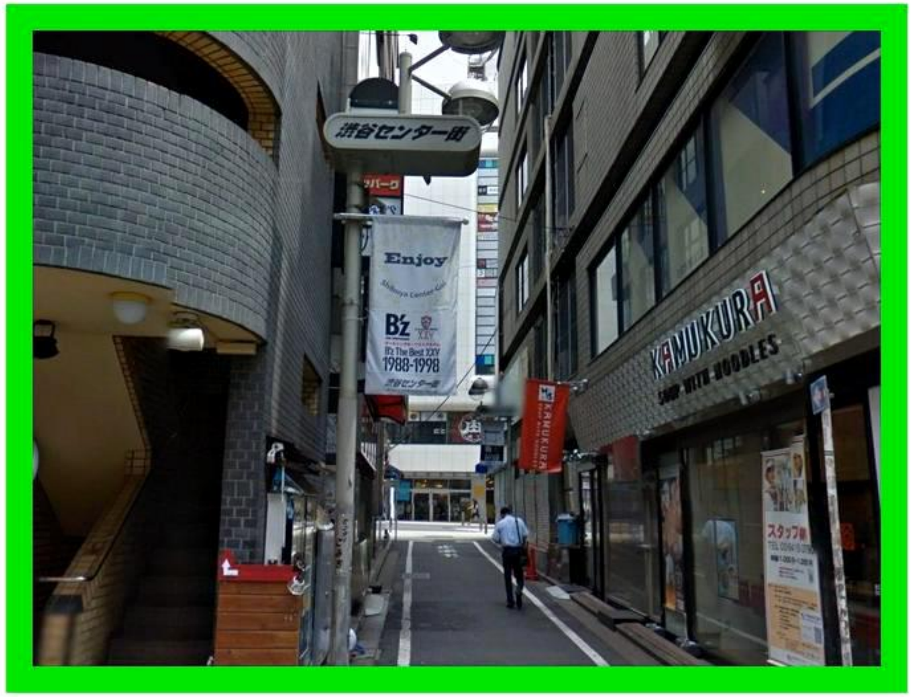
- Code, dataset and trained network online: give it a try!

  http://www.di.ens.fr/willow/research/netvlad/

Query



Top result

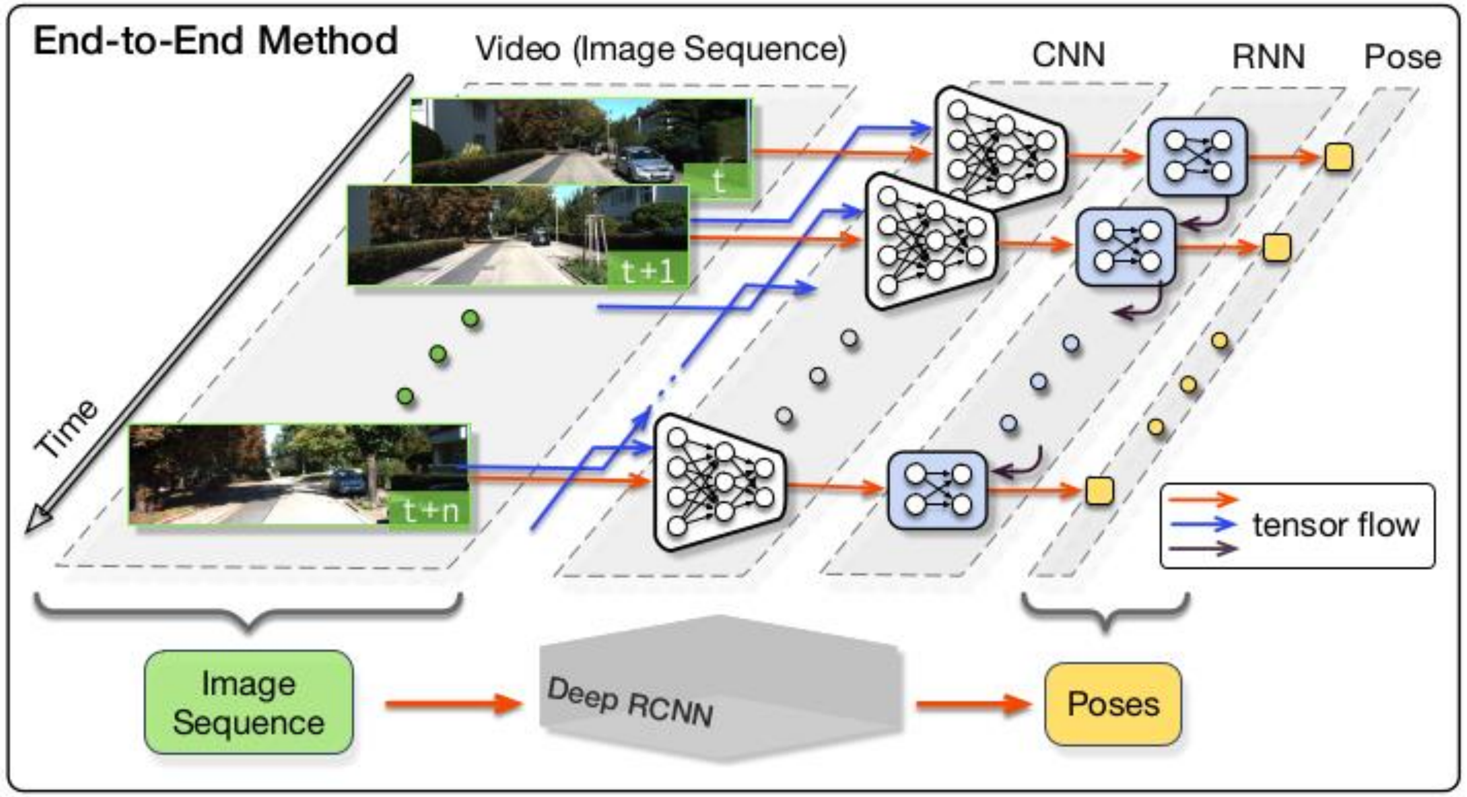


Slide adapted from NetVLAD presentation, CVPR 2017

Green: Correct
Red: Incorrect

Deep Visual Odometry (DeepVO)

- End-to-end trained method that computes the camera pose from images directly
- Encodes image pairs to geometric features using a **convolutional neural network** (CNN)
- Feeds these features to a **recurrent neural network** (RNN) which outputs the 6-DOF pose



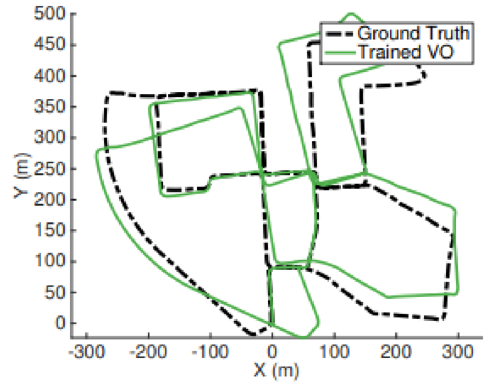
DeepVO End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks, S. Wang et al. ICRA 2017

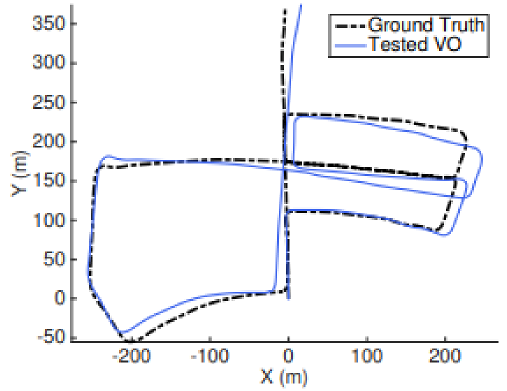
Deep Visual Odometry (DeepVO) - Training

- The network is trained on the KITTI VO/SLAM [1] benchmark with 7410 samples. To deal with the small amount of data they use a network pretrained on FlowNet [2]
- During training DeepVO tries to minimize the difference between ground-truth and predicted position and orientation (in euler angles):

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{t} \|\widehat{\mathbf{p}}_k - \mathbf{p}_k\|_2^2 + \kappa \|\widehat{\boldsymbol{\varphi}}_k - \boldsymbol{\varphi}_k\|_2^2$$

- The network is later tested on sequences of KITTI which were not seen during training

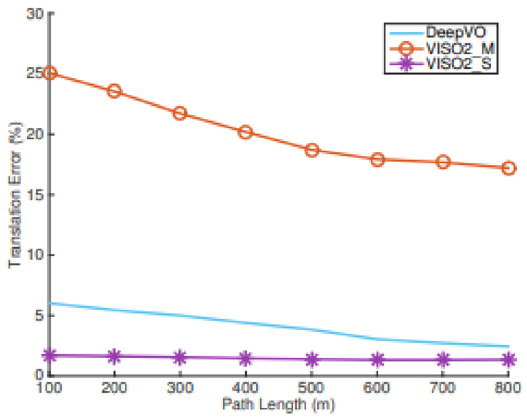




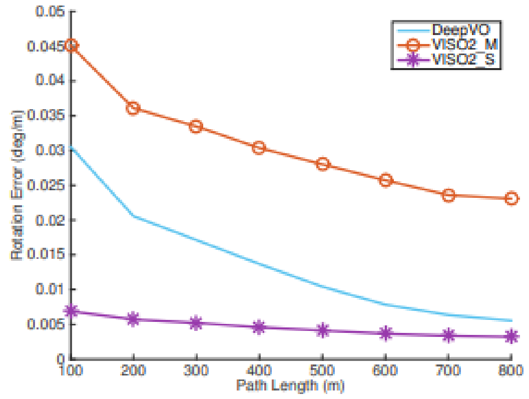
[1] Are we ready for autonomous driving? the KITTI vision benchmark suite, A. Geiger et al. CVPR 2012.
[2] FlowNet: Learning Optical Flow with Convolutional Networks, A. Dosovitskiy et al. ICCV 2017.

Deep Visual Odometry (DeepVO) - Results

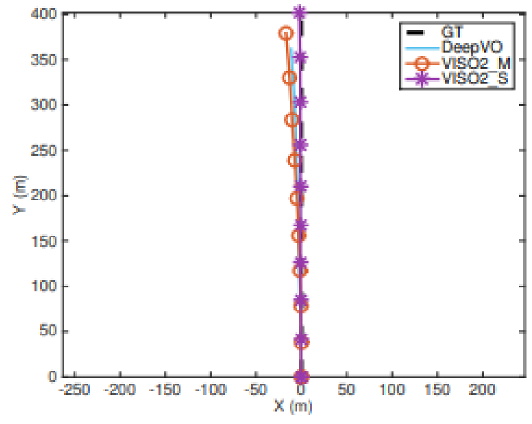
- DeepVO is compared to VISO2 [1] (Monocular and Stereo setup) in terms of rotation and translation error for different subtrajectory lengths and speeds.
- While DeepVO outperforms the monocular setup for VISO2 (VISO2_M) it is still outperformed by the stereo setup (VISO2_S)
- An important thing to note is that learning based visual odometry is not yet solved since many of these method overfit to the driving scenario (2D motion). However, this field is growing rapidly.



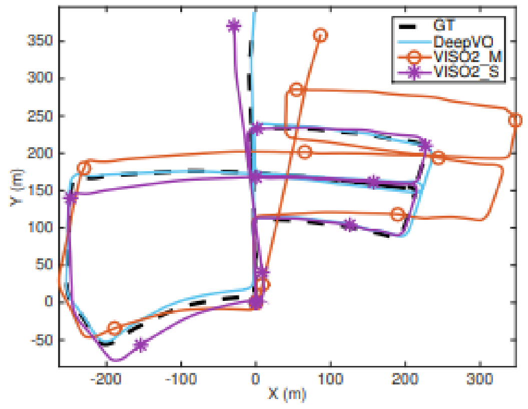
(a) Translation against path length.



(b) Rotation against path length.



(a) Sequence 04.

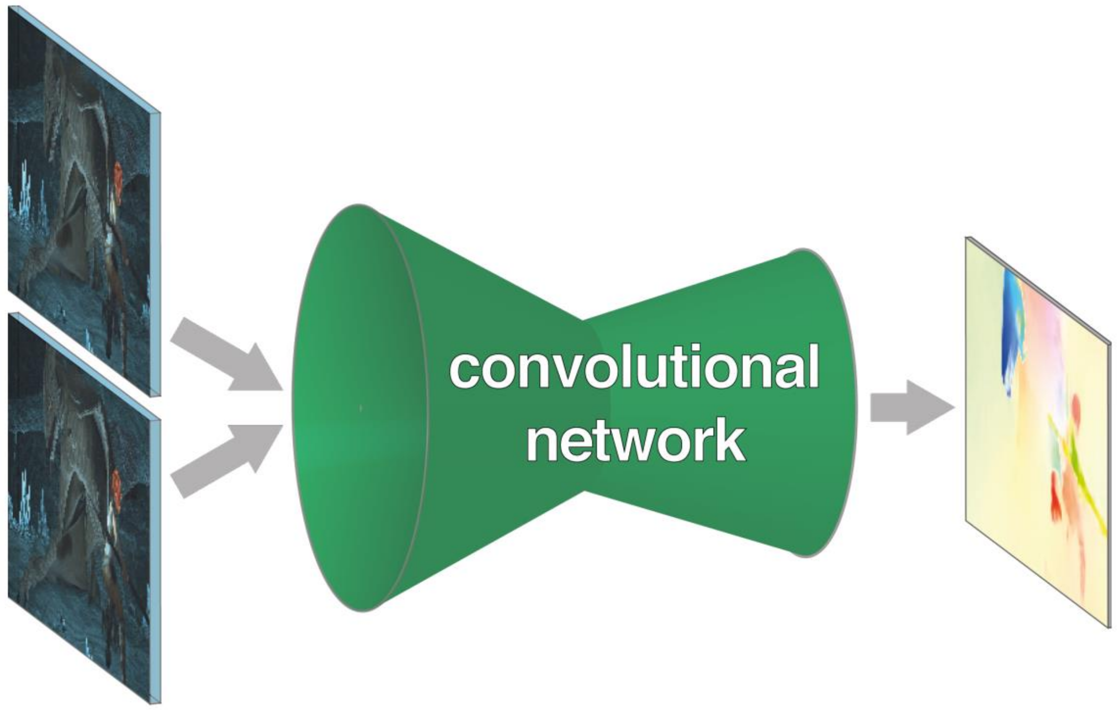


(b) Sequence 05.

[1] Are we ready for autonomous driving? the KITTI vision benchmark suite, A. Geiger et al. CVPR 2012

FlowNet

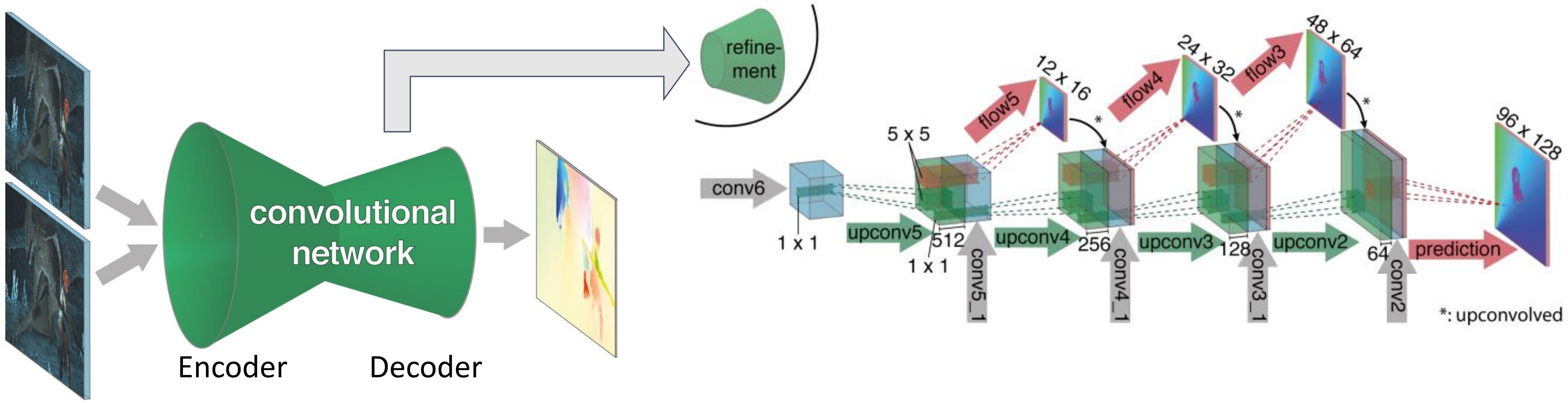
- The task here is to predict **dense correspondences** between two image, also called **optical flow**
- This is particularly challenging for model based methods if there are **large displacements**, **textureless regions** or **repetitive patterns**
- FlowNet shows how a **Convolutional Neural Network (CNN)** can solve this task



FlowNet: Learning Optical Flow with Convolutional Networks, A. Dosovitskiy et al. ICCV 2017.

FlowNet - Architecture

- FlowNet takes a stack of two images as input and consists of two modules:
  - an **Encoder** which convolves and downsamples the feature maps with learnt weights
  - a **Decoder** which convolves and upsamples the feature maps. In addtion, **intermediate flow predictions at lower resolution** are generated that help learning flow across different scales

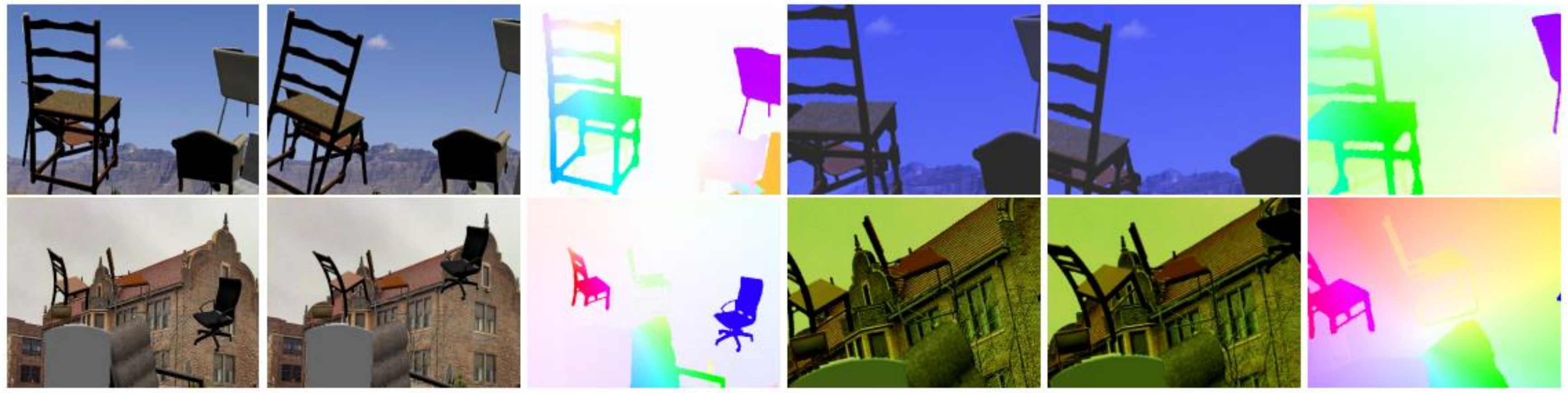


FlowNet: Learning Optical Flow with Convolutional Networks, A. Dosovitskiy et al. ICCV 2017.

FlowNet - Training

- Training is performed on a simulated dataset call the **Flying Chairs Dataset.** This dataset features pairs of images of flying chairs with random backgrounds and ground truth optical flow.
- Care was taken to move the chairs in a way **to mimic the distribution of optical flow in real data**
- During training the **average end point error** (AEE) between predicted and ground truth optical flow is minimized for all different scales

$$AEE = \frac{1}{N}\sum_i ||f_i - \hat{f}_i||_2$$

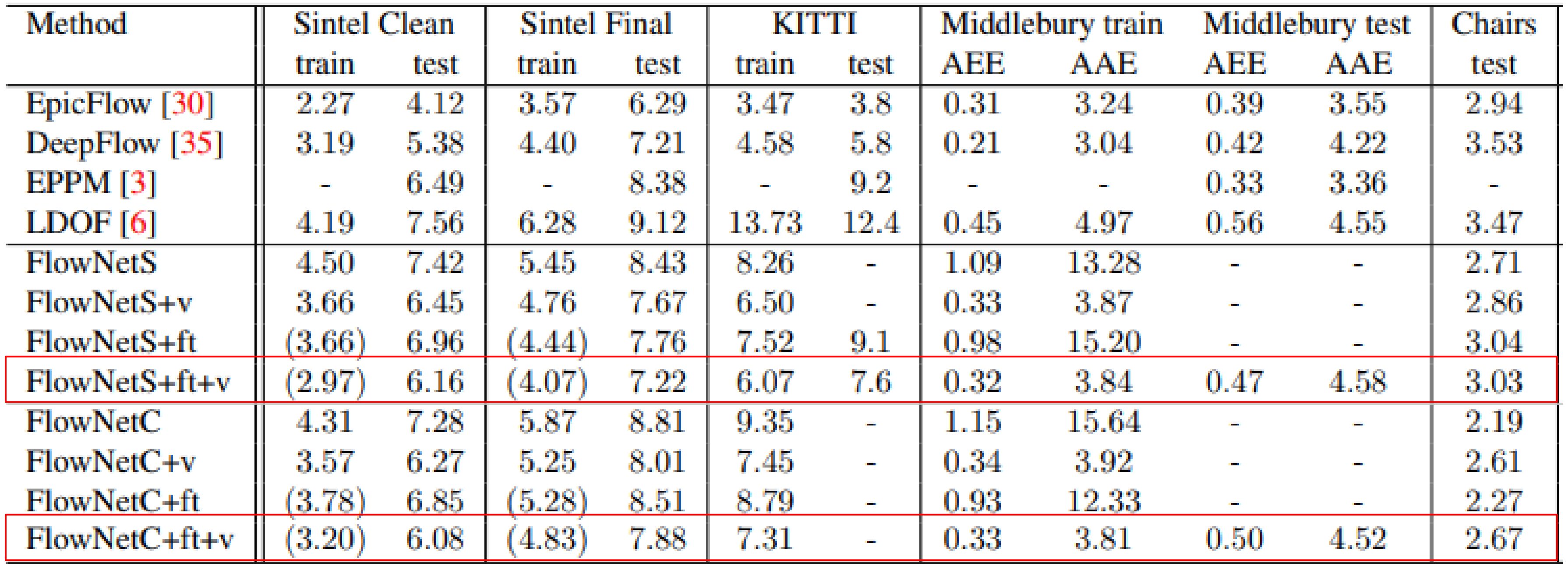


FlowNet - Results

- Although the training data for FlowNet was very crude and only consisted of chairs it generalizes surprisingly well. This is because optical flow is a **low level task,** i.e the network does not need to know about high level features like chairs to solve it.
- FlowNet outperforms model based methods on all benchmark datasets

model based methods

variations of FlowNet



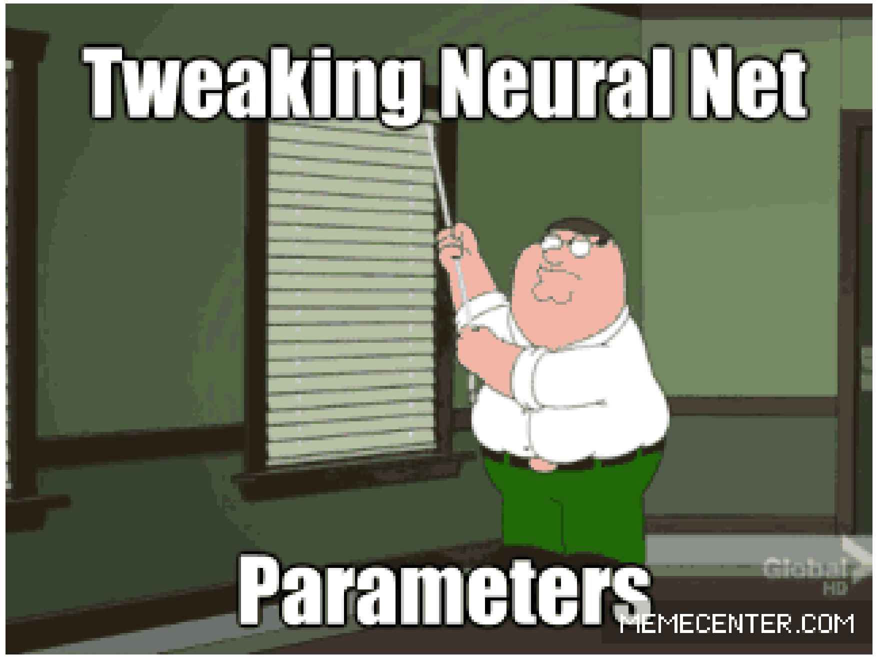
Method	Sintel Clean		Sintel Final		KITTI		Middlebury train		Middlebury test		Chairs
	train	test	train	test	train	test	AEE	AAE	AEE	AAE	test
EpicFlow [30]	2.27	4.12	3.57	6.29	3.47	3.8	0.31	3.24	0.39	3.55	2.94
DeepFlow [35]	3.19	5.38	4.40	7.21	4.58	5.8	0.21	3.04	0.42	4.22	3.53
EPPM [3]	-	6.49	-	8.38	-	9.2	-	-	0.33	3.36	-
LDOF [6]	4.19	7.56	6.28	9.12	13.73	12.4	0.45	4.97	0.56	4.55	3.47
FlowNetS	4.50	7.42	5.45	8.43	8.26	-	1.09	13.28	-	-	2.71
FlowNetS+v	3.66	6.45	4.76	7.67	6.50	-	0.33	3.87	-	-	2.86
FlowNetS+ft	(3.66)	6.96	(4.44)	7.76	7.52	9.1	0.98	15.20	-	-	3.04
FlowNetS+ft+v	(2.97)	6.16	(4.07)	7.22	6.07	7.6	0.32	3.84	0.47	4.58	3.03
FlowNetC	4.31	7.28	5.87	8.81	9.35	-	1.15	15.64	-	-	2.19
FlowNetC+v	3.57	6.27	5.25	8.01	7.45	-	0.34	3.92	-	-	2.61
FlowNetC+ft	(3.78)	6.85	(5.28)	8.51	8.79	-	0.93	12.33	-	-	2.27
FlowNetC+ft+v	(3.20)	6.08	(4.83)	7.88	7.31	-	0.33	3.81	0.50	4.52	2.67

Conclusion

Deep Learning Limitations

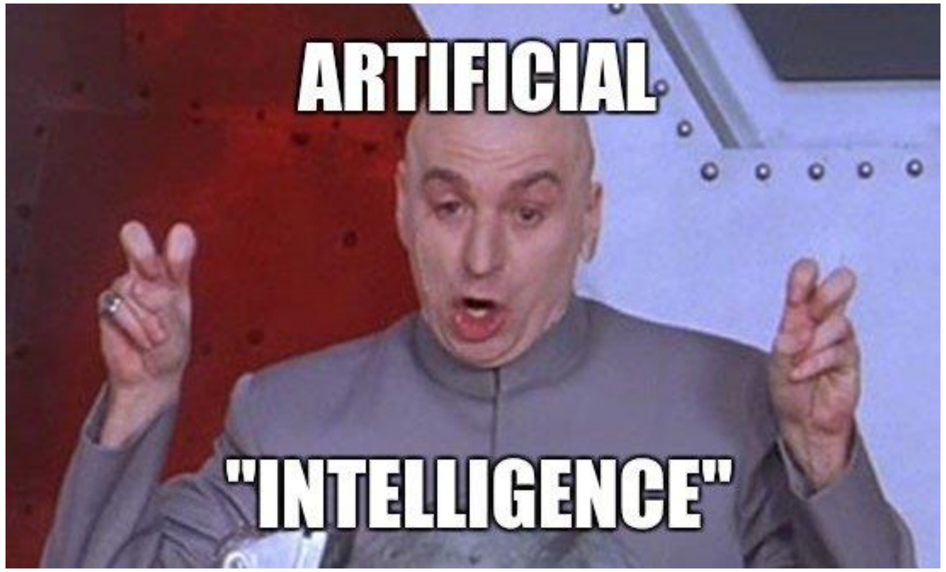
- Require **lots of data** to learn
- **Difficult debugging** and finetuning
- **Poor generalization** across similar tasks

Neural Networks Practitioners



Things to remember

- Deep Learning is able to **extract meaningful patterns** from data.
- It can be applied to **a wide range of tasks**.
- **Artificial Intelligence** ≠ **Deep Learning**



Come over for projects in DL!

Visit our webpage for projects!

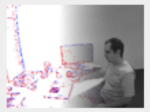
http://rpg.ifi.uzh.ch/student_projects.php

Learning 3D Reconstruction using an Event Camera

Description
Event cameras such as the Dynamic Vision Sensor (DVS) are recent sensors with large potential for high-speed and high dynamic range robotic applications. In particular, they have been used to generate high speed video and for high speed visual odometry. In this project we want to explore the possibility using an event camera to do asynchronous 3D reconstruction with very high temporal resolution. These properties are critical in applications such as fast obstacle avoidance and fast mapping. Applicants should have a background in C++ programming and low-level vision. In addition, familiarity with learning frameworks such as pytorch or tensorflow are required.

Contact Details
Daniel Gehrig (dgehrig (at) ifi (dot) uzh (dot) ch), Mathias Gehrig (mgehrig (at) ifi (dot) uzh (dot) ch)

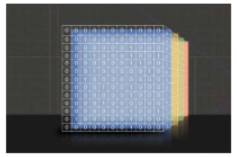
Learning an Event Camera



Description
Event cameras such as the Dynamic Vision Sensor (DVS) are recent sensors with a lot of potential for high-speed and high dynamic range robotic applications. They have been successfully applied in many applications, such as high speed video and high speed visual odometry. In spite of this success, the exact operating principle of event cameras, that is, how events are generated from a given visual signal and how noise is generated, is not well understood. In his work we want to explore new techniques for modelling the generation of events in an event camera, which would have wide implications for existing techniques. Applicants should have a background in C++ programming and low-level vision. In addition, familiarity with learning frameworks such as pytorch or tensorflow are required.

Contact Details
Daniel Gehrig (dgehrig (at) ifi (dot) uzh (dot) ch), Mathias Gehrig (mgehrig (at) ifi (dot) uzh (dot) ch)

High-Performance Simulation of Spiking Neural Network on GPUs - Available



Description: One major complication in research of biologically-inspired spiking neural Networks (SNNs) is simulation performance on conventional hardware (CPU/GPU). Computation in SNNs is dominated by operations on sparse tensors but usually this potential benefit is ignored to save development time. However, the exploitation of sparsity could be beneficial to scale simulation of SNNs to larger datasets. Requirements: - Experience with deep learning frameworks (e.g. TensorFlow or PyTorch) - Excellent programming skills and experience in CUDA

Goal: In this project, you will leverage sparse computation to develop high-performance simulations of SNNs that can be used for optimization. This will help to scale experiments and drastically improve results obtained by SNNs.

Contact Details: Mathias Gehrig, mgehrig (at) ifi (dot) uzh (dot) ch

Thesis Type: Semester Project

See project on SiROP

Learning to Deblur Images with Events - Available

Description: Images suffer from motion blur due to long exposure in poor light condition or rapid motion. Unlike conventional cameras, event-cameras do not suffer from motion blur. This is due to the fact that event-cameras provide events together with the exact time when they were triggered. In this project, we will make use of hybrid sensors which provide both conventional images and events such that we can leverage the advantages of both. Requirements: - Background in computer vision and machine learning - Deep learning experience preferable but not strictly required - Programming experience in C++ and Python

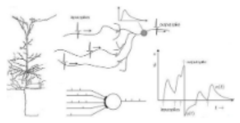
Goal: The goal is to develop an algorithm capable producing a blur-free image from the captured, blurry image, and events within the exposure time. To this end, synthetic data can be generated by our simulation framework which is able to generate both synthetic event data and motion blurred images. This data can be used by machine learning algorithms designed to solve the task at hand. At the end of the project, the algorithm will be adapted to perform optimally with real-world data.

Contact Details: Mathias Gehrig (mgehrig at ifi.uzh.ch); Daniel Gehrig (dgehrig at ifi.uzh.ch)

Thesis Type: Semester Project / Master Thesis

See project on SiROP

Optimization for Spiking Neural Networks - Available



Description: Spiking neural networks (SNNs) are neural networks that process information with timing of events/spikes rather than numerical values. Together with event-cameras, SNNs show promise to both lower latency and computational burden compared to artificial neural networks. In recent years, researchers have proposed several methods to estimate gradients of SNN parameters in a supervised learning context. In practice, many of these approaches rely on assumptions that might not hold in all scenarios. Requirements: - Background in machine learning; especially deep learning - Good programming skills; experience in CUDA is a plus.

Goal: In this project we explore state-of-the-art optimization methods for SNNs and their suitability to solve the temporal credit-assignment problem. As a first step, an in-depth evaluation of a selection of algorithms is required. Based on the acquired insights, the prospective student can propose improvements and implement their own method.

Contact Details: Mathias Gehrig, mgehrig (at) ifi (dot) uzh (dot) ch

Thesis Type: Master Thesis

See project on SiROP

Additional Readings

- Neural Networks and Deep Learning, by Michael Nielsen [Chapter 2]
- Practical Recommendations for Gradient-Based Training of Deep Architectures, Y. Bengio
- Deep Learning, I. Goodfellow, Y. Bengio, A. Courville
- All the references above!