CVPRVIRTUAL

Differentiable Approaches for Robust Estimation

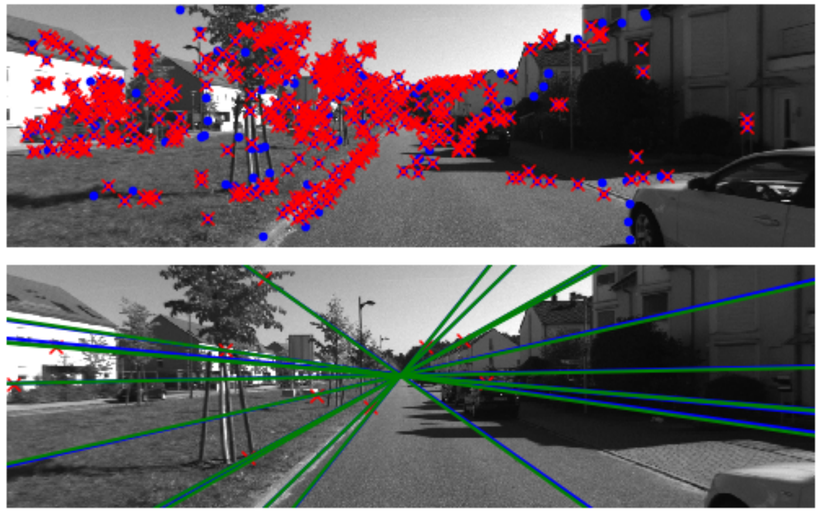
René Ranftl, Intel Labs

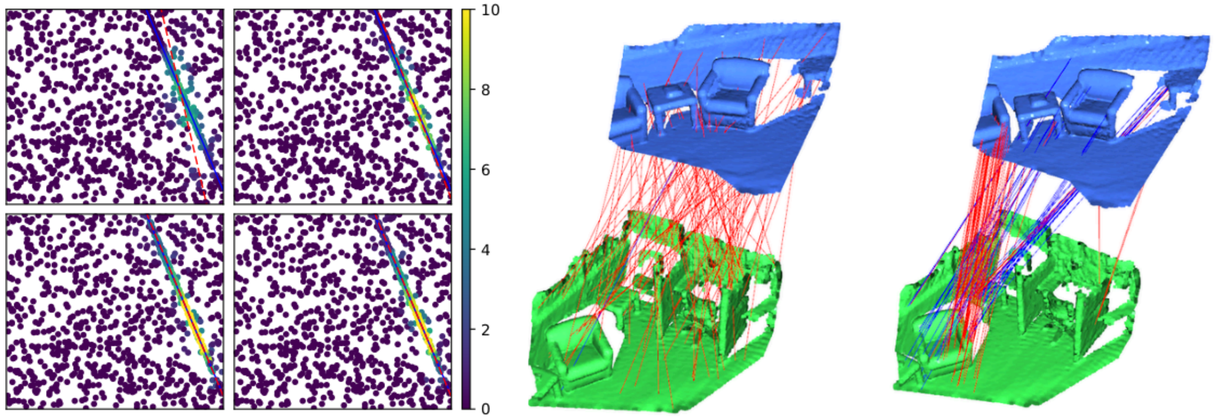
RANSAC in 2020: CVPR tutorial

Differentiable approaches

Differentiable == learnable

- Deep networks in the loop
  - Trained with stochastic gradient descent
  - Learn statistical properties of the data
  - Can integrate additional information
- Challenges:
  - High-dimensional, unordered data
  - Acquiring training data
  - Guaranteeing transfer
  - Design of loss functions





Differentiable approaches

An active research topic

- Current work mostly targeting specific applications
  - Need for training data
  - Requirements for integrating domain knowledge
- Two research clusters:
  - Two-view correspondence filtering (2D-2D)
  - 3D Point cloud registration (3D-3D)
- Other applications: Camera localisation (2D-3D), vanishing point estimation

Differentiable approaches

An active research topic

- Current work mostly targeting specific applications
  - Need for training data
  - Requirements for integrating domain knowledge
- Two research clusters:
  - Two-view correspondence filtering (2D-2D)
  - 3D Point cloud registration (3D-3D)
- Other applications: Camera localisation (2D-3D), vanishing point estimation

Differentiable vs. classic

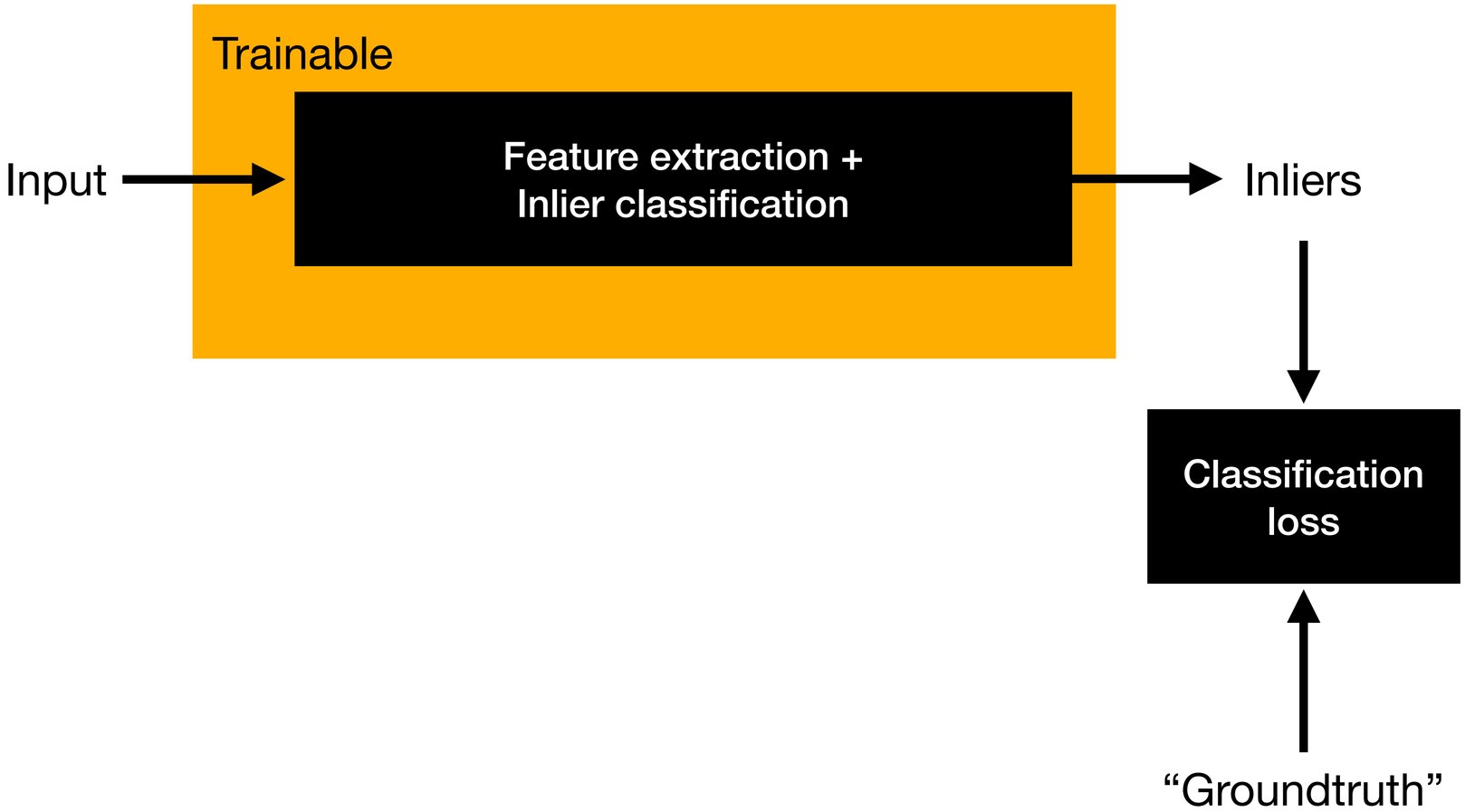
	Classic approach	Differentiable approach
Training data	No	Yes
Architecture specification	No	Yes
Loss function	Yes	Yes
All parts differentiable	No	(Yes)
Minimal solvers	(Yes)	(No)
Sampling-based	Yes	(No)
IRLS	No	(Yes)

- Promises of differentiable methods:
  - Adapt to scenario
  - Improve with more data and capacity
  - Leverage additional information
  - Complementary to classic approaches

Overview

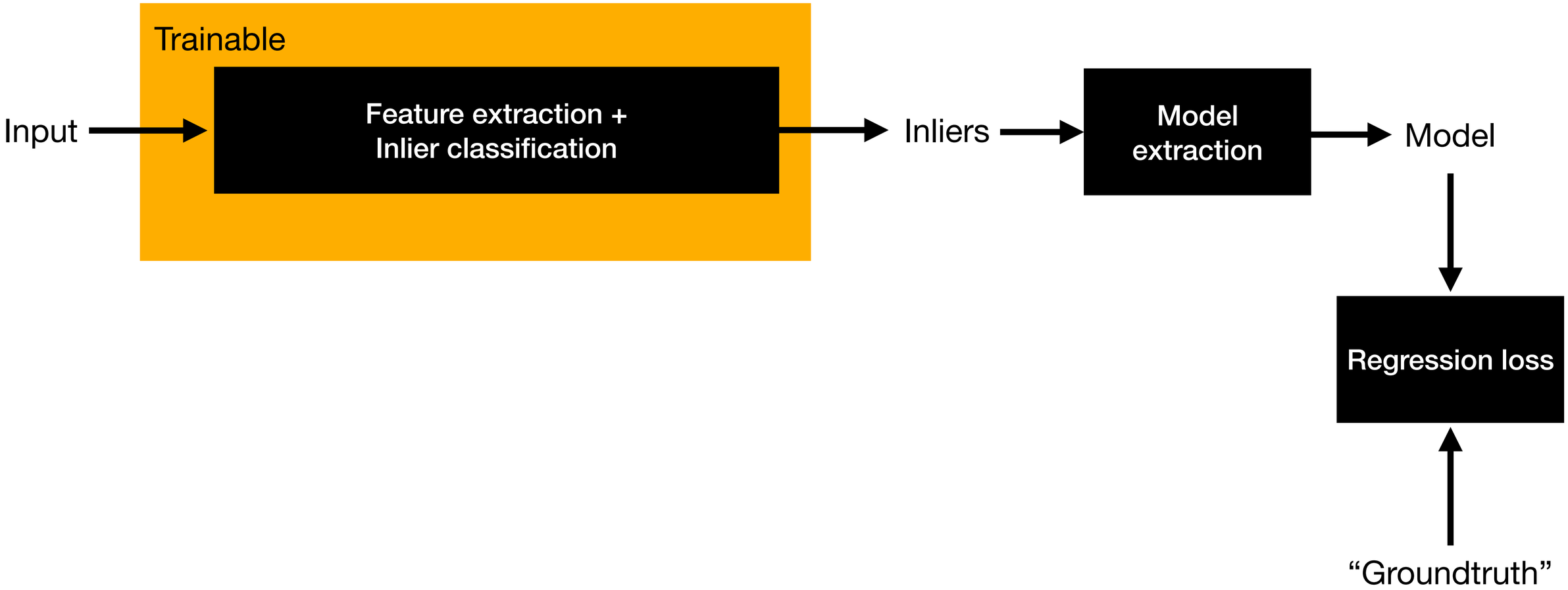
A prototypical pipeline

Training



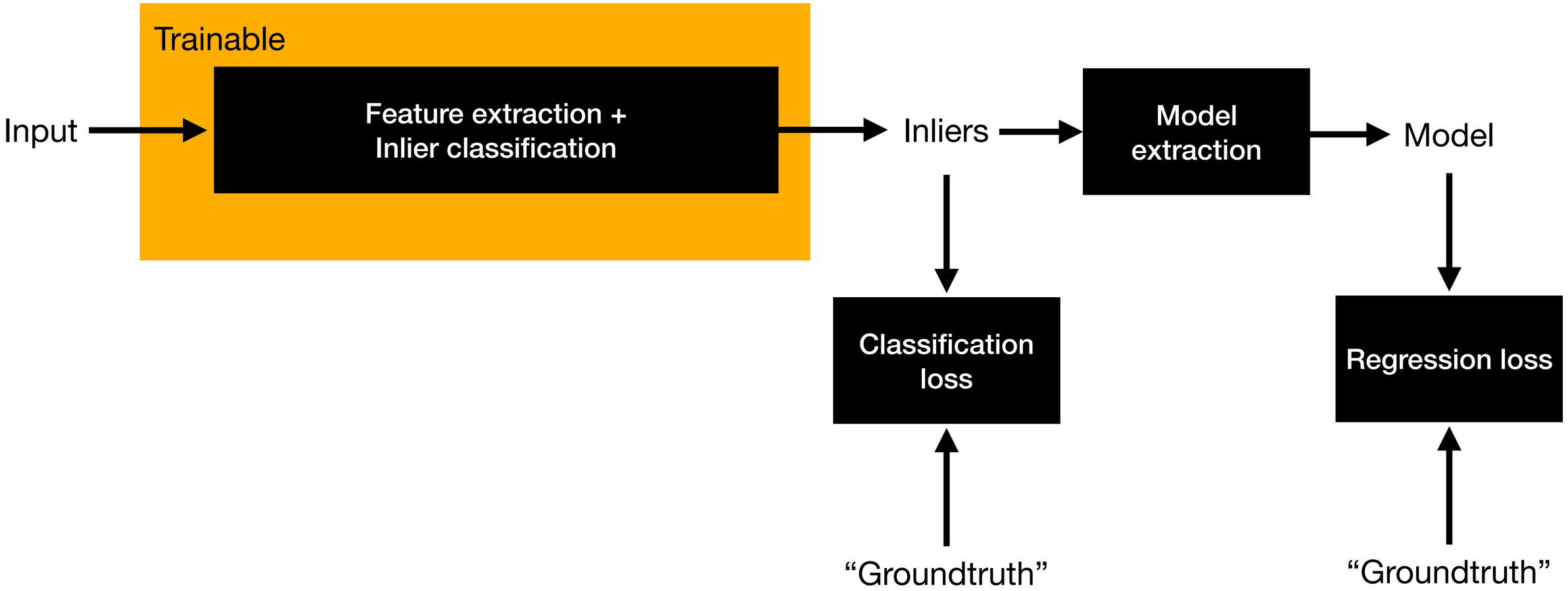
A prototypical pipeline

Training



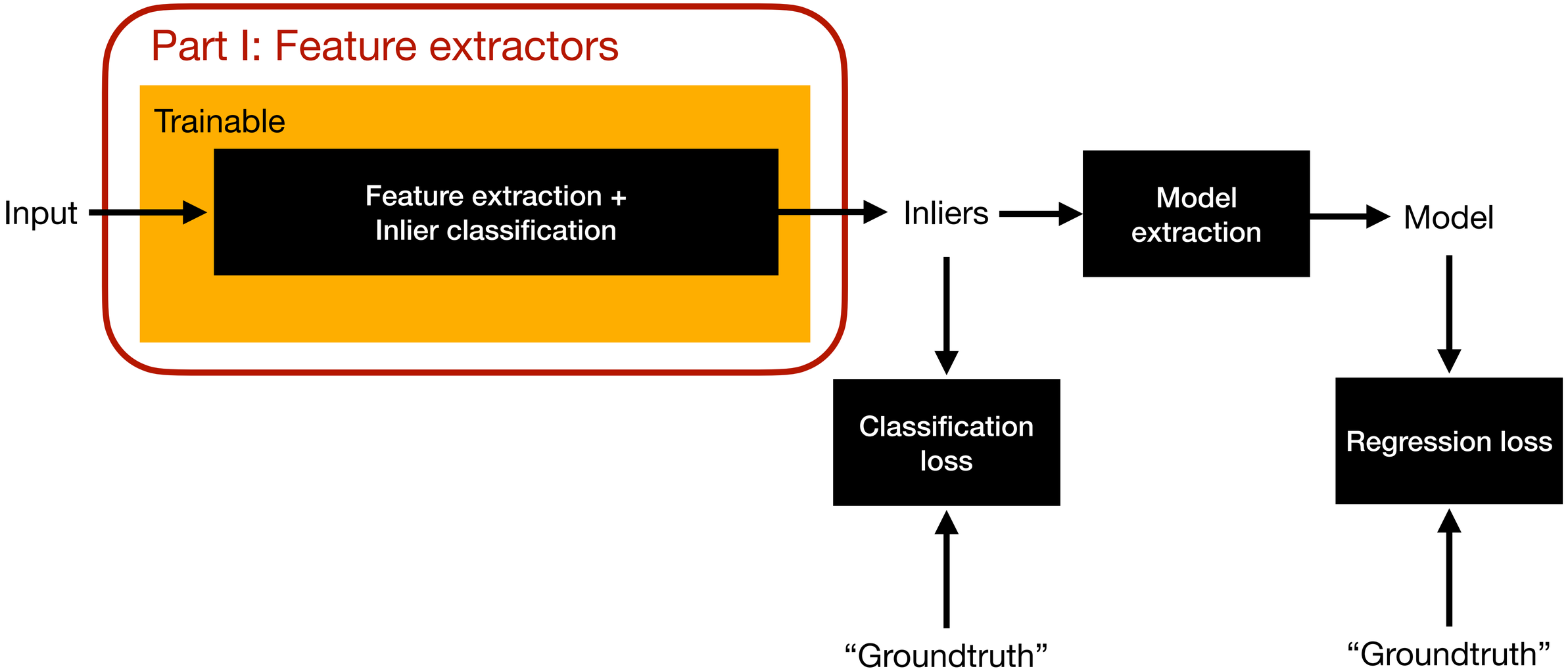
A prototypical pipeline

Training



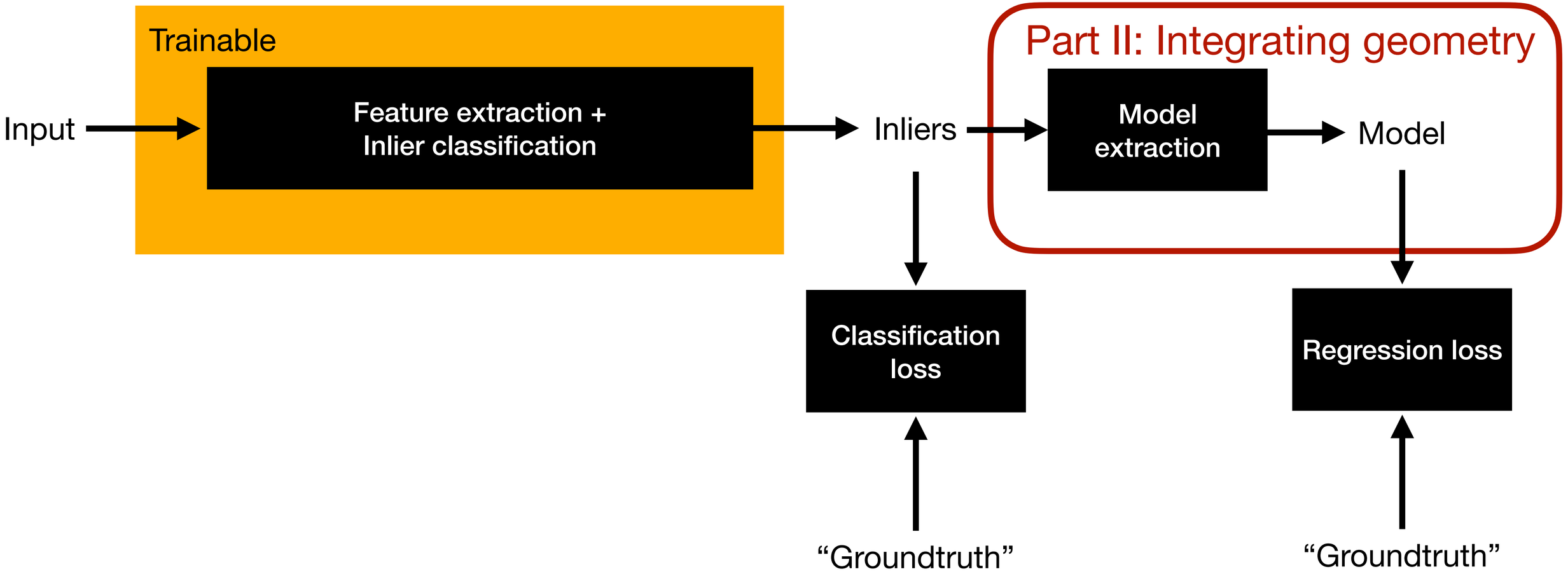
A prototypical pipeline

Training



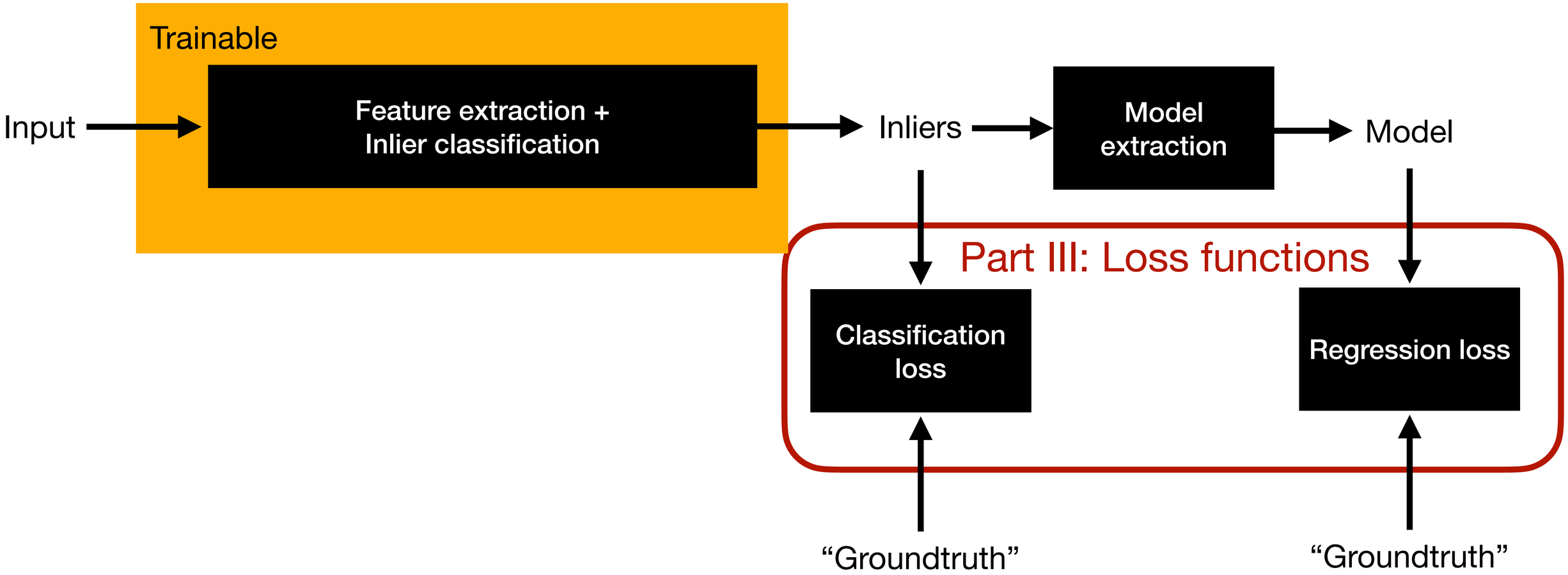
A prototypical pipeline

Training



A prototypical pipeline

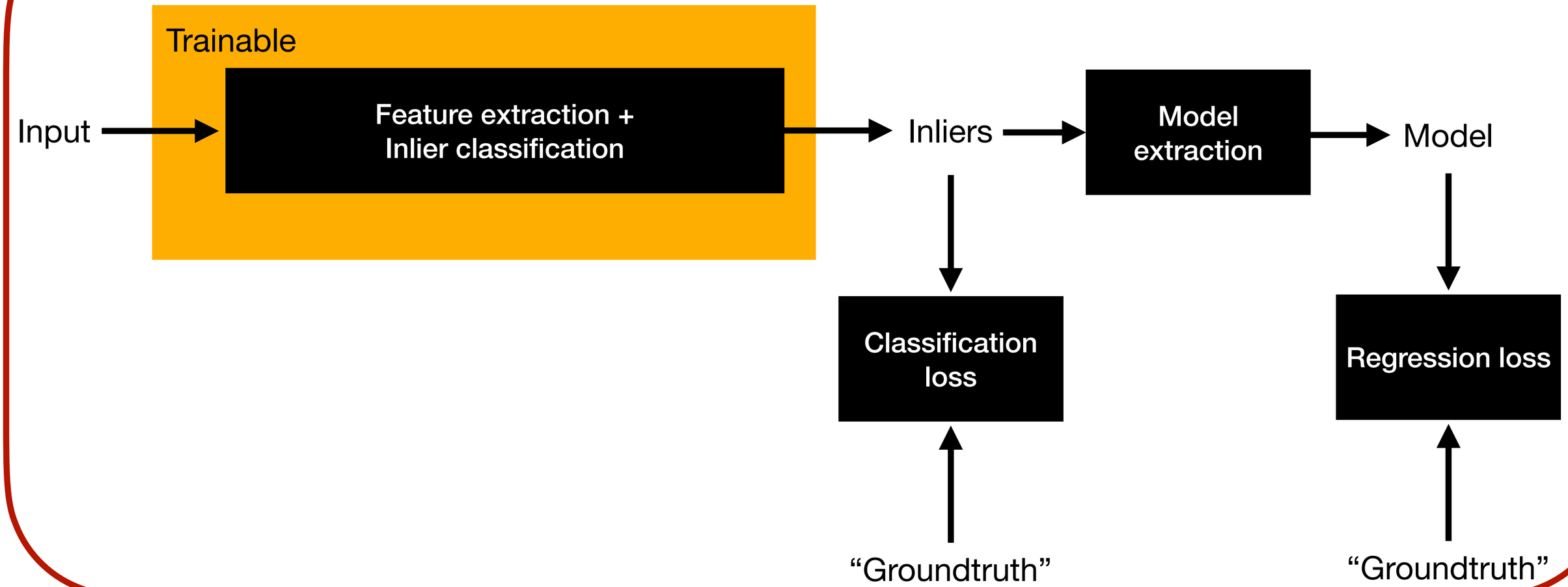
Training



A prototypical pipeline

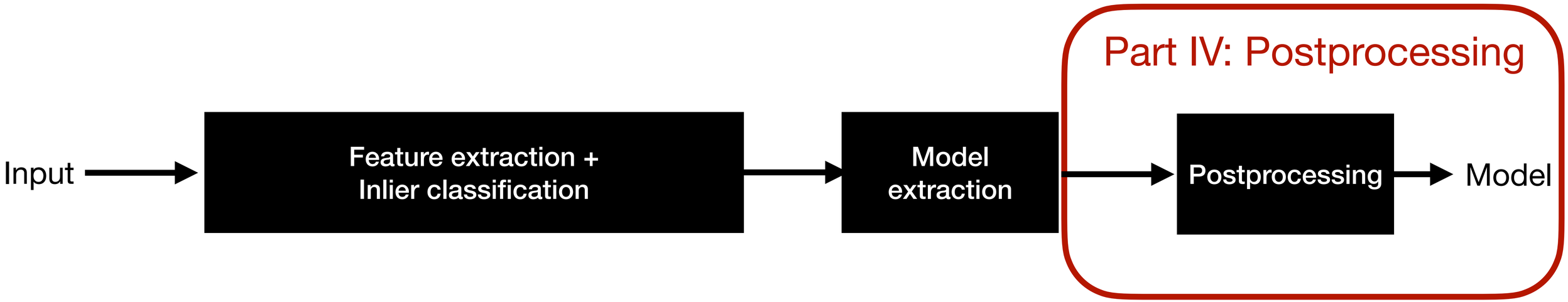
Training

Part IV: Practical considerations

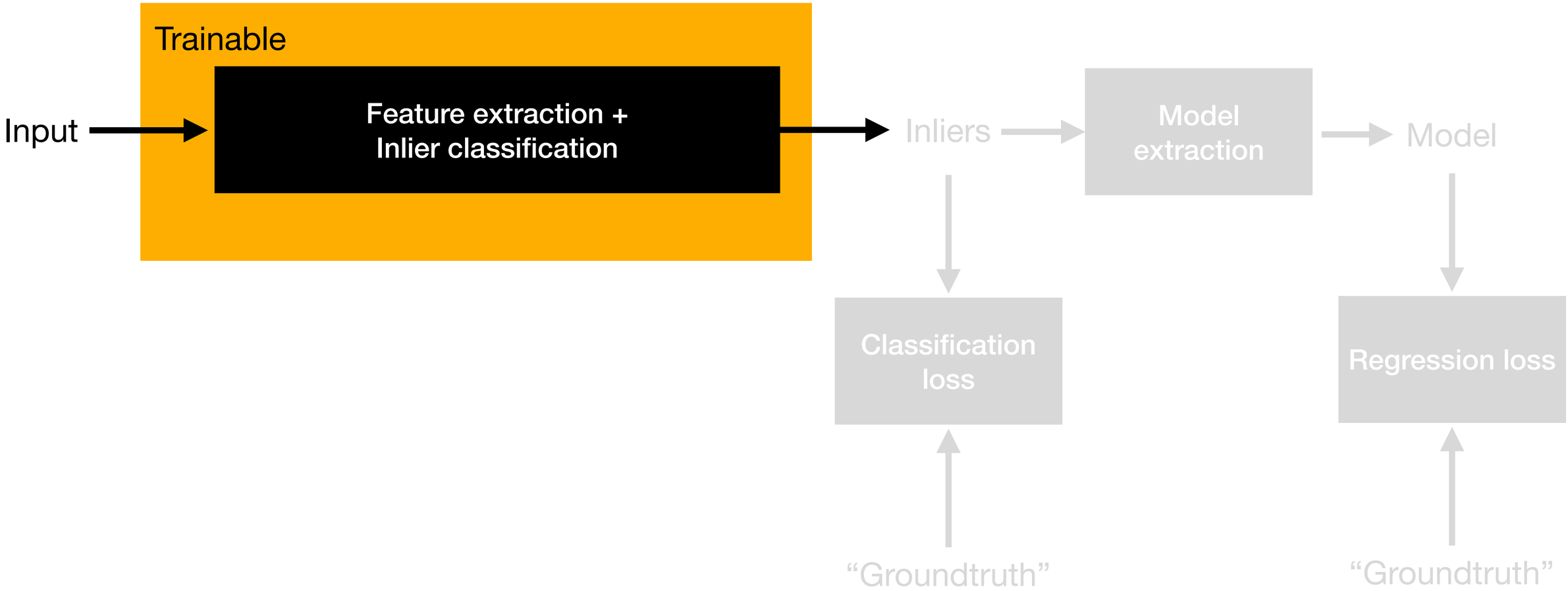


A prototypical pipeline

Testing

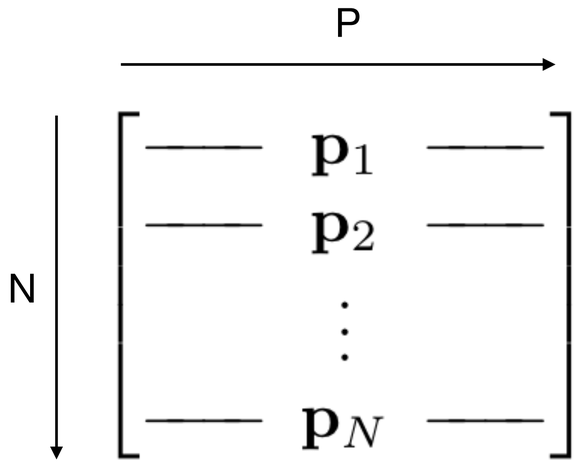


Part I: Feature extractors



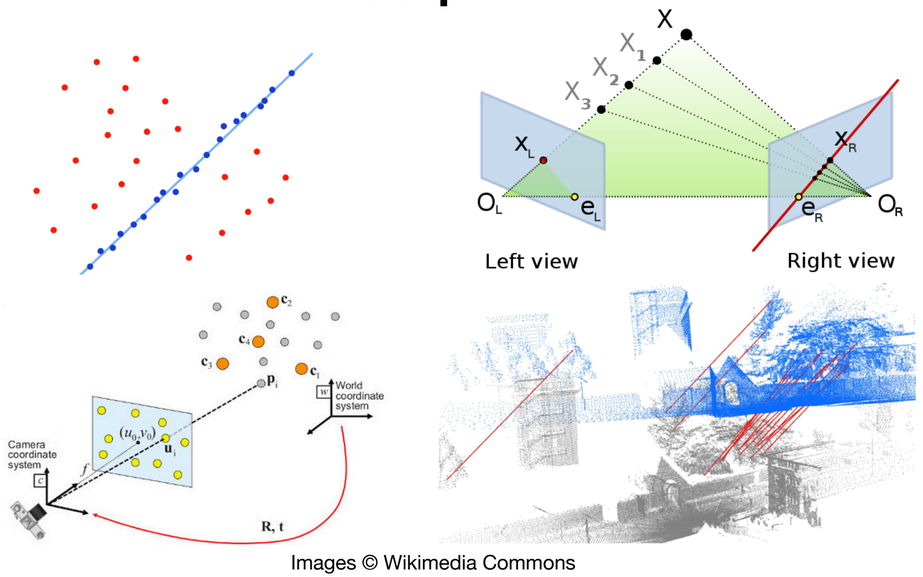
Input and output data

Input



- Putative correspondences
- Putative correspondences + additional properties
- Points in high-dimensional space

Output



Images © Wikimedia Commons

- Fundamental/Essential Matrix, Homography
- Camera pose
- Parametric curve

Requirements

- Problems require networks defined on **sets**:
  - Permutation-equivariance
  - Independent of size

- Prototypical networks:
  - PointNet [Qi 2016]
  - DeepSets [Zaheer 2017]

Set functions in deep learning

Permutation-equivariance

A function is permutation-equivariant if and only if

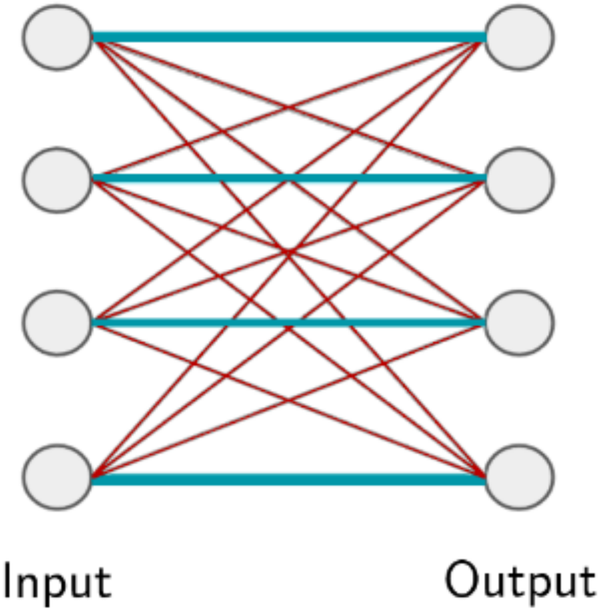
$$\mathbf{f}([p_{\pi(i)}, \ldots, p_{\pi(N)}]) = [f_{\pi(1)}(\mathbf{p}), \ldots, f_{\pi(N)}(\mathbf{p})]$$

for all permutations π

A network layer $f_\Theta(\mathbf{p}) = \sigma(\Theta\mathbf{p})$

is permutation-equivariant if and only if

$$\Theta = \lambda \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix} + \gamma \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & 1 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 1 \end{bmatrix} \qquad \lambda, \gamma \in \mathbb{R}$$



M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. “DeepSets”, NeuRIPS 2017

Set functions in deep learning

Permutation-equivariance

A function is permutation-equivariant if and only if

$$\mathbf{f}([p_{\pi(i)}, \ldots, p_{\pi(N)}]) = [f_{\pi(1)}(\mathbf{p}), \ldots, f_{\pi(N)}(\mathbf{p})]$$

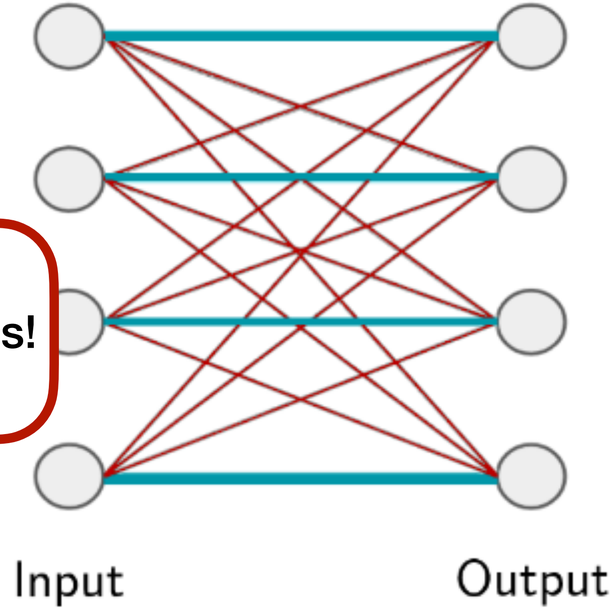
for all permutations π

A network layer $f_\Theta(\mathbf{p}) = \sigma(\Theta\mathbf{p})$

is permutation-equivariant if and only if

$$\Theta = \lambda \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix} + \gamma \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & 1 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 1 \end{bmatrix} \quad \lambda, \gamma \in \mathbb{R}$$

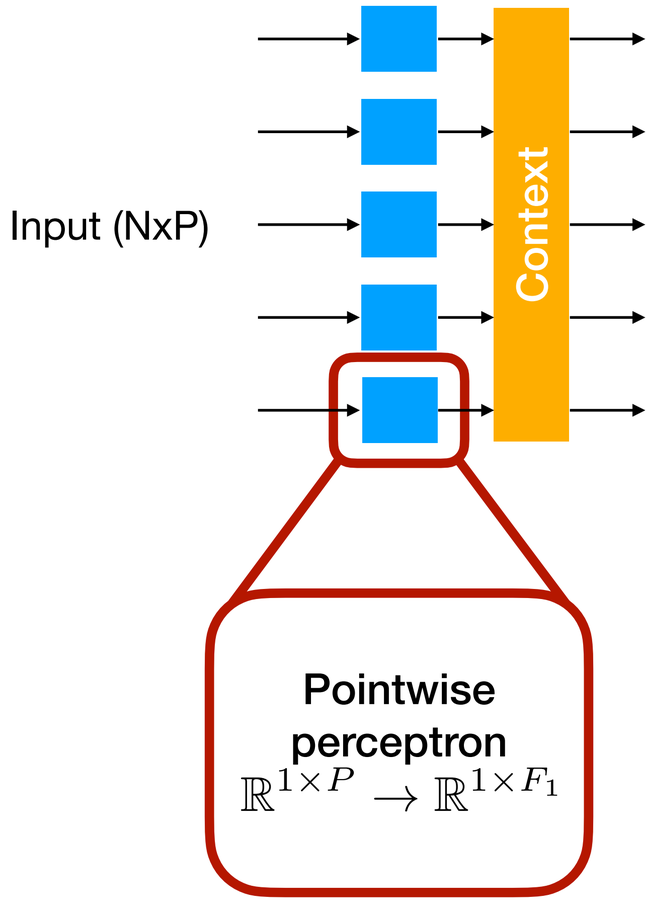
Not the complete space of permutation-equivariant functions! Simplistic model!



M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. “DeepSets”, NeuRIPS 2017

PointNet-based architectures

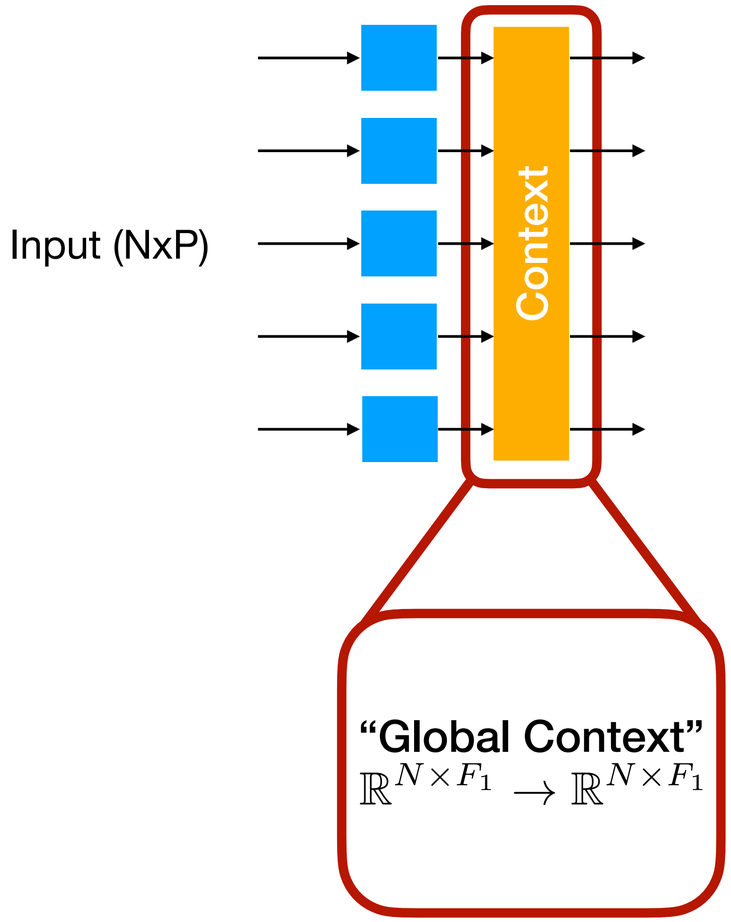
Basic building block



$$\Theta = \lambda \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} + \gamma \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

PointNet-based architectures

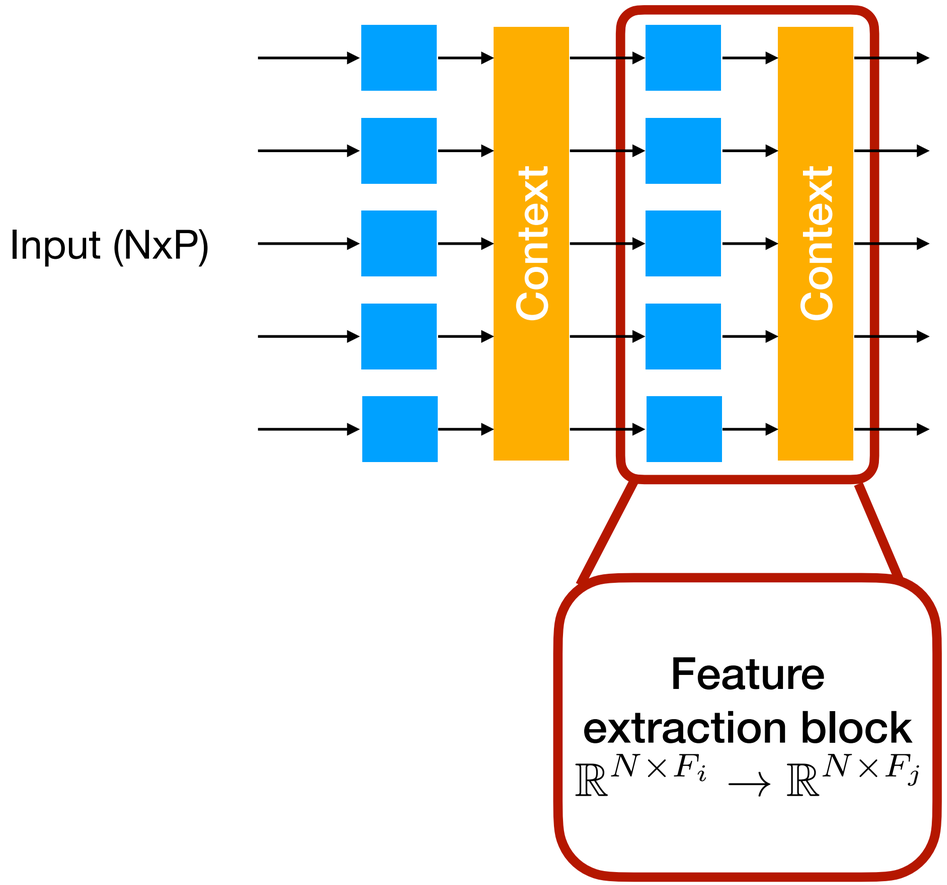
Basic building block



$$\Theta = \lambda \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} + \gamma \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

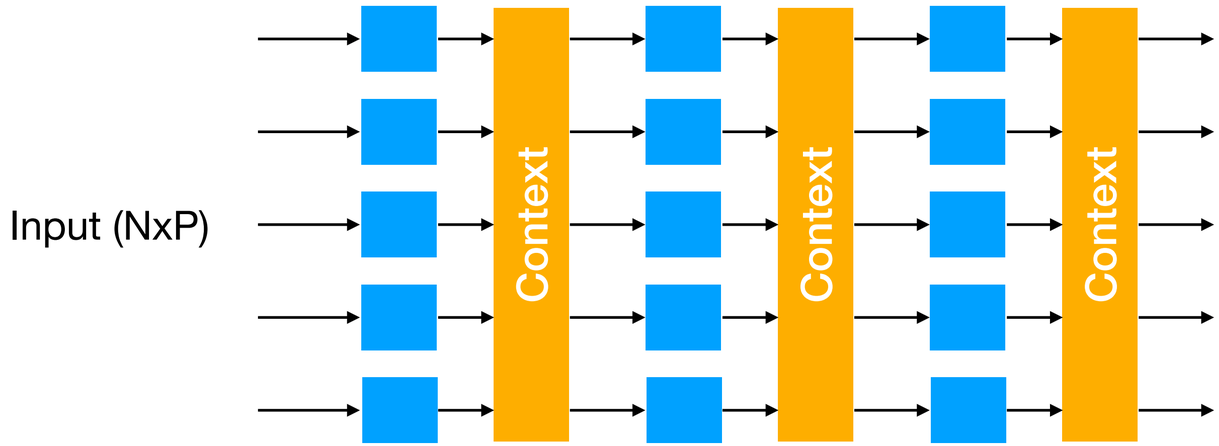
PointNet-based architectures

Basic building block



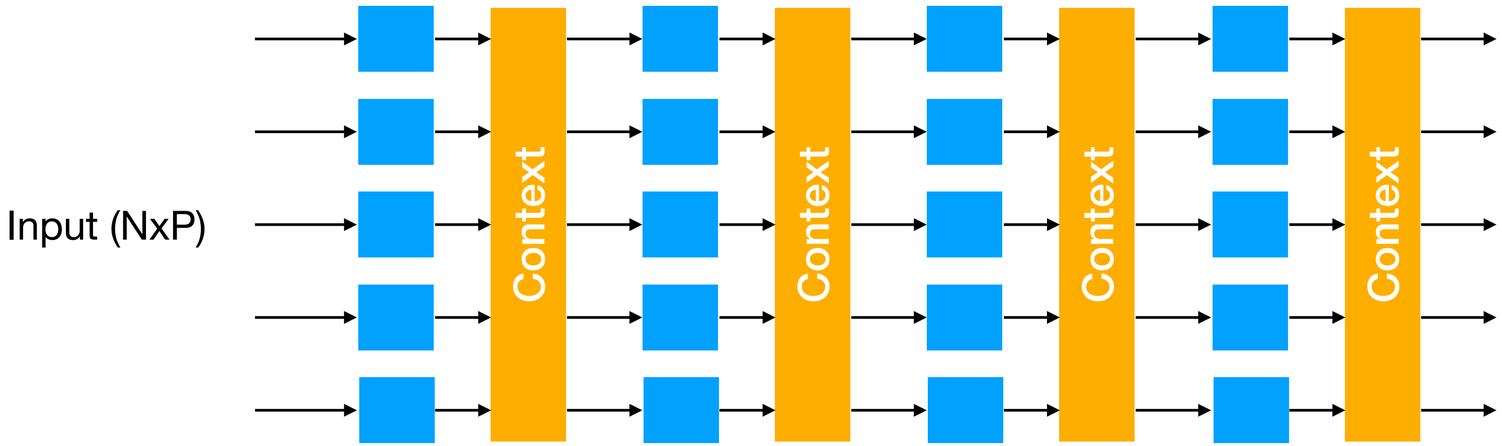
PointNet-based architectures

Basic building block



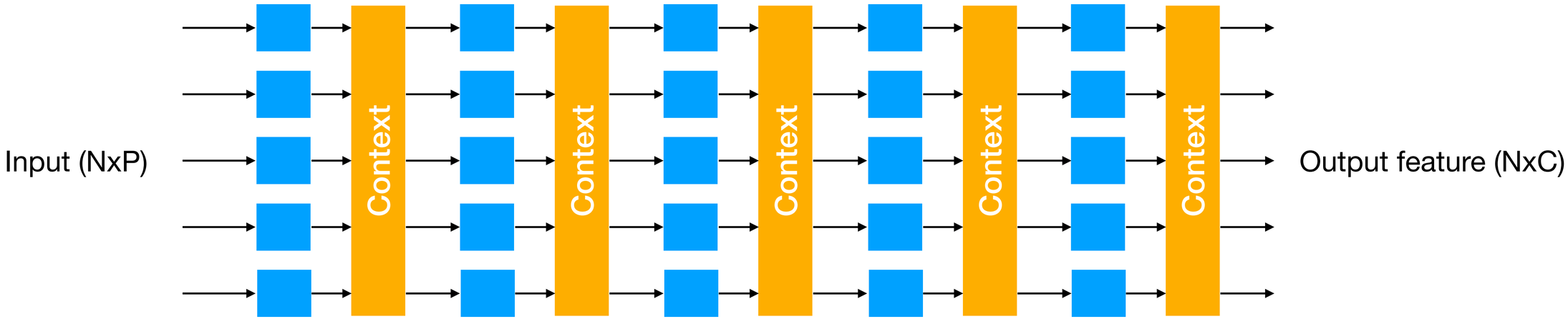
PointNet-based architectures

Basic building block



PointNet-based architectures

Basic building block

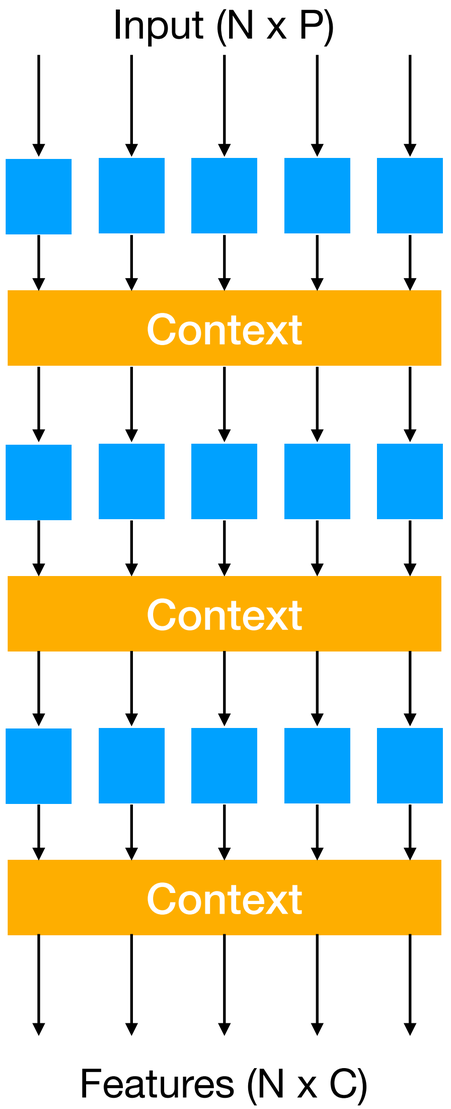


- Additional components
  - Residual connections
  - Batch normalization
  - All standard non-linearities

PointNet-based architectures

Existing architectures

- Permutation-equivariant context:
  - Context Normalization [Yi 2018, Ranftl 2018]
  - Canonical ordering [Zhao 2019, Zhang 2019]
  - Attention [Sun 2020]
- Other architectures
  - High-dimensional convolutions [Choy 2020a]



Context Normalization

AKA Instance Normalization

- Normalize feature to zero-mean and unit variance

$$\mathrm{CN}(\mathbf{f_i}) = \frac{\mathbf{f_i} - \mu}{\sigma}$$

$$\mu = \frac{1}{N}\sum_{i=1}^{N} \mathbf{f_i}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{f_i} - \mu)^2}$$

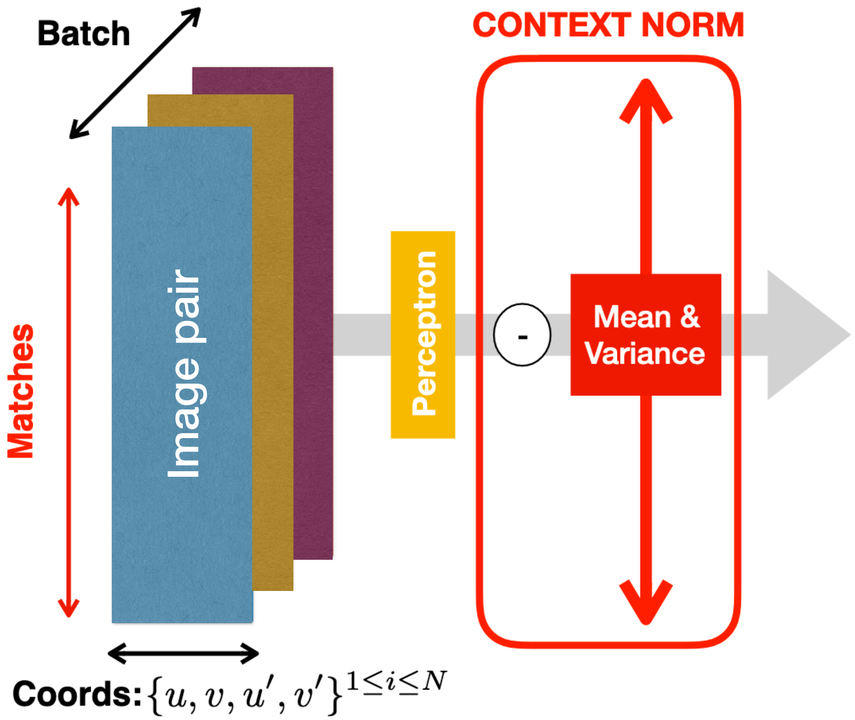


Image by Eduard Trulls
(https://etrulls.github.io/slides/cvpr18_slides.pdf)

- Operation distributes information globally
- Simple and surprisingly effective

K.M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, P. Fua. “Learning to find good correspondences”, CVPR 2018

R. Ranftl, V. Koltun. “Deep Fundamental Matrix Estimation”, ECCV 2018

Attentive Context Normalization

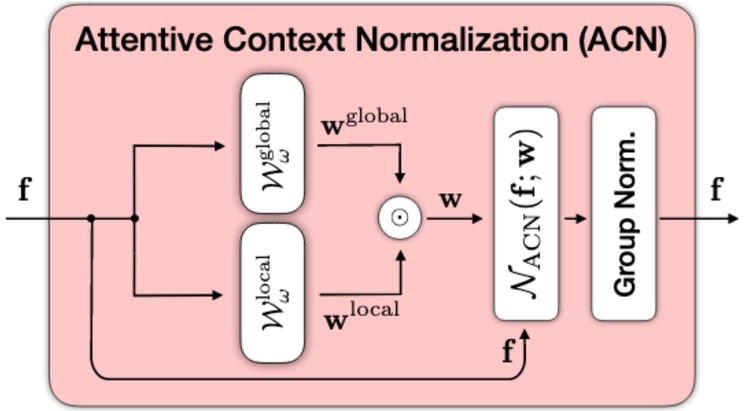
Robust version of CN

- Context Normalization treats every point equally
- Attentive CN uses weighted global statistics to focus normalization on inliers

$$ACN(\mathbf{f_i}) = \frac{\mathbf{f_i} - \mu}{\sigma}$$

$$\mu = \frac{1}{N}\sum_{i=1}^{N} w_i \mathbf{f_i}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} w_i (\mathbf{f_i} - \mu)^2}$$

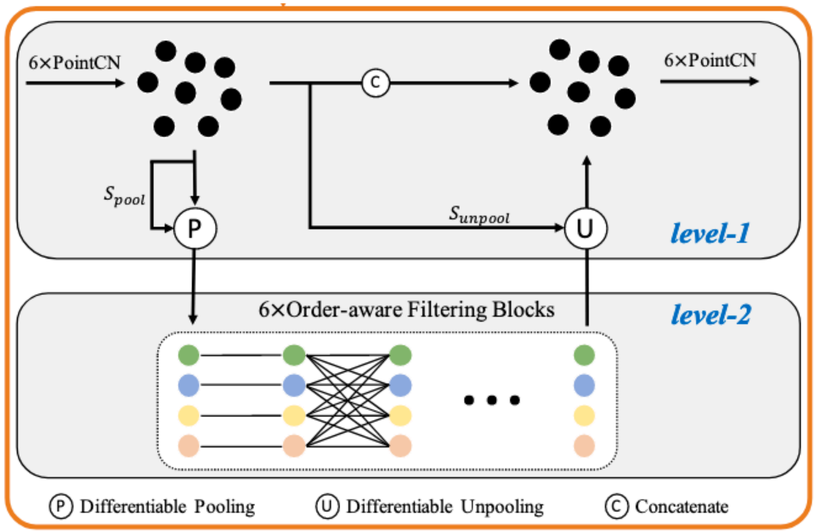


W. Sun, W. Jiang, E. Trulls, A. Tagliasacchi, K. M. Yi. “ACNe: Attentive Context Normalization For Robust, Permutation-Equivariant Learning”, CVPR 2020

Order-Aware Networks

Inferring a canonical ordering

- Proposes three operations
  - Map points to a fixed set of clusters in canonical order
  - Correlate clusters feature space
  - Unpool to append context to points

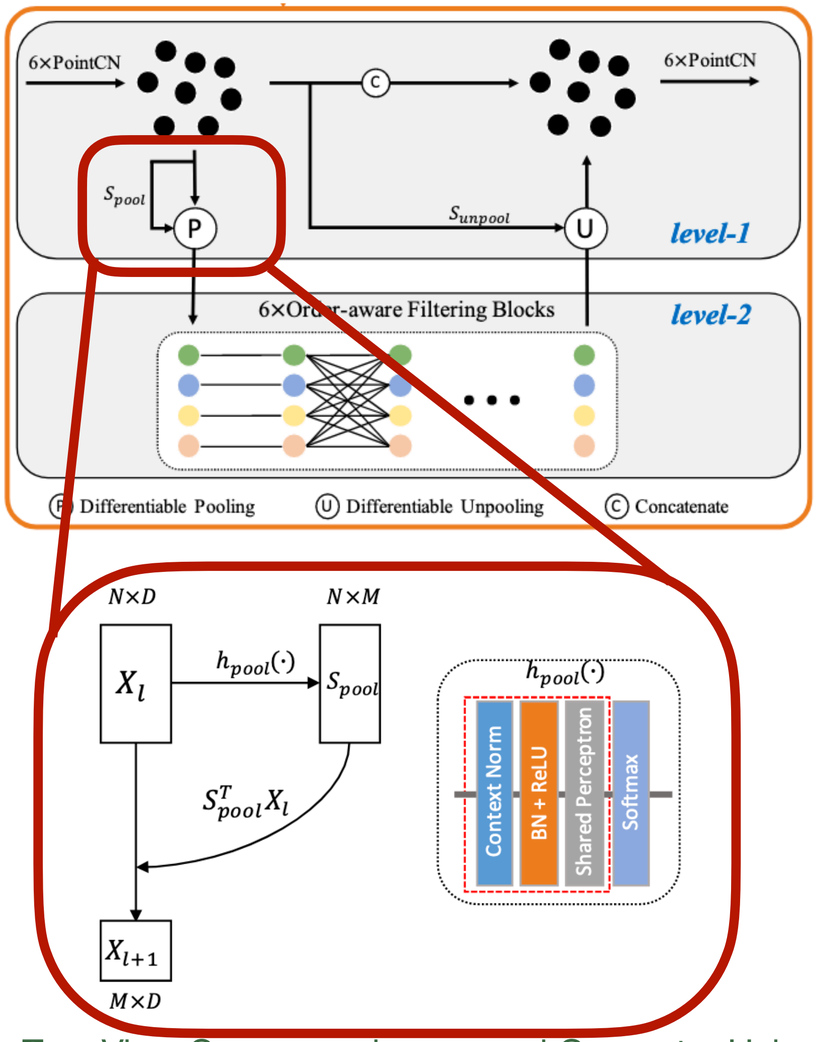


J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, H. Liao. "Learning Two-View Correspondences and Geometry Using Order-Aware Network", ICCV 2019

Order-Aware Networks

Inferring a canonical ordering

- Proposes three operations
  - Map points to a fixed set of clusters in canonical order
  - Correlate clusters feature space
  - Unpool to append context to points

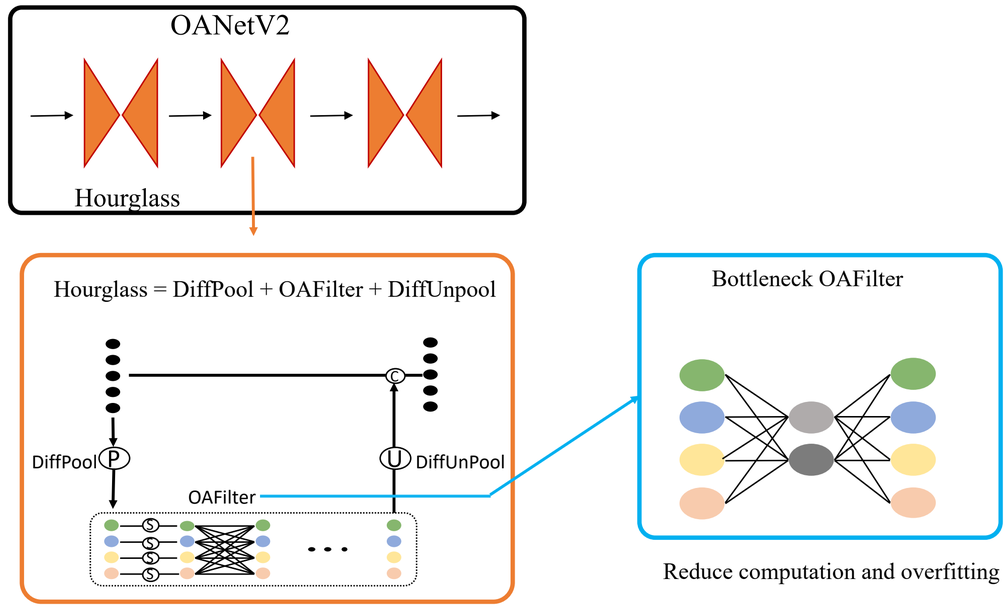


J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, H. Liao. "Learning Two-View Correspondences and Geometry Using Order-Aware Network", ICCV 2019

Order-Aware Networks v2

Inferring a canonical ordering

- Original OANet proposed one layer
- Can stack multiple networks
  - “Stacked Hourglass”
  - Bottleneck layer for reduced computation



YFCC100M	mAP5°(w/ & w/o RANSAC)
OANetV1	52.2/39.3
OANetV2	**56.8/50.5**

Thanks to Chen Hongkai for providing slides!

NM-Net

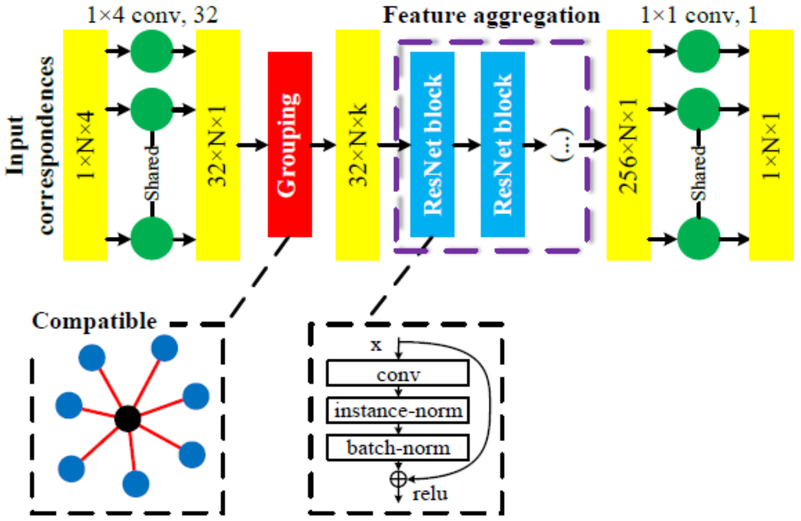
A graph-based view

- Adaptively build KNN graph based on geometric consistency
- Based on local structural compatibility

$$\mathbf{T}_i = \begin{bmatrix} \mathbf{A}_i & \mathbf{k}_i \\ \mathbf{0} & 1 \end{bmatrix} \quad \mathbf{H}_i = \mathbf{T}_i' \mathbf{T}_i^{-1} \quad e_j(c_i) = \left| \rho(\mathbf{H}_j \cdot \begin{bmatrix} \mathbf{k}_i \\ 1 \end{bmatrix}) - \rho(\mathbf{H}_i \cdot \begin{bmatrix} \mathbf{k}_i \\ 1 \end{bmatrix}) \right|$$

$$s_{ij} = e^{-\lambda(e_j(c_i)+e_i(c_j))}$$

Local affine frame Keypoint location

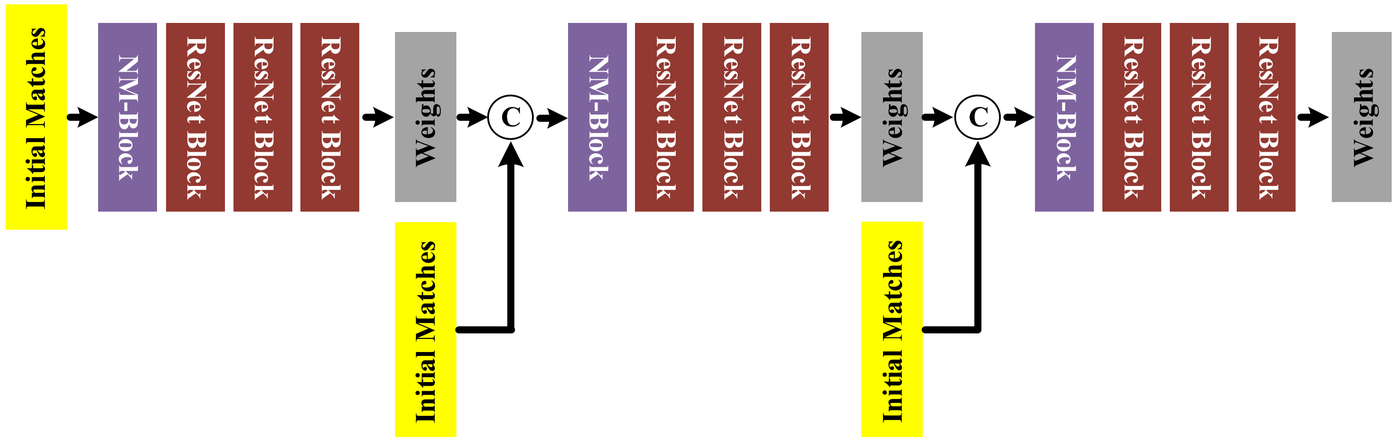


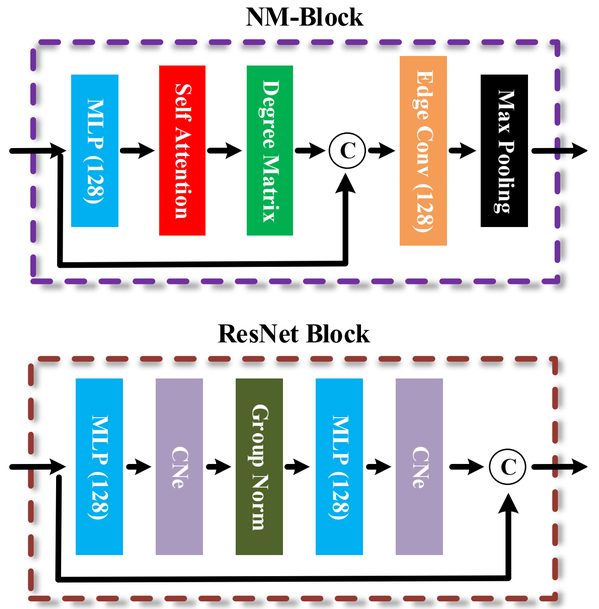
- Perform grouping and convolution on graph

C. Zhao, Z. Cao, C. Li, X. Li, and J. Yang. “NM-Net: Mining Reliable Neighbors for Robust Feature Correspondences”, CVPR 2019

NM-Net v2

A graph-based view





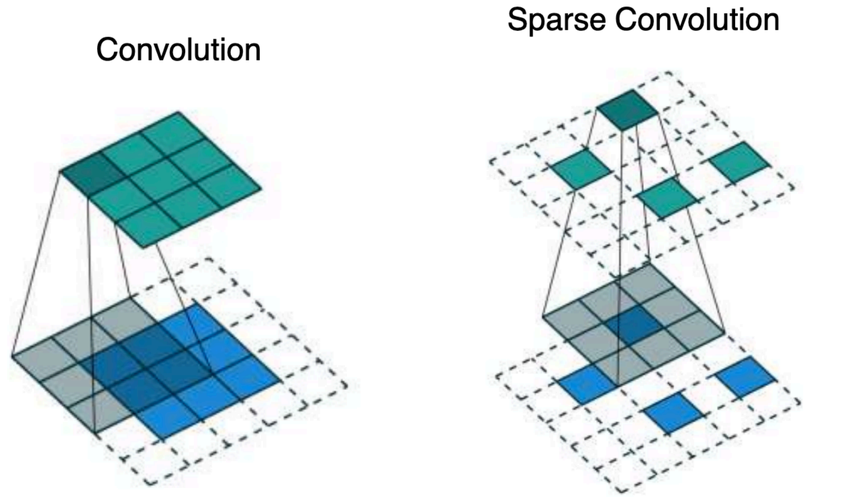
- Iterate multiple times to refine weights

Thanks to Chen Zhao for providing slides!

High-Dimensional Convolutional Networks

Leveraging insights from CNNs

- Define neighborhoods directly in voxelized high-dimensional space:
  - 4D for image matching
  - 6D for 3D point cloud registration
- Define sparse operations in this space
  - Dense not tracktable
  - Efficient implementation in MinkowskiEngine



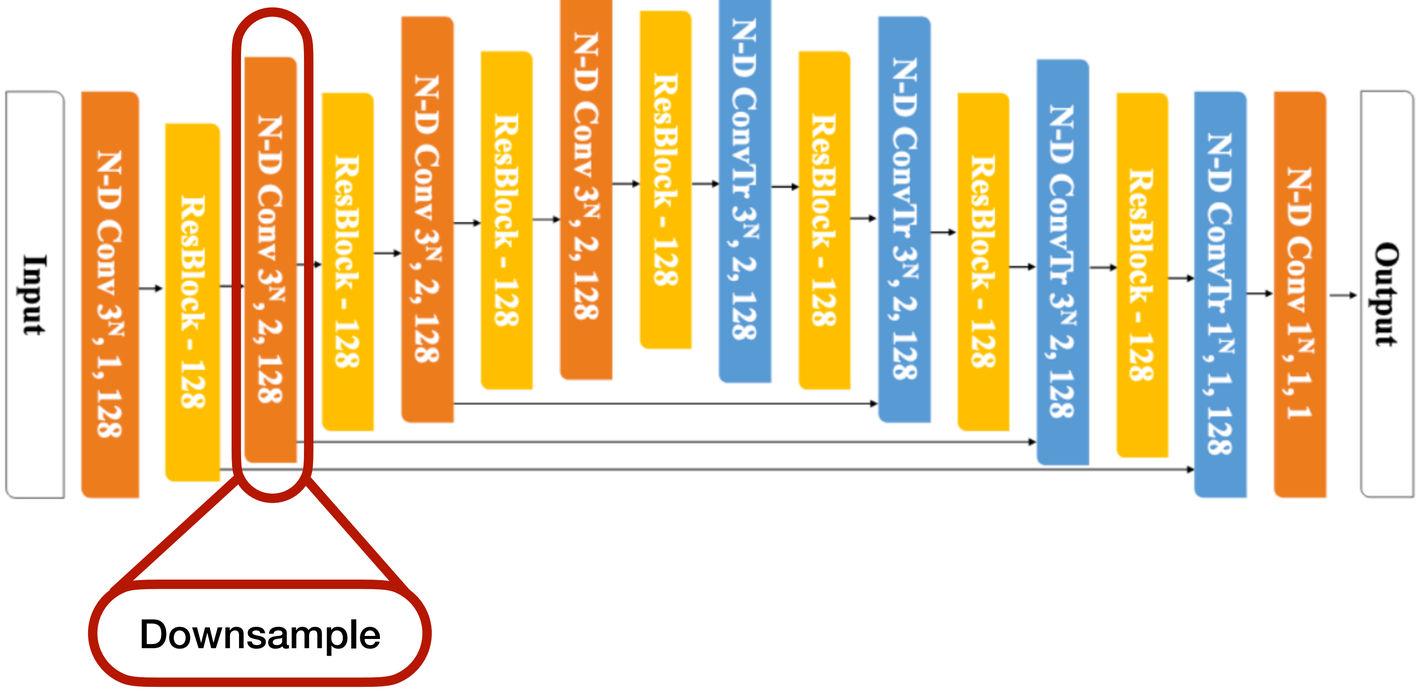
https://github.com/StanfordVL/MinkowskiEngine

C. Choy, J. Lee, R. Ranftl, J. Park, V. Koltun. “High-dimensional Convolutional Networks for Geometric Pattern Recognition”, CVPR 2020

High-Dimensional Convolutional Networks

Leveraging insights from CNNs

- U-Net-style network
- Expanding receptive field

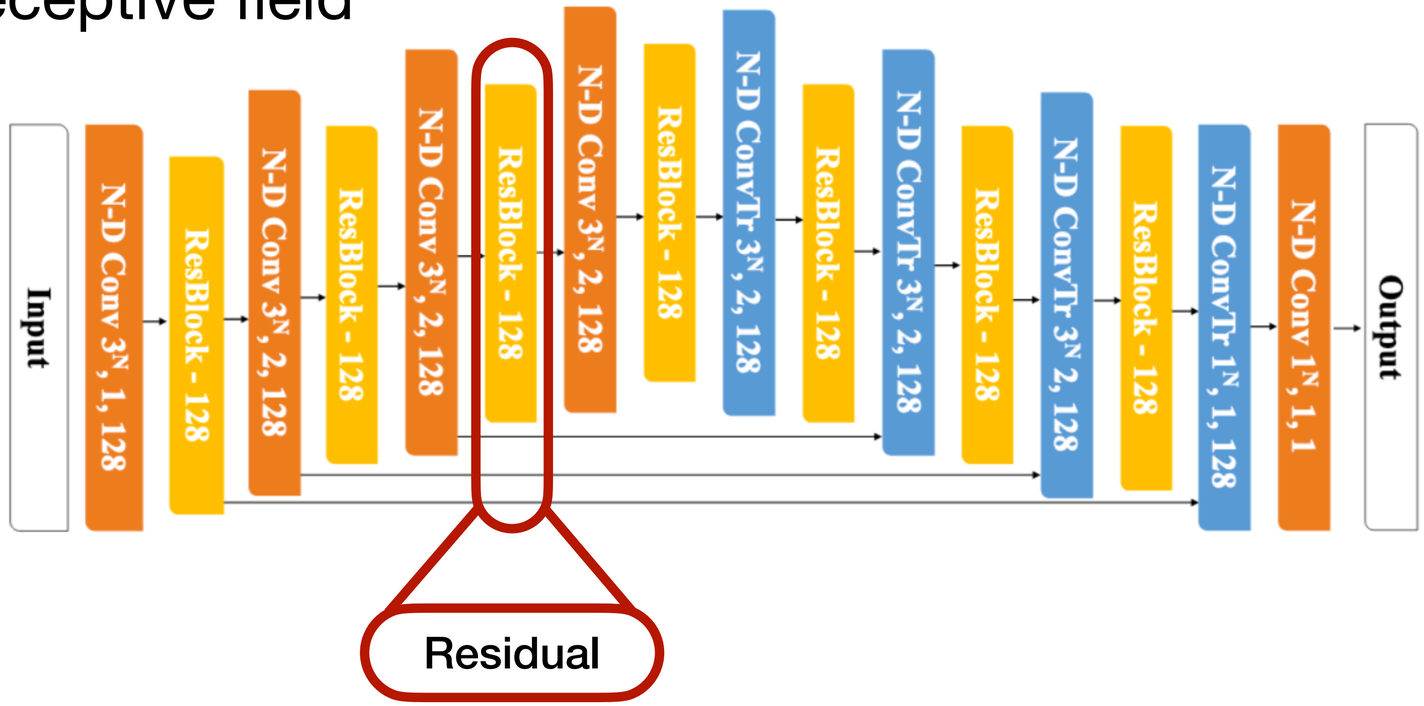


C. Choy, J. Lee, R. Ranftl, J. Park, V. Koltun. “High-dimensional Convolutional Networks for Geometric Pattern Recognition”, CVPR 2020

High-Dimensional Convolutional Networks

Leveraging insights from CNNs

- U-Net-style network
- Expanding receptive field



C. Choy, J. Lee, R. Ranftl, J. Park, V. Koltun. “High-dimensional Convolutional Networks for Geometric Pattern Recognition”, CVPR 2020

High-Dimensional Convolutional Networks

Leveraging insights from CNNs

- U-Net-style network
- Expanding receptive field

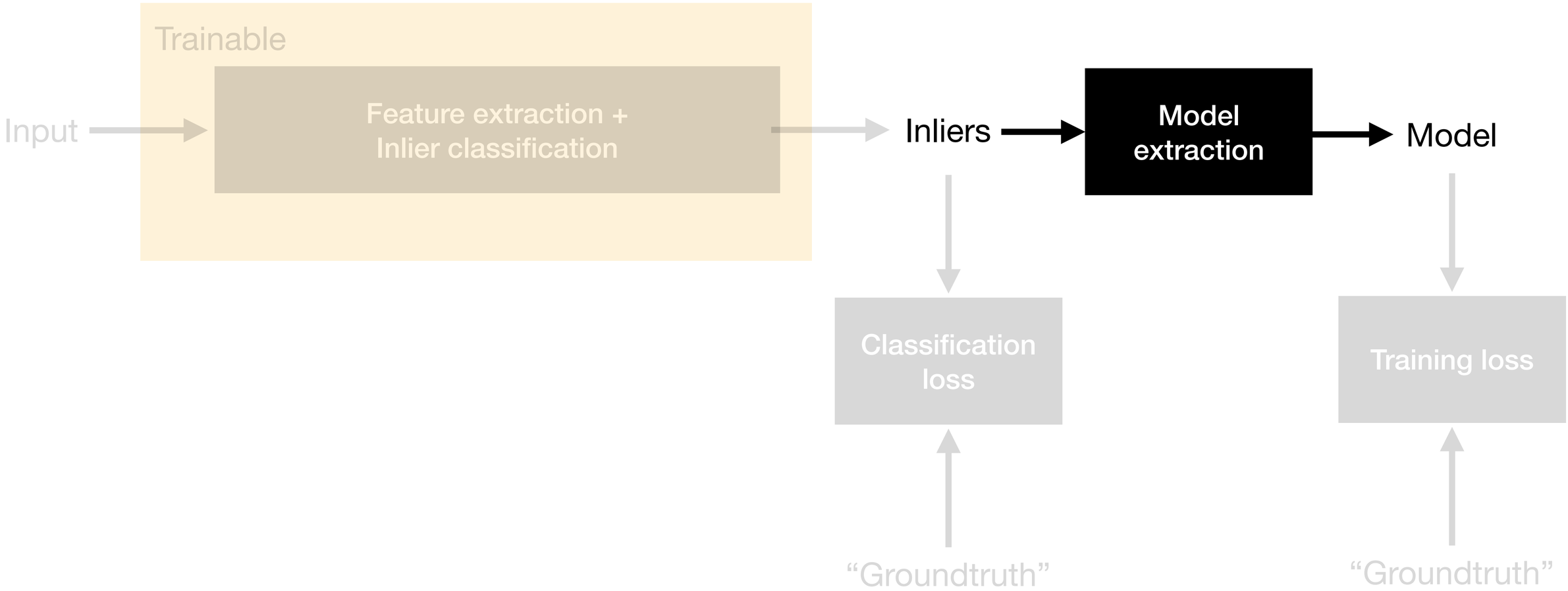
C. Choy, J. Lee, R. Ranftl, J. Park, V. Koltun. “High-dimensional Convolutional Networks for Geometric Pattern Recognition”, CVPR 2020

Feature extractors

An intermediate conclusion

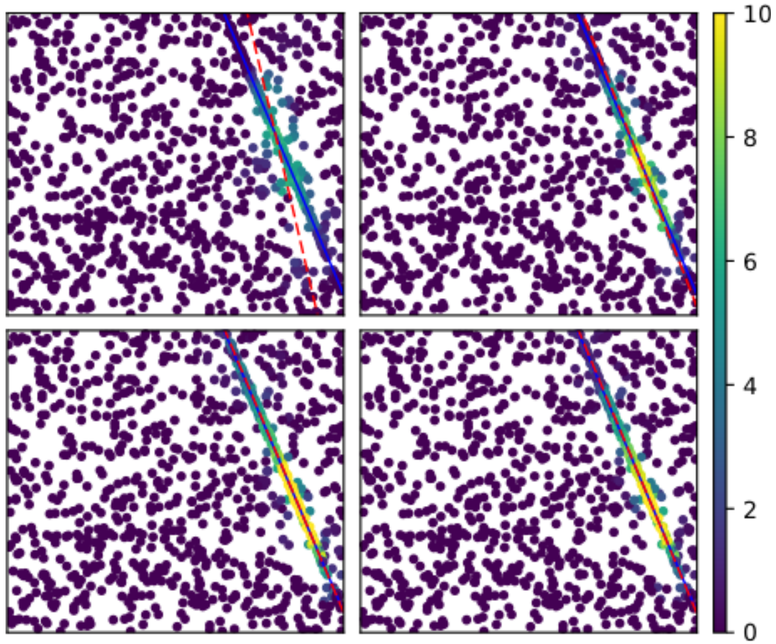
- Data is different from other applications
  - Unordered data
  - High-dimensional
  - Non-metric relations
- Many techniques from “standard” DL carry over
  - Multi-scale processing
  - Normalizing batch statistics
  - Residual connection
  - Attention

Part II: Integrating Geometry



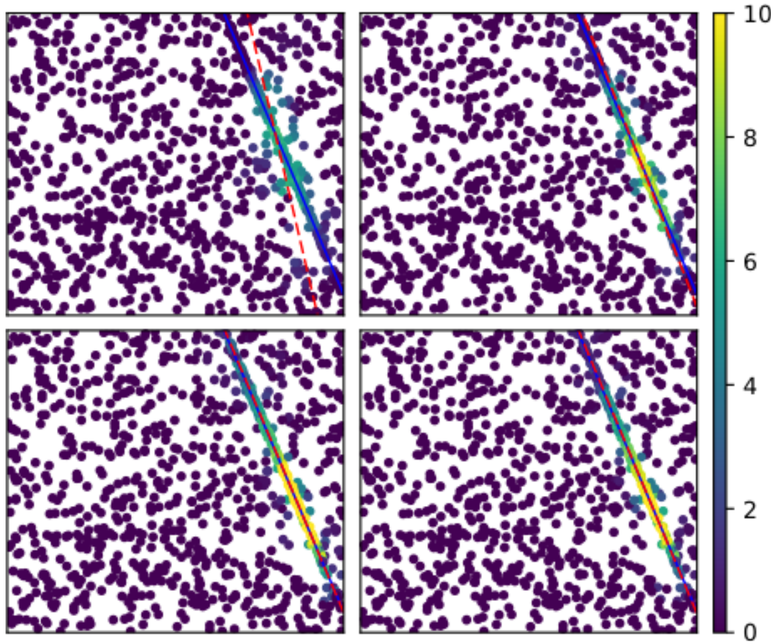
Domain Knowledge

- Feature extractors are generally applicable
  - **Example I:** Append a regression layer that directly outputs model parameters (usually doesn't work well)
  - **Example II:** Append a layer that outputs inlier probabilities for each point
- Domain knowledge can help
  - Faster training
  - Less data
  - Better accuracy



Domain Knowledge

- Good news 🙂
  - Problems can be defined as **optimization problems**
  - Optimization can be integrated in deep networks
    - Closed-form solutions
    - Implicit differentiation
    - Unrolled optimization
  - Least-squares as a workhorse



Least Squares

Basic setup

- Outlier-free problems can be naturally formulated as Least-Squares
  - Standard least-squares

$$\mathbf{x} = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} \|(\mathbf{A})_i \cdot \mathbf{x} - \mathbf{b}_i\|^2$$

  - Homogenous least-squares (for DLT-type problems)

$$\mathbf{x} = \arg\min_{\mathbf{x}:\, \|\mathbf{x}\|=1} \sum_{i=1}^{N} \|(\mathbf{A})_i \cdot \mathbf{x}\|^2$$

Least Squares

Examples

- Outlier-free problems can be naturally formulated as Least-Squares

  - Homogenous least-squares (for DLT-type problems)

Hyperplanes: $(\mathbf{A})_i = \mathbf{p}_i$

Essential/Fundamental Matrix: $(\mathbf{A})_i = \mathrm{vec}(\mathbf{p}_i(\mathbf{p}'_i)^\top)$

Homography: $(\mathbf{A})_{2i-1:2i} = \begin{pmatrix} -\mathbf{p}_i^\top & \mathbf{0} & (\mathbf{p}'_i)_1\mathbf{p}_i^\top \\ \mathbf{0} & -\mathbf{p}_i^\top & (\mathbf{p}'_i)_2\mathbf{p}_i^\top \end{pmatrix}$

Solving Least Squares

Is differentiable (almost everywhere) for overdetermined systems

- Least Squares problems solved using Singular Value Decomposition (SVD)
- SVD is differentiable (as long as there are no repeated singular values)

Proposition 2 *Assume that the singular vectors are ordered according to the magnitude of their singular values* $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{d'})$ *such that* $\mathbf{v}_{d'}$ *is the singular vector corresponding to the smallest singular value. We need to backpropagate through* $g(f(\mathbf{X}))$ *with* $f(\mathbf{X}) = \mathbf{v}_{d'}$*. The gradient of* g *with respect to the input* $\mathbf{X}$ *is given by*

$$\frac{\partial g}{\partial \mathbf{X}} = \mathbf{U}\Big\{2\mathbf{\Sigma}\Big(\mathbf{K}^\top \circ \Big(\mathbf{V}^\top \frac{\partial g}{\partial \mathbf{v}_{d'}}\Big)_{sym}\Big)\Big\}\mathbf{V}^\top, \tag{8}$$

where

$$\mathbf{K}_{ij} = \begin{cases} \frac{1}{\sigma_i^2 - \sigma_j^2}, & \text{if } i \neq j \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

and σ_i *denotes the* i*-th singular value.*

- Already implemented in PyTorch and Tensorflow

```
torch.svd(input, some=True, compute_uv=True, out=None) -> (Tensor, Tensor, Tensor)
```

This function returns a namedtuple `(U, S, V)` which is the singular value decomposition of a input real matrix or batches of real matrices `input` such that $input = U \times diag(S) \times V^T$.

```
tf.linalg.svd(
    tensor, full_matrices=False, compute_uv=True, name=None
)
```

Iteratively Reweighted Least Squares

A classic tool in robust statistics

- Consider the problem

$$\arg\min_{\mathbf{x}} \sum_{i=1}^{N} \|(\mathbf{A})_i \cdot \mathbf{x} - \mathbf{b}_i\|_p$$

- IRLS algorithm:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} w_i(\mathbf{x}^t) \|(\mathbf{A})_i \cdot \mathbf{x} - \mathbf{b}_i\|^2$$

$$w_i(\mathbf{x}^t) = \|(\mathbf{A})_i \cdot \mathbf{x}^t - \mathbf{b}_i\|^{(p-2)}$$

- Also applicable for robust penalties

Iteratively Reweighted Least Squares

A classic tool in robust statistics

- Consider the problem

$$\arg\min_{\mathbf{x}} \sum_{i=1}^{N} \|(\mathbf{A})_i \cdot \mathbf{x} - \mathbf{b}_i\|_p$$

- IRLS algorithm:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} w_i(\mathbf{x}^t) \|(\mathbf{A})_i \cdot \mathbf{x} - \mathbf{b}_i\|^2$$

$$w_i(\mathbf{x}^t) = \|(\mathbf{A})_i \cdot \mathbf{x}^t - \mathbf{b}_i\|^{(p-2)}$$

- Also applicable for robust penalties

Can still be solved using SVD

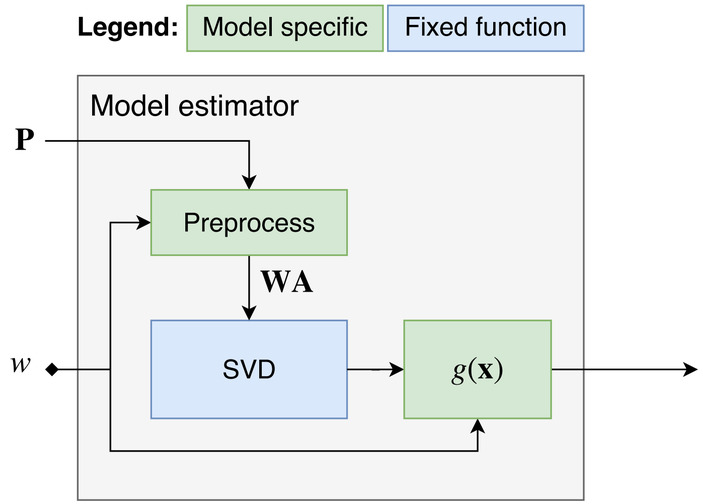
Reweighted Least Squares

- Standard least squares

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} w_i(\mathbf{x}^t) \|(\mathbf{A})_i \cdot \mathbf{x} - \mathbf{b}_i\|^2$$

- Homogenous least squares

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}:\, \|\mathbf{x}\|=1} \sum_{i=1}^{N} w_i(\mathbf{x}^t) \|(\mathbf{A})_i \cdot \mathbf{x}\|^2$$



Reweighted Least Squares

- Standard least squares

Can be represented by neural network

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} w_i(\mathbf{x}^t) \|(\mathbf{A})_i \cdot \mathbf{x} - \mathbf{b}_i\|^2$$

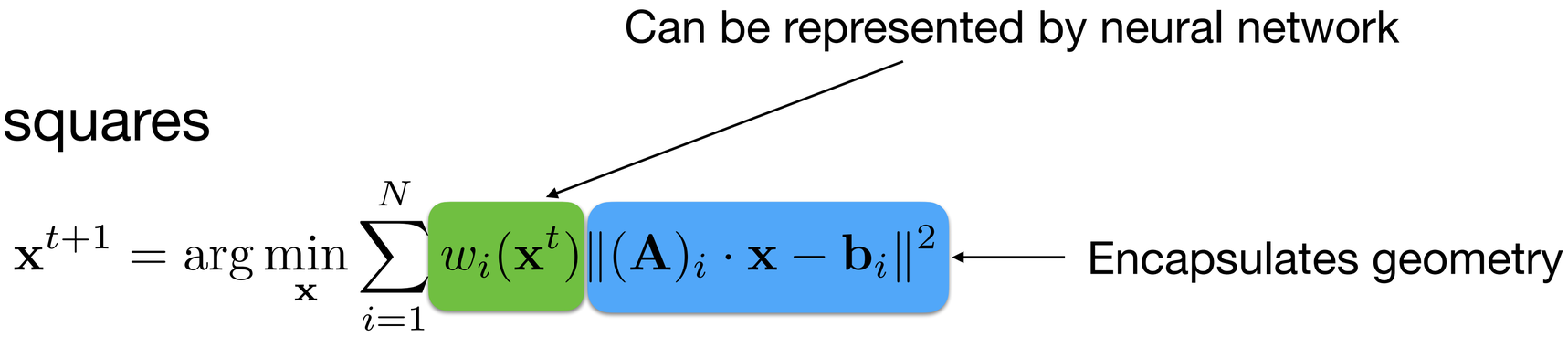
- Homogenous least squares

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}:\, \|\mathbf{x}\|=1} \sum_{i=1}^{N} w_i(\mathbf{x}^t) \|(\mathbf{A})_i \cdot \mathbf{x}\|^2$$

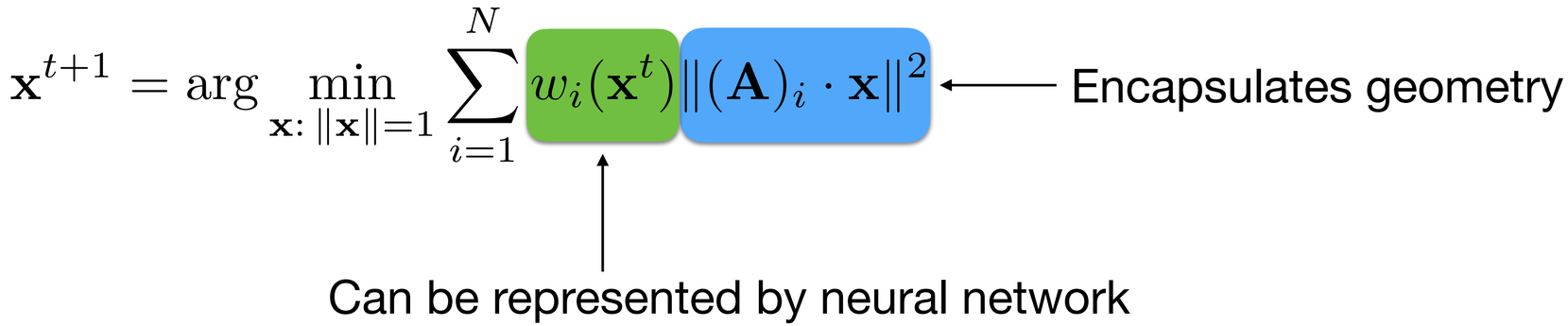
Can be represented by neural network

Reweighted Least Squares

- Standard least squares



- Homogenous least squares



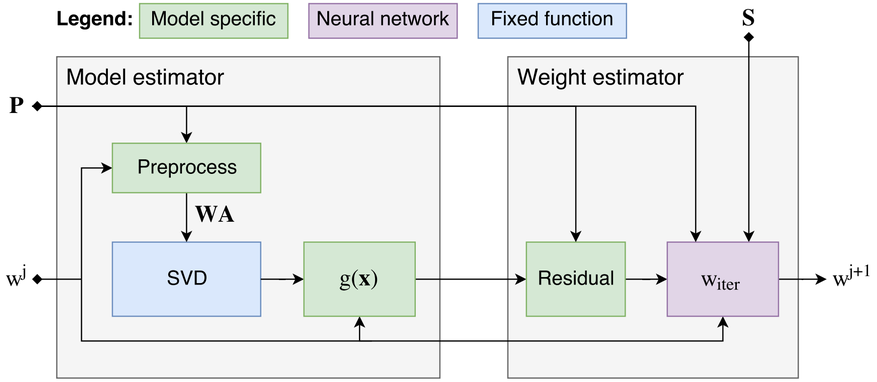
Neural Reweighted Least-Squares

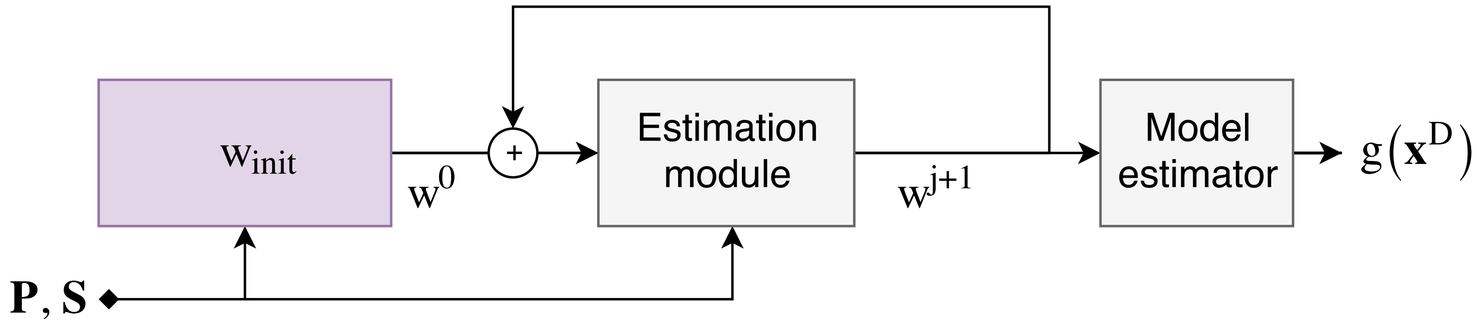
Examples

- Define a regression loss [Yi 2017]

$$\mathcal{L} = \min\{\|E_{gt} \pm g(\mathbf{P}, \mathbf{w})\|\} \qquad \mathcal{L} = \min\{\|F_{gt} \pm g(\mathbf{P}, \mathbf{w})\|\}$$

- Iterate weight estimation and model estimation [Ranftl 2018]





Neural Reweighted Least-Squares

Examples

- Define a regression loss [Yi 2017]

$$\mathcal{L} = \min\{\|E_{gt} \pm g(\mathbf{P}, \mathbf{w})\|\}$$

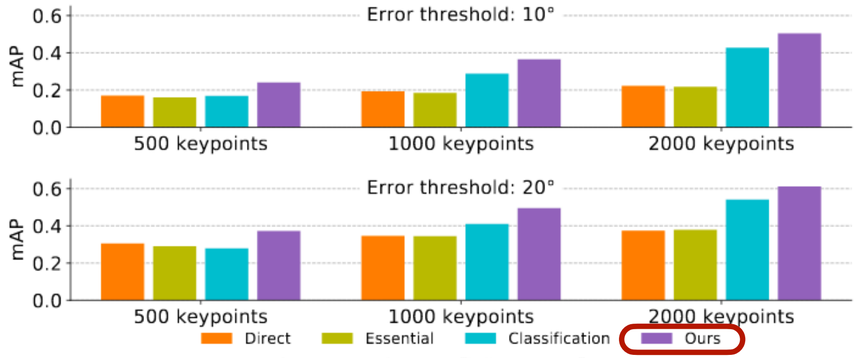


Image from [Yi 2017]

- Iterate weight estimation and model estimation [Ranftl 2018]

PointCN	UnA	UnB	OF	L3	Geo	Iter	Known	Unknown
✓							34.36/13.93	47.98/23.55
✓	✓						34.38/14.04	47.93/24.10
✓		✓					36.33/17.88	49.65/28.78
✓		✓	✓				40.78/25.94	51.63/32.55
✓		✓	✓	✓			39.69/26.04	50.70/30.48
✓		✓	✓		✓		40.79/28.39	51.10/33.68
✓		✓	✓		✓	✓	**42.46**/33.06	**52.18**/39.33

Table from [Zhang 2019]

	% Inliers	F-score	Mean	Median	Min	Max	Time [ms]
$D = 1$	42.30	44.80	3.45	1.00	0.08	1912.67	7
$D = 3$	44.91	47.25	1.98	0.82	0.08	566.70	18
$D = 5$	45.02	46.99	2.04	0.83	0.11	285.36	26
$D = 5^*$	44.60	46.42	2.23	0.84	0.10	391.64	26
Direct reg.	4.42	9.14	16.67	11.96	0.83	386.15	3

Table from [Ranftl 2018]

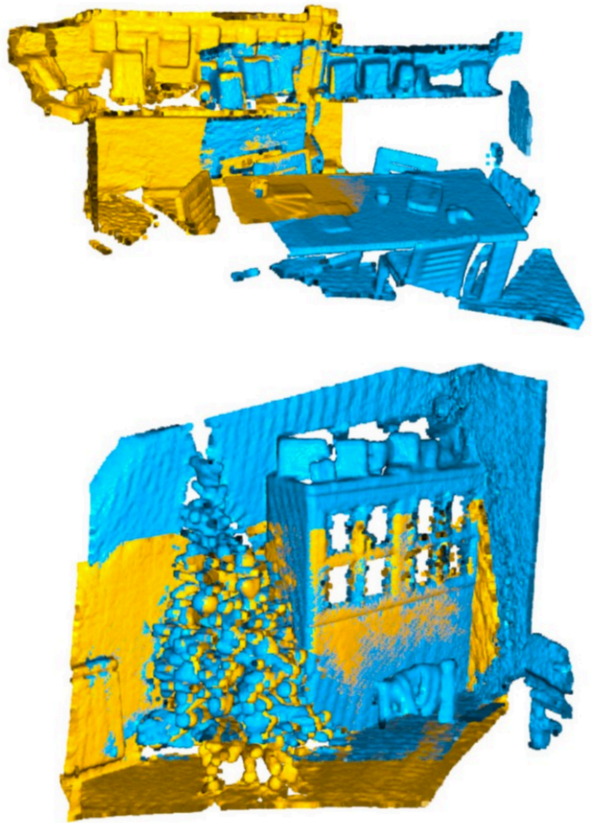
Neural Reweighted Least-Squares

Examples

- Weighted procrustes analysis
- Estimate rotation and translation between 3D pointclouds

$$\min \mathrm{Tr}((Y - RX - \mathbf{t}\mathbf{1}^\top) W (Y - RX - \mathbf{t}\mathbf{1}^\top)^\top$$

Learned weights



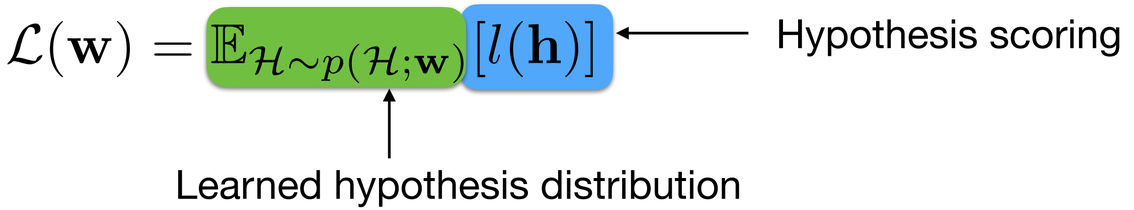
- Network provides initialization for iterative refinement

C. Choy, W. Dong, V. Koltun. “Deep Global Registration”, CVPR 2020

RANSAC in the training loop

Neural Guidance

- Learn a hypothesis sampling distribution
  - Select outlier-free hypothesis sets with high probability
  - Parametrized by network



E. Brachmann and C. Rother. “Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses”, ICCV 2019

RANSAC in the training loop

Neural Guidance

- Learn a hypothesis sampling distribution
  - Select outlier-free hypothesis sets with high probability
  - Parametrized by network

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathcal{H}\sim p(\mathcal{H};\mathbf{w})}[l(\mathbf{h})]$$

Only backprop through distribution

  - Hypothesis selection (and potentially scoring non-differentiable)
    - Stochastic estimate of the gradient $\frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}) = \mathbb{E}[l(\mathbf{h})\frac{\partial}{\partial \mathbf{w}}\log p(\mathcal{H};\mathbf{w})]$
    - Variance reduction

E. Brachmann and C. Rother. “Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses”, ICCV 2019

RANSAC in the training loop

Neural Guidance

- In the spirit of guided RANSAC variants
  - PROSAC
  - USAC
- Can be trained self-supervised
- Very flexible
- Can be combined with DSAC to also learn observations

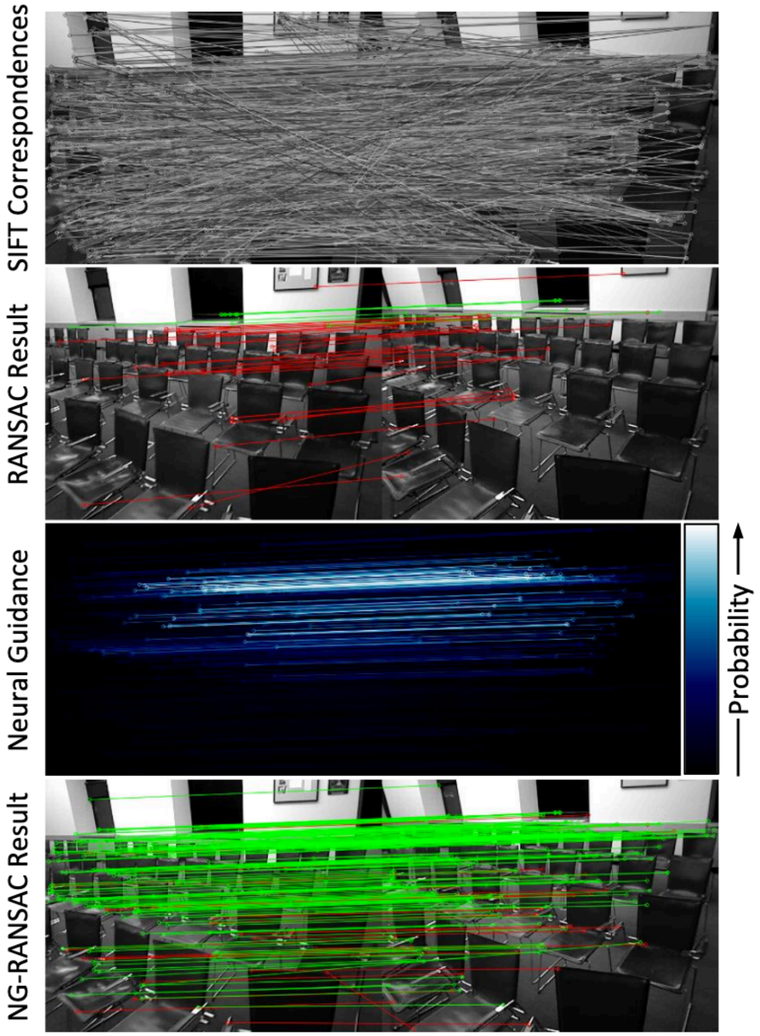
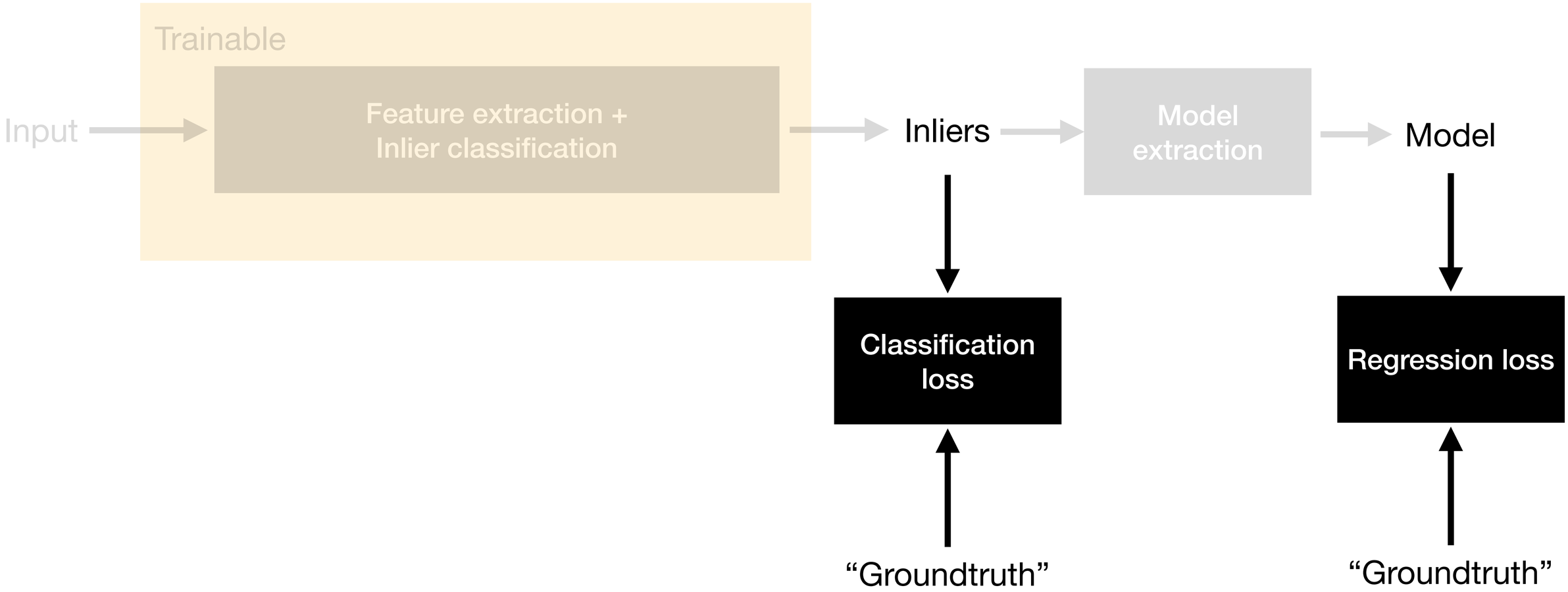


Image from [Brachmann 2019]

E. Brachmann and C. Rother. “Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses”, ICCV 2019

E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. “DSAC - Differentiable RANSAC for Camera Localization”, CVPR 2017

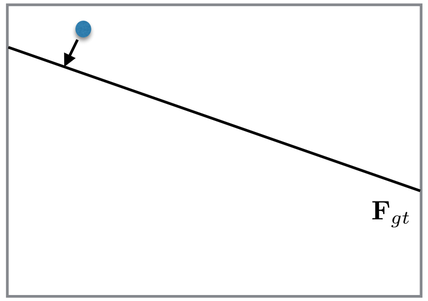
Part III: Loss functions



Loss functions

General tips

- Algebraic usually problematic
  - Errors arbitrarily scaled
- Use geometrically meaningful metrics
  - Sampson error, epipolar distance
  - Reprojection error
  - Point-to-point distance, point-to-plane distance
- Generally needs robust loss formulations
  - Certain points can have large influence
  - Clipping/robust loss formulations



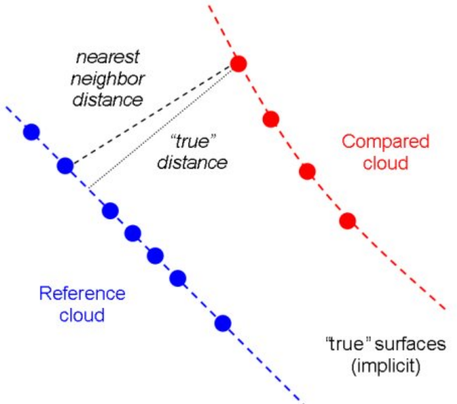


Image from https://www.cloudcompare.org/

Loss functions

An example

- From F_{gt} can derive a set of ground truth inliers p_j

  ➔ Can always cast the problem as a classification problem!

- F_{gt} is defined up to scale
  - Reasonable to have

$$\mathcal{L}(F) = \mathcal{L}(aF)$$

  - Naive loss $\mathcal{L}(F) = \|F - F_{gt}\|$ doesn't have this property
  - Algebraic error can be scaled arbitrarily

$$\mathcal{L}(F) = \sum_j \|p_j^\top F p_j\|^2$$

Loss functions

An example

- From F_{gt} can derive a set of ground truth inliers p_j

➡ Can always cast the problem as a classification problem!

- F_{gt} is defined up to scale
  - Reasonable to have

$$\mathcal{L}(F) = \mathcal{L}(aF)$$

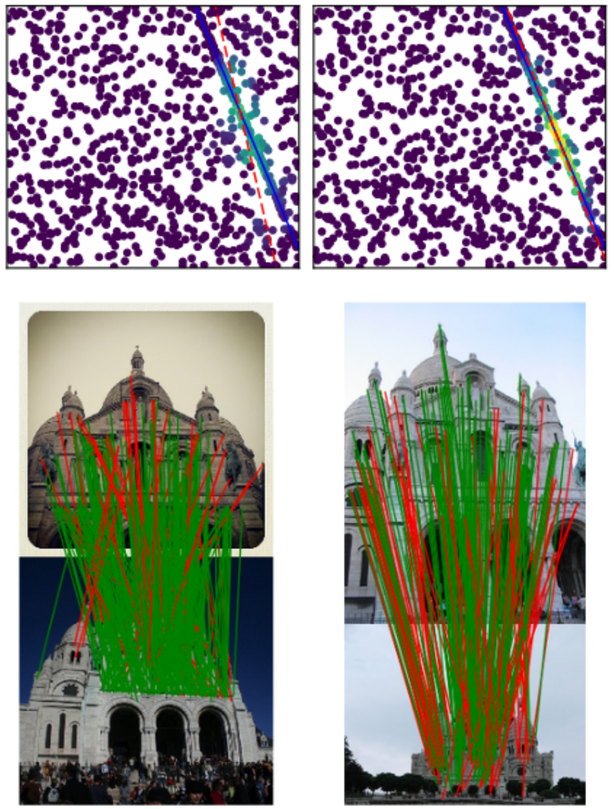
  - Naive loss $\mathcal{L}(F) = \|F - F_{gt}\|$ doesn't have this property
  - Algebraic error can be scaled arbitrarily

$$\mathcal{L}(F) = \sum_j \|p_j^\top F p_j\|^2$$

Loss functions

Design considerations

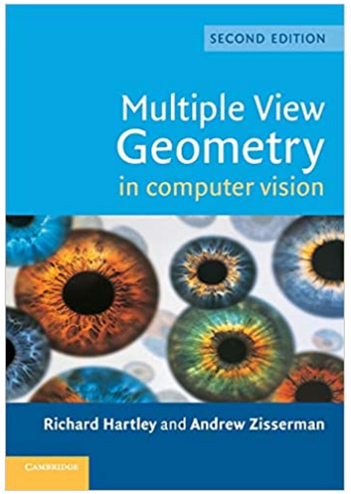
- Problem has a dual classification/regression character
  - Classify model inliers
    - Requires definition of inlier
    - Simply use binary cross-entropy (might require re-balancing if many more outliers than inliers)

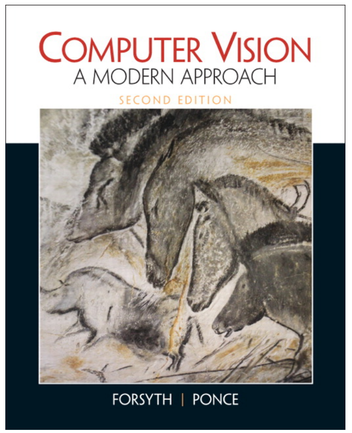


Loss functions

Design considerations

- Problem has a dual classification/regression character
  - Classify model inliers
    - Requires definition of inlier
    - Simply use binary cross-entropy (might require re-balancing if many more outliers than inliers)
  - Regress a model
    - Requires geometry-aware layers
    - Enforce constraints (rank, orthogonality)
    - Parametrization can be an issue





Loss functions

An example for epipolar geometry

Find compatible points

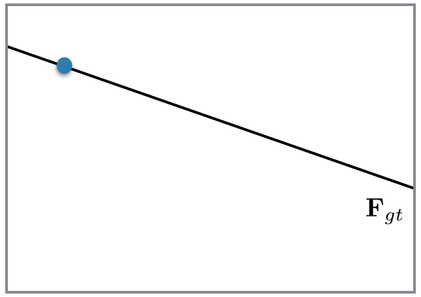


Image 1

Option A: Reproject

$$\min_{\mathbf{p}} d(\hat{\mathbf{p}}, \mathbf{p}) + d(\hat{\mathbf{p}}', \mathbf{p}')$$

subject to $\mathbf{p}^\top \mathbf{F} \mathbf{p}' = 0$

Option B: Threshold

$$d(\mathbf{p}, \mathbf{F}_{gt}) < t$$

Measure distance of estimate to these points

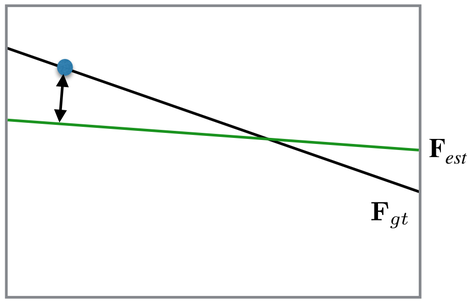
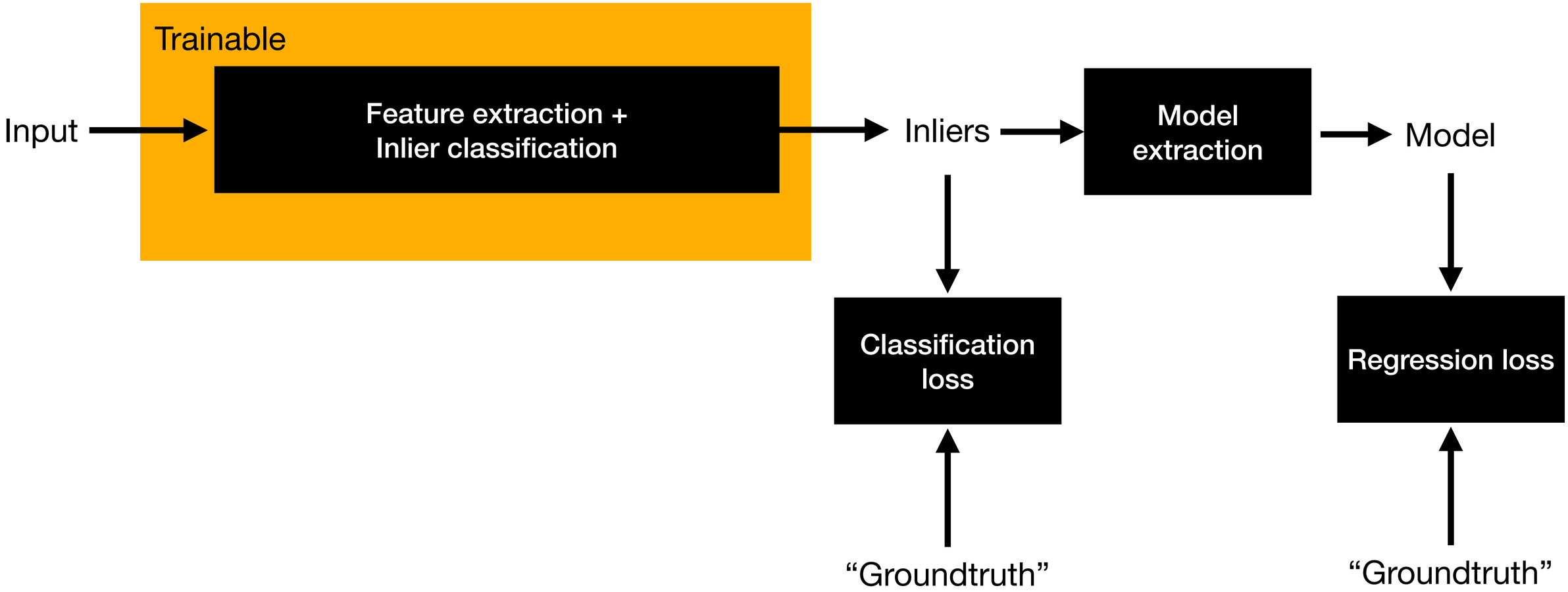


Image 1

R. Ranftl, V. Koltun. “Deep Fundamental Matrix Estimation”, ECCV 2018

J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan and H. Liao. "Learning Two-View Correspondences and Geometry Using Order-Aware Network”, ICCV 2019

Part IV: Practical Considerations



Training data

There is never enough

- Perfect training data is hard to get
  - No manual labeling possible
  - No measurement device exists
- Need sufficient size
- Need sufficient diversity
- Very application dependent
- **Your method is only as good as your training data**

Datasets

Two-view geometry

- Use SfM reconstructions as proxy-groundtruth

	Type	Size	Source
YFCC100M	Phototourism	72 scenes	SfM
SUN3D	Indoor	~250 scenes	RGB-D
GL3D-v2	Drone footage + Small objects	540 scenes	SfM
Megadepth	PhotoTourism	196 scenes	SfM
Tanks and Temples	Single Camera	30 sequences	SfM
KITTI	Road Scenes	22 sequences	LiDaR

- Possible to use synthetic data (but mostly unexplored)

Datasets

Two-view geometry

- Use SfM reconstructions as proxy-groundtruth

	Type	Size	Source
YFCC100M	Phototourism	72 scenes	SfM
SUN3D	Indoor	~250 scenes	RGB-D
GL3D-v2	Drone footage + Small objects	540 scenes	SfM
Megadepth	PhotoTourism	196 scenes	SfM
Tanks and Temples	Single Camera	30 sequences	SfM
KITTI	Road Scenes	22 sequences	LiDaR

- Possible to use synthetic data (but mostly unexplored)

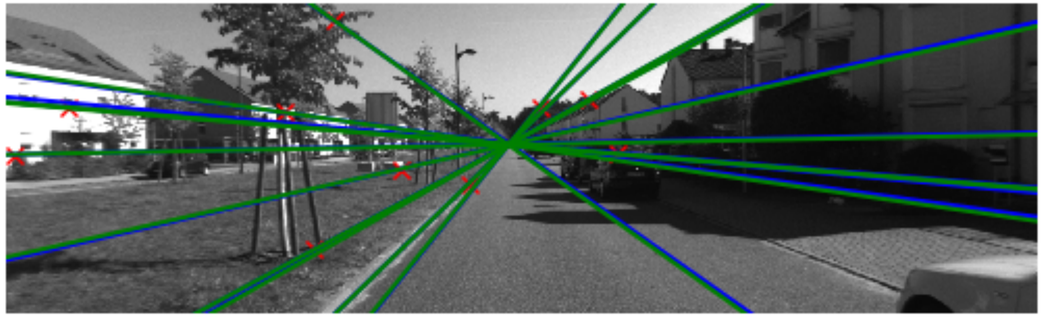
Datasets

Two-view geometry

- Diversity (mostly) dependent on capture mode

Unconstrained

Handheld video



Constrained video

Datasets

3D Registration

	Type	Size	Source
3D Match	Indoor	62 scenes	RGB-D
KITTI	Outdoor	~70 sequences	LiDaR
ICL-NIUM	Indoor	4 scenes	Synthetic
S3-DIS	Indoor	6 areas	RGB-D
ModelNet40	CAD	40 models	Synthetic
ApolloScapes	Outdoor	80k point clouds	LiDAR

Datasets

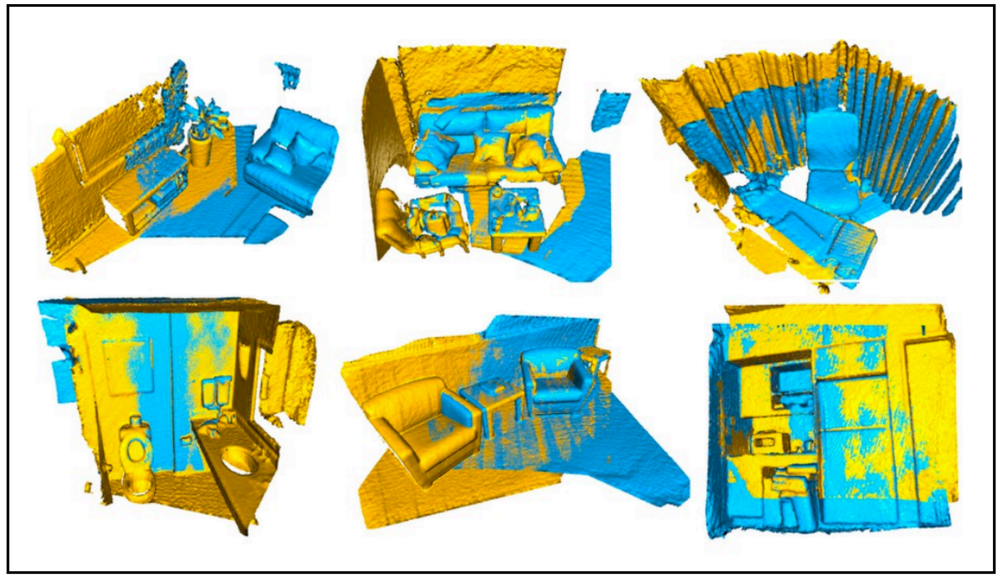
3D Registration

	Type	Size	Source
3D Match	Indoor	62 scenes	RGB-D
KITTI	Outdoor	~70 sequences	LiDaR
ICL-NIUM	Indoor	4 scenes	Synthetic
S3-DIS	Indoor	6 areas	RGB-D
ModelNet40	CAD	40 models	Synthetic
ApolloScapes	Outdoor	80k point clouds	LiDAR

Datasets

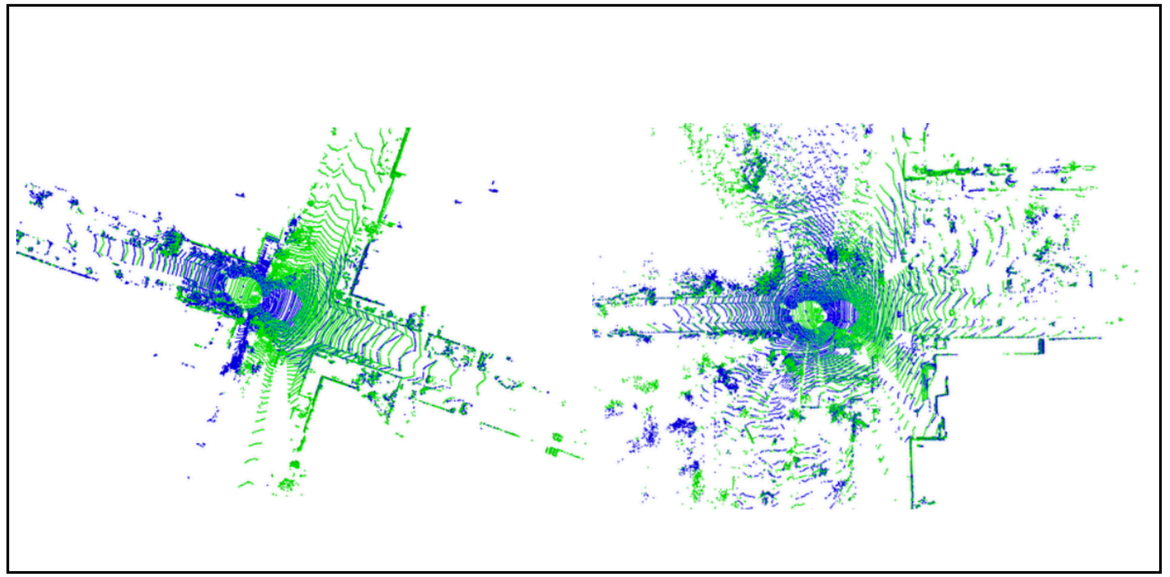
3D Point Clouds

- Strong dependence on acquisition sensor!



RGB-D

vs.



LiDaR

Images from [Choy 2020b]

Datasets

Other applications

	Problem	Type
[Dang 2018]	Absolute pose	Synthetic
[Kendall 2015]	PnP	SfM
[Ranftl 2018]	Homography	Synthetic
[Choy 2020a]	Linear subspaces	Synthetic
[Sun 2020]	Line	Synthetic
[Workman 2019]	Horizon Line	SfM + Manual cleaning

- For PnP transfer to real data demonstrated [Dang 2018]
- Acquiring data is both important and non-trivial!

Numerical Stability

- Training can be unstable
  - SVD non-differentiable if repeated singular values
  - Eigen-vector switching
  - Large contributions from gross outliers in the beginning
- You are likely to see NaNs

with $f(\mathbf{X}) = \mathbf{v}_{d'}$. *The gradient of* g *with respect to the input* $\mathbf{X}$ *is given by*

$$\frac{\partial g}{\partial \mathbf{X}} = \mathbf{U}\Big\{2\mathbf{\Sigma}\Big(\mathbf{K}^\top \circ \Big(\mathbf{V}^\top \frac{\partial g}{\partial \mathbf{v}_{d'}}\Big)_{sym}\Big)\Big\}\mathbf{V}^\top,$$

where

$$\mathbf{K}_{ij} = \begin{cases} \frac{1}{\sigma_i^2 - \sigma_j^2}, & \text{if } i \neq j \\ 0, & \text{otherwise} \end{cases}$$

and σ_i *denotes the* i*-th singular value.*

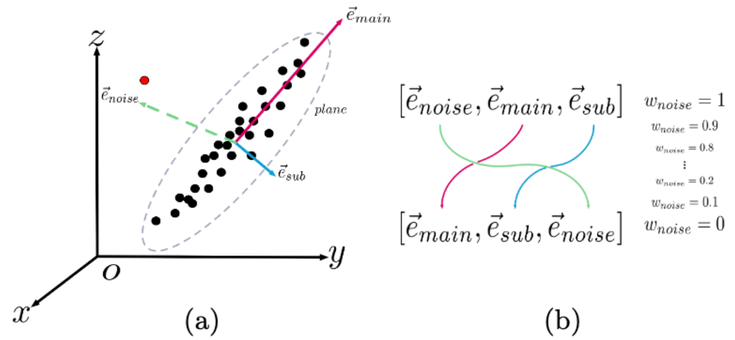


Image from [Dang 2018]

Numerical Stability

Ways to mitigate

- Eigendecomposition-free training [Dang 2018]

$$\mathcal{L}(\mathbf{W}) = \tilde{\mathbf{e}}\mathbf{X}^\top \mathbf{W}\mathbf{X}\tilde{\mathbf{e}} + \alpha \exp(-\beta \mathrm{tr}(\hat{\mathbf{X}}^\top \mathbf{W}\hat{\mathbf{X}}))$$

Weight from network Groundtruth EV Regularization

with $f(\mathbf{X}) = \mathbf{v}_{d'}$. The gradient of g with respect to the input $\mathbf{X}$ is given by

$$\frac{\partial g}{\partial \mathbf{X}} = \mathbf{U}\Big\{2\mathbf{\Sigma}\Big(\mathbf{K}^\top \circ \Big(\mathbf{V}^\top \frac{\partial g}{\partial \mathbf{v}_{d'}}\Big)_{sym}\Big)\Big\}\mathbf{V}^\top,$$

where

$$\mathbf{K}_{ij} = \begin{cases} \frac{1}{\sigma_i^2 - \sigma_j^2}, & \text{if } i \neq j \\ 0, & \text{otherwise} \end{cases}$$

and σ_i denotes the i-th singular value.

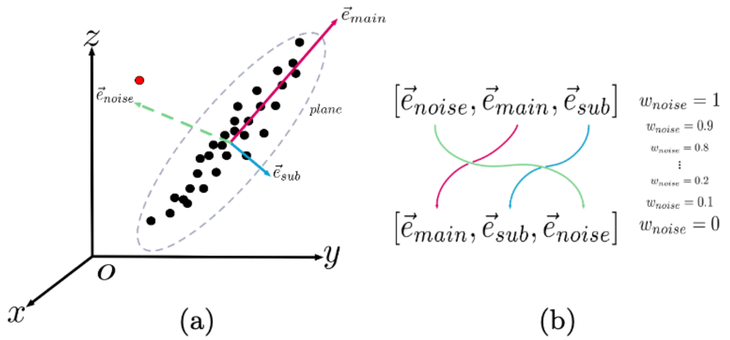


Image from [Dang 2018]

Numerical Stability

Ways to mitigate

- Eigendecomposition-free training [Dang 2018]

$$\mathcal{L}(\mathbf{W}) = \tilde{\mathbf{e}}\mathbf{X}^\top \mathbf{W}\mathbf{X}\tilde{\mathbf{e}} + \alpha \exp(-\beta \mathrm{tr}(\hat{\mathbf{X}}^\top \mathbf{W}\hat{\mathbf{X}}))$$

with $f(\mathbf{X}) = \mathbf{v}_{d'}$. The gradient of g with respect to the input $\mathbf{X}$ is given by

$$\frac{\partial g}{\partial \mathbf{X}} = \mathbf{U}\Big\{2\boldsymbol{\Sigma}\Big(\mathbf{K}^\top \circ \Big(\mathbf{V}^\top \frac{\partial g}{\partial \mathbf{v}_{d'}}\Big)_{sym}\Big)\Big\}\mathbf{V}^\top,$$

where

$$\mathbf{K}_{ij} = \begin{cases} \frac{1}{\sigma_i^2 - \sigma_j^2}, & \text{if } i \neq j \\ 0, & \text{otherwise} \end{cases}$$

and σ_i denotes the i-th singular value.

- Alternative:
  - Careful normalization
  - Clip loss function
  - Code safeguard

```
do_step = True
for param in model.parameters():
    if param.grad is not None:
        if (1 - torch.isfinite(param.grad).long()).sum() > 0:
            print("NaN in gradient")
            do_step = False
            break

if do_step:
    optimizer.step()
```

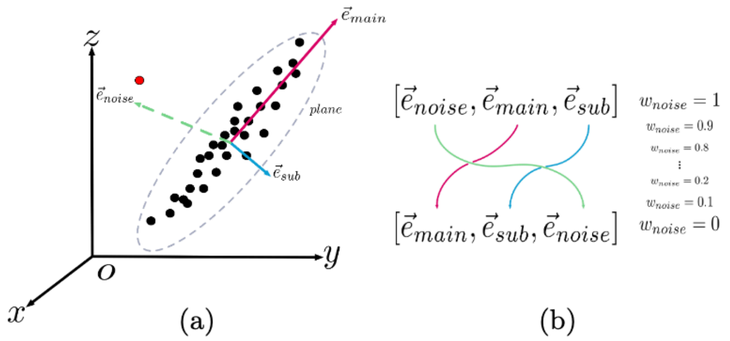
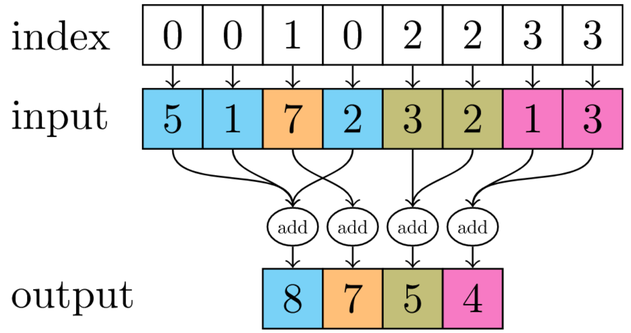


Image from [Dang 2018]

Mini-Batching

- Mini-batching stabilizes and speeds up learning
- Different number of points per sample
- Makes effective batching hard
- Two possible solutions:
  1. Resample to a fixed size
  2. Retain sample indices and use them for aggregation ops



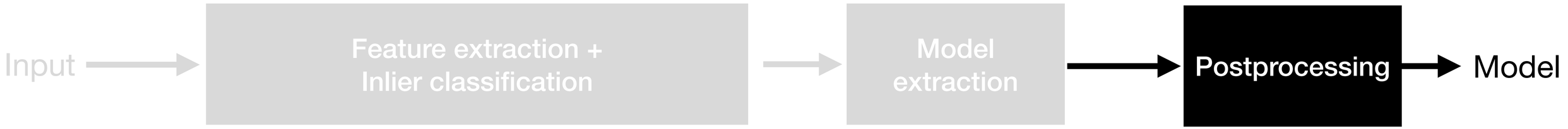
PyTorch geometric

class **Batch**(*batch=None, **kwargs*) [source]

A plain old python object modeling a batch of graphs as one big (dicconnected) graph. With `torch_geometric.data.Data` being the base class, all its methods can also be used here. In addition, single graphs can be reconstructed via the assignment vector `batch`, which maps each node to its respective graph identifier.

https://github.com/rusty1s/pytorch_geometric

Part V: Postprocessing



Postprocessing

Everything is allowed at test time

- Simple integration
  - Remove samples with small weights
  - Send surviving points to *SAC
- Other approaches possible
  - Evaluate different percentiles of points
- Classic and learned are complementary

	Method	Outdoors	Indoors
Traditional	LMedS	.296/.383	.142/.235
	RANSAC	.356/.437	.172/.272
	MLESAC	.148/.216	.135/.230
	DegenSAC	.328/.394	.191/.291
	MAGSAC	.385/.457	.185/.282
Learned	CNe (weighted-8pt)	.323/.469	.189/.331
	CNe+RANSAC	.449/.554	.201/.315
	CNe+MAGSAC	.500/.598	.213/.326
	DFE (weighted-8pt)[3]	.319/.470	.167/.294
	DFE+RANSAC	.414/.508	.193/.303
	DFE+MAGSAC	.452/.541	.211/.320
	OANet (weighted-8pt)	.439/.581	.256/.392
	OANet+RANSAC	.482/.592	.211/.331
	OANet+MAGSAC	.514/.615	.230/.346
Ours	ACNe (weighted-8pt)	.501/**.638**	**.278/.428**
	ACNe+RANSAC	.478/.590	.209/.329
	ACNe+MAGSAC	**.518**/.621	.226/.343

Table from [Sun 2020]

	Outdoor(%)		Indoor(%)	
	Known	Unknown	Known	Unknown
RANSAC	5.82/-	9.08/-	4.38/-	2.86/-
PointCN[21]	34.36/13.93	47.98/23.55	20.44/11.28	15.98/9.36
PointNet++[27]	34.15/9.28	46.23/14.04	20.28/7.15	15.61/5.59
N^3Net[26]	34.18/12.49	49.13/23.18	20.31/7.95	15.38/7.13
DFE[30]	36.87/18.40	49.45/29.70	20.97/14.09	16.45/12.45
Ours	40.78/25.94	51.63/32.55	21.82/16.09	16.51/12.54
Ours++	**42.46**/33.06	**52.18**/39.33	**22.50**/21.44	**17.50**/16.39
RANSAC*	15.21/-	21.95/-	18.17/-	14.58/-
PointCN*[21]	30.48/13.82	43.18/24.83	23.66/12.04	18.52/10.21
Ours*	**33.42**/23.85	**46.28**/32.18	**24.31**/14.81	**19.04**/12.12

Table from [Zhang 2019]

Conclusion

- A novel and exciting research field
- Novel type of data in the context of deep learning
  - Feature extractors are likely not optimal
- Can integrate geometric knowledge
  - Need to keep differentiability in mind
- Opportunity to improve with more and better data
- Some questions are (and will be hard to answer)
  - Break-down points
  - Dependence on dimensionality of data

Open questions

- Many avenues to make progress
  - Feature extractors
  - Tighter integration of non-differentiable parts
  - Complete end-to-end pipelines
  - Theoretical questions
  - Loss functions
  - Differentiable minimal solvers
  - Handling degeneracy
  - Datasets
  - New applications

Shall we only focus on deep learning?

Check out the next talk by Dmytro Mishkin!

There might be surprises :)

Questions?

References

[Qi 2016] C. Qi, H. Su, K. Mo, and L. Guibas. “PointNet: Deep learning on point sets for 3D classification and segmentation”, CVPR 2016

[Zaheer 2017] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. “DeepSets”, NeuRIPS 2017

[Yi 2018] K.M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. “Learning to find good correspondences”, CVPR 2018

[Ranftl 2018] R. Ranftl, V. Koltun. “Deep Fundamental Matrix Estimation”, ECCV 2018

[Zhang 2019] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao. "Learning Two-View Correspondences and Geometry Using Order-Aware Network”, ICCV 2019

[Sun 2020] W. Sun, W. Jiang, E. Trulls, A. Tagliasacchi, and K. M. Yi. “ACNe: Attentive Context Normalization For Robust, Permutation-Equivariant Learning”, CVPR 2020

[Brachmann 2019] E. Brachmann and C. Rother, “Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses”, ICCV 2019

[Zhao 2019] C. Zhao, Z. Cao, C. Li, X. Li, and J. Yang, “NM-Net: Mining Reliable Neighbors for Robust Feature Correspondences”, CVPR 2019

[Dang 2018] Z. Dang, K.M. Yi, Y. Hu, F. Wang, P. Fua, and M. Salzmann, “Eigendecomposition-free Training of Deep Networks for Zero Eigenvalue-based Losses”, ECCV 2018

[Choy 2020a] C. Choy., J. Lee, R. Ranftl, J. Park, and V. Koltun. “High-dimensional Convolutional Networks for Geometric Pattern Recognition”, CVPR 2020

[Choy 2020b] C. Choy, W. Dong, and V. Koltun. “Deep Global Registration”, CVPR 2020

[Workman 2016] S. Workman, M. Zhai, and N. Jacobs. "Horizon Lines in the Wild”, BMVC 2016

[Kendall 2015] A. Kendall, M. Grimes, and R. Cipolla. “PoseNet: A convolutional network for real-time 6-DoF camera relocalization”, ICCV 2015