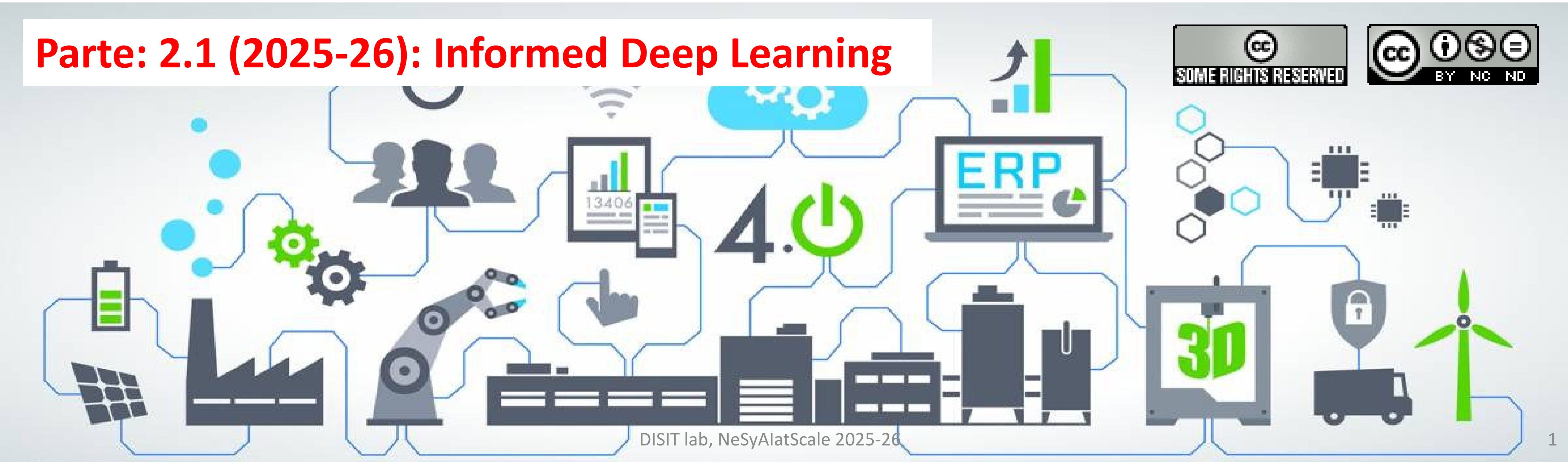
https://www.snap4City.org

NeuroSymbolic Artificial Intelligence at Scale

Marco Fanfani, marco.fanfani@unifi.it

https://www.disit.org/



Parte: 2.1 (2025-26): Informed Deep Learning

Informed Deep Learning

Moving beyond the "Black Box" paradigm toward physically and logically consistent AI

The Limits of "Pure" Deep Learning

Why standard Deep Learning isn't enough for critical applications:

- **Data Hunger** (High Sample Complexity): Modern neural networks require massive volumes of training data—often millions of frames or samples—to be effective.
- **Lack of Interpretability**: Most architectures are "black boxes" where internal computations do not correspond to human-comprehensible reasoning steps or semantics.
- **Poor Generalization**: Standard models are prone to failure when exposed to data outside their specific training distribution, struggling to reuse expertise in radically different contexts.

Why Domain Knowledge?

- **Reducing Complexity**: By introducing an Inductive Bias, we guide the network to focus on relevant parameters, drastically reducing the data needed to learn.
- **Ensuring Consistency**: Domain knowledge (geometry, logic, or physics) acts as a supervisor that guarantees results are physically plausible and internally coherent.
- **Synergistic Learning**: We leverage the perception strengths of Deep Learning (automatic feature discovery) and the reasoning strengths of symbolic AI (reusability and abstraction)

A Taxonomy of Solutions

How can we inject knowledge into a Deep Learning pipeline?

- **Architectural Constraints**: Designing the network topology itself to enforce properties.

- **Constraints in the Loss Function**: Adding penalty terms (Semantic Loss, Geometric Loss) to steer predictions toward valid states.

Topics

- **Domain-driven Architectures**: How to build "physics-aware" layers, focusing on the case of SincNet for raw waveform processing.
- **Geometric & Photometric Constraints**: Using the laws of optics and epipolar geometry for Self-supervised depth estimation (e.g., MonoDepth, D3VO).
- **Semantic Based Regularization**: Bridging the gap between continuous probability vectors and discrete Boolean Logic.

Domain-driven Architectures

SincNet example

Speaker Recognition – Who is Talking?

- **Problem**: Identifying a person just by their voice.
- **Input**: A digital recording of sound (a "raw waveform").
- **Goal**: A computer needs to "listen" and extract specific biological traits that make your voice unique.
- **Challenge**: Sound is **messy**, **fast-moving**, and **contains a lot of "noise"** that isn't useful for identity the speaker.

M. Ravanelli & Y. Bengio, "Speaker Recognition From Raw Waveform With Sincnet", IEEE STL 2018.

What do we know about Sound?

- Sound isn't random; it follows laws of physics.
- Human voices have specific "colors" (frequencies):
  - **Pitch**: The "low" or "high" of your voice. With frequencies ranging from 85 to 155 Hz for male, and from 165 and 255 Hz for female
  - **Formants**: Peaks of energy caused by the shape of your throat and mouth. First Formant has frequencies ranging from 200 to 1000 Hz. Second Formant from 800 to 2500 Hz
- Speaker identity is highly dependent on **specific frequency regions**
- The Key Idea: If we know that identity is hidden in these specific frequency "bands," **why not build an AI that only looks for those**?.

M. Ravanelli & Y. Bengio, "Speaker Recognition From Raw Waveform With Sincnet", IEEE STL 2018.

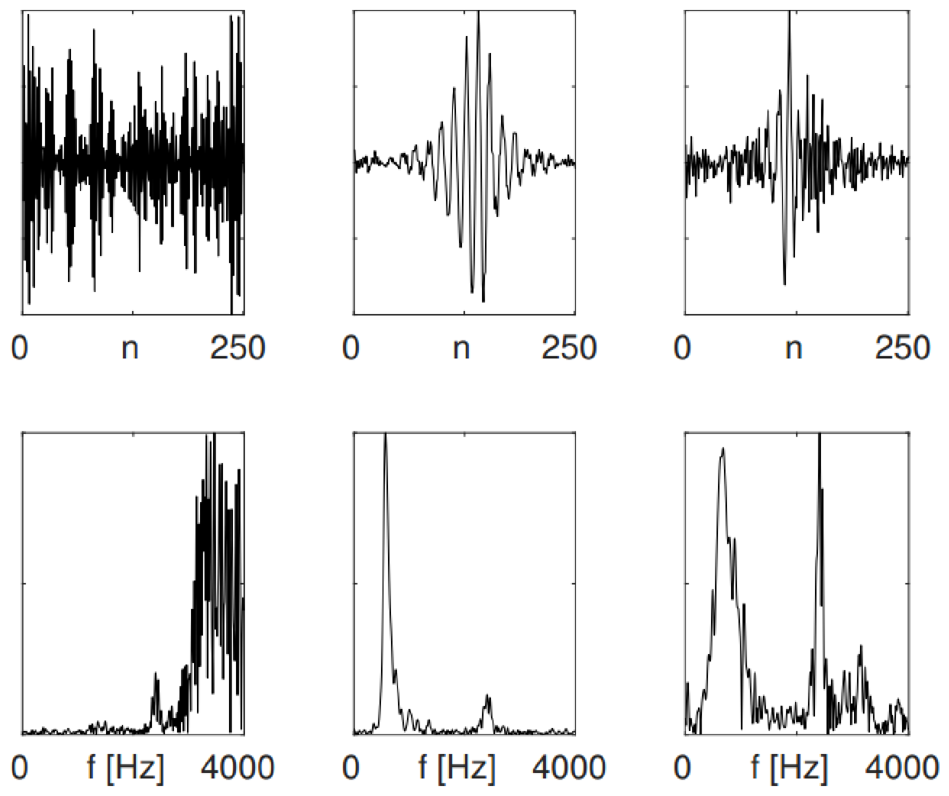
What do we know about Sound?

- A net receives raw audio as input, which is a signal that varies over time (waveform).
- To clip frequency at given position, we need to perform a **convolution** between the signal and a mathematical function (the filter)
- Instead of letting the AI learn any random pattern, we force it to **use Band-pass filters**.
- It’s like giving the AI a set of high-quality "tuning knobs" for a radio. It can only hear sounds within specific ranges, ignoring the "static" in between.

M. Ravanelli & Y. Bengio, “Speaker Recognition From Raw Waveform With Sincnet”, IEEE STL 2018.

The "Black Box" Struggle with Sound

- Standard AI (CNNs) looks at sound like a wall of random numbers.
- The first "layer" of the AI tries to find patterns from scratch.
- Without guidance, the AI often **learns messy, noisy filters** that don't make sense to humans and waste computational energy.



(a) CNN Filters

M. Ravanelli & Y. Bengio, "Speaker Recognition From Raw Waveform With Sincnet", IEEE STL 2018.

SincNet

- Standard AI must learn every single point of a complex shape (hundreds of parameters).
- SincNet: Only needs to learn two numbers for each filter:
  - Where the "useful" sound starts.
  - Where the "useful" sound ends.
- Everything else is handled by **enforcing the `sinc` function** into the net ensuring the band-pass shape is physically perfect.

M. Ravanelli & Y. Bengio, "Speaker Recognition From Raw Waveform With Sincnet", IEEE STL 2018.

Sinc Function

- A general sinc function used is usually expressed as:

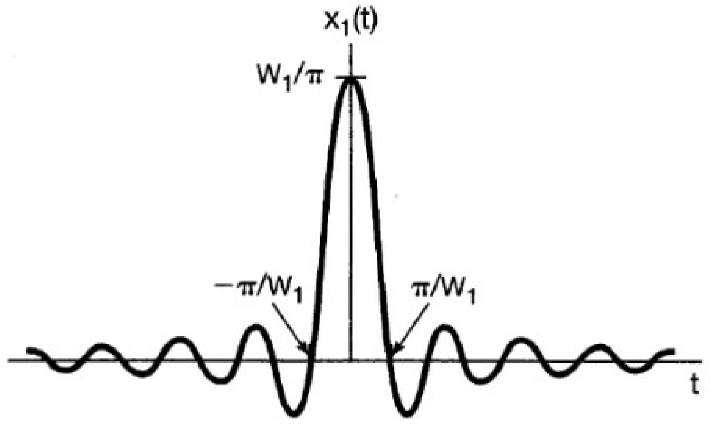
$$x(t) = 2f_c \cdot \mathrm{sinc}(2\pi f_c t) = 2f_c \frac{\sin(2\pi f_c t)}{2\pi f_c t}$$

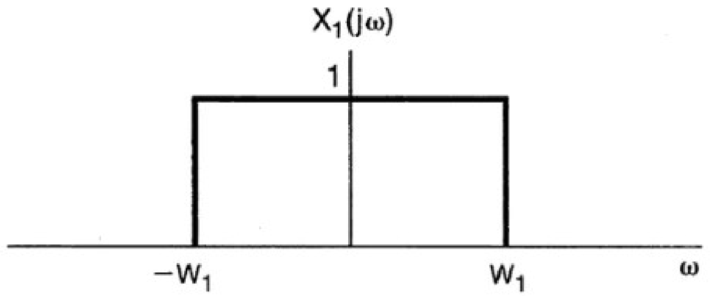
- In the frequency domain, the sinc becomes a perfect low-pass filter
- A band-pass filter can be obtained as the difference of two low-pass filters

$$G[f, f_1, f_2] = \mathrm{rect}\left(\frac{f}{2f_2}\right) - \mathrm{rect}\left(\frac{f}{2f_1}\right)$$

and in time

$$g[t, f_1, f_2] = 2f_2 \mathrm{sinc}(2\pi f_2 t) - 2f_1 \mathrm{sinc}(2\pi f_1 t)$$

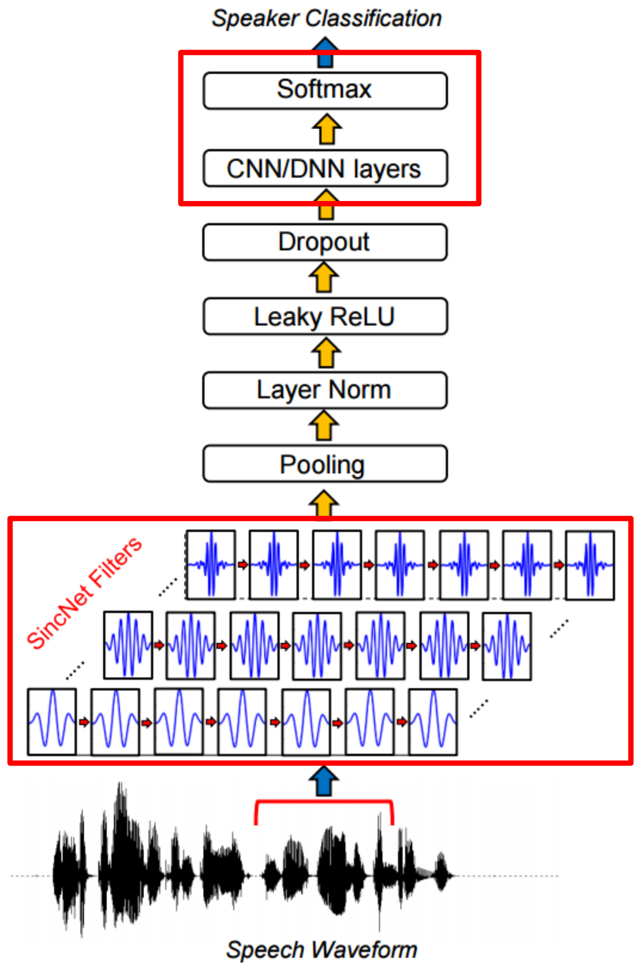




M. Ravanelli & Y. Bengio, "Speaker Recognition From Raw Waveform With Sincnet", IEEE STL 2018.

SincNet Architecture

- Organized in two main blocks
  - Classical CNN+DNN layers with Softmax for classification
  - Sinc Convolution layers in which the sinc function is enforced, named SincNet Filters
- Each SincNet Filter must learn only two parameter that are the cut-off frequency of the filter
  - From the learnt parameters a filter of length L is computed



M. Ravanelli & Y. Bengio, "Speaker Recognition From Raw Waveform With Sincnet", IEEE STL 2018.

SincNet

- Code available at https://github.com/mravanelli/SincNet/

- the `sinc_conv` do not define a classical convulitional layer, but only two `Parametes` to be trained

```python
class sinc_conv(nn.Module):
    def __init__(self, N_filt, Filt_dim, fs):
        super(sinc_conv, self).__init__()
        # --- Mel Initialization (Domain Knowledge) ---
        # The paper suggests initializing filters using the mel-scale [3].
        # This allocates more filters to the lower spectrum where crucial speaker
        # identity clues (like pitch and formants) are located [3, 4].
        low_freq_mel = 80
        high_freq_mel = (2595 * np.log10(1 + (fs / 2) / 700))  # Hz to Mel conversion
        mel_points = np.linspace(low_freq_mel, high_freq_mel, N_filt)  # Equally spaced in Mel
        f_cos = (700 * (10**(mel_points / 2595) - 1)) # Mel to Hz conversion
        # Determine initial frequency bands (b1=start, b2=end)
        b1 = np.roll(f_cos, 1)
        b2 = np.roll(f_cos, -1)
        b1 = 30
        b2[-1] = (fs / 2) - 100
        self.freq_scale = fs * 1.0
        # --- Learnable Parameters (2 per filter) ---
        # Instead of learning L elements (taps), SincNet only learns 2:
        # the low (f1) and high (f2) cutoff frequencies [1, 5].
        # In this implementation, the parameters are the start (b1) and the bandwidth (b2-b1).
        self.filt_b1 = nn.Parameter(torch.from_numpy(b1 / self.freq_scale))
        self.filt_band = nn.Parameter(torch.from_numpy((b2 - b1) / self.freq_scale))

        self.N_filt = N_filt
        self.Filt_dim = Filt_dim
        self.fs = fs
```

M. Ravanelli & Y. Bengio, “Speaker Recognition From Raw Waveform With Sincnet”, IEEE STL 2018.

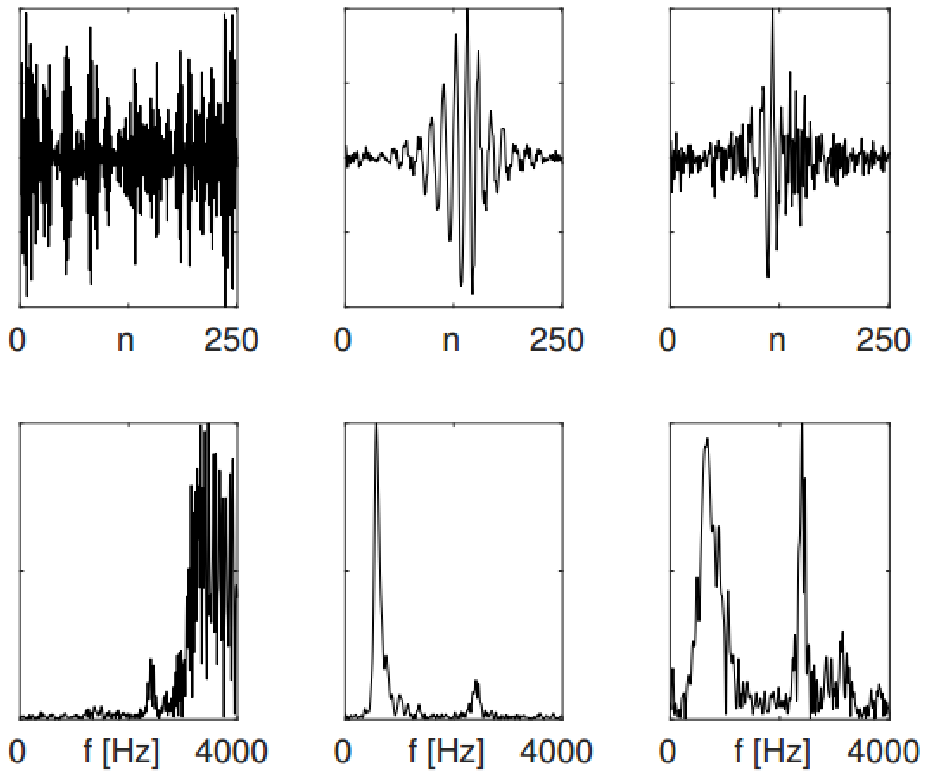
SincNet

- In the forward pass the sinc filter are then built from the Parameters

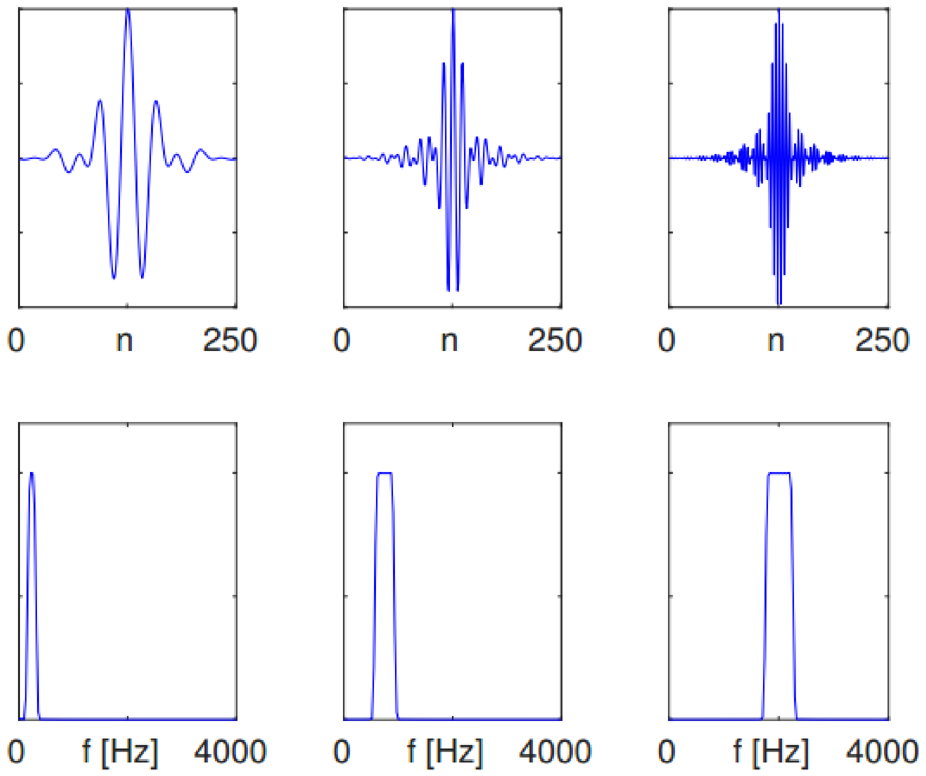
```python
def forward(self, x):
    ...
    # --- Filter Generation (The Sinc Formula) ---
    # The layer creates a rectangular band-pass filter in the frequency domain.
    # In the time domain, this is equivalent to the difference between two
    # sinc functions: g[n, f1, f2] = 2f2*sinc(2pi*f2n) - 2f1*sinc(2pi*f1n) [3].
    for i in range(self.N_filt):
        # Difference of two sincs to create the band-pass effect [3]
        low_pass1 = 2 * filt_beg_freq[i].float() * sinc(filt_beg_freq[i].float() * self.freq_scale, t_right)
        low_pass2 = 2 * filt_end_freq[i].float() * sinc(filt_end_freq[i].float() * self.freq_scale, t_right)
        band_pass = (low_pass2 - low_pass1)
        # Normalization and application of the Hamming window [8]
        band_pass = band_pass / torch.max(band_pass)
        filters[i, :] = band_pass.cuda() * window
    # --- Differentiable Convolution ---
    # All operations are fully differentiable, allowing the cutoff frequencies
    # to be optimized via backpropagation [8].
    out = F.conv1d(x, filters.view(self.N_filt, 1, self.Filt_dim))
    return out
```

M. Ravanelli & Y. Bengio, "Speaker Recognition From Raw Waveform With Sincnet", IEEE STL 2018.

Improved results



(a) CNN Filters



(b) SincNet Filters

M. Ravanelli & Y. Bengio, “Speaker Recognition From Raw Waveform With Sincnet”, IEEE STL 2018.

Benefits

- By using physics to simplify the problem, we need way fewer parameters
  - Standard AI layer: ~8,000 parameters to learn.
  - SincNet layer: Only 160 parameters.
- SincNet naturally tunes itself to human pitch and speech structures without being told to
- Physics tells us these filters are Symmetric (mirrored) and SincNet math exploits this: it calculates only 50% of the data and "flips" it to get the rest.
  - This makes the first layer twice as fast as a standard "blind" network.

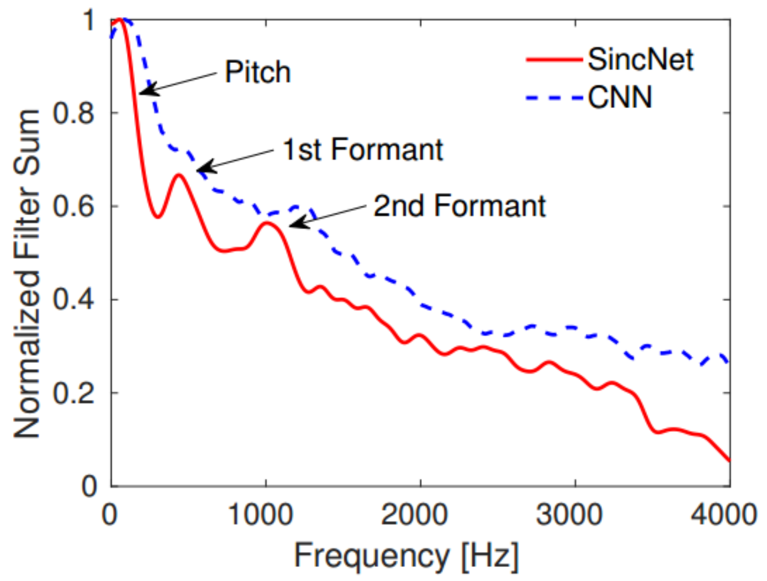


Fig. 3: Cumulative frequency response of the SincNet filters.

Knowledge of the domain (Sound Physics) makes AI **faster, more reliable, and easier to understand**

M. Ravanelli & Y. Bengio, "Speaker Recognition From Raw Waveform With Sincnet", IEEE STL 2018.

Geometric constraints

3D stereo and deep visual odometry

The Challenge of Monocular Depth Estimation

- Estimating 3D depth from a 2D image is an **inherently ill-posed problem**, as different 3D scenes can project into the same 2D image.
- Standard networks must learn to interpret **perspective**, **object scaling**, and **shading** without a direct 3D measurement.
- Traditionally, training requires LIDAR or laser scans, which are **expensive**, **hard to synchronize** with images, and often **sparse**.

Beyond Supervision: Learning from Stereo Video

- Binocular Supervision: Instead of ground truth depth, we can use rectified stereo image pairs (left/right images) during training.

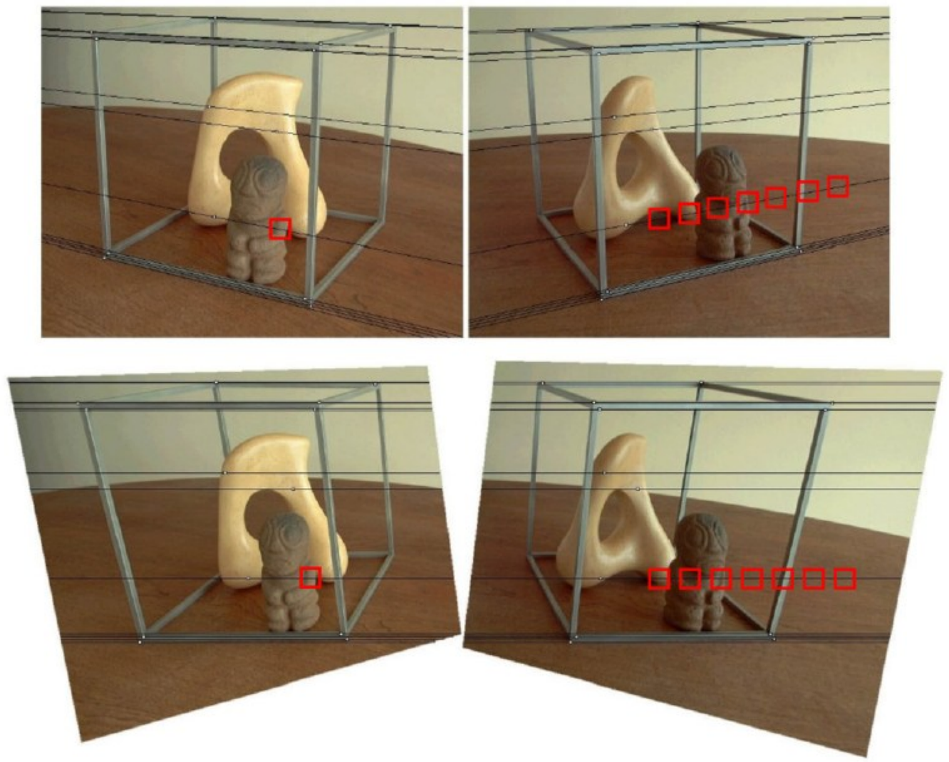


Image from A. Geiger

Beyond Supervision: Learning from Stereo Video

- If we know the **camera baseline and calibration**, the geometric relationship between the two images **acts as a supervisor**.
- Train the network to **synthesize one view from the other**; to do this, it must **learn depth** (or disparity d) as an intermediate step.

Stereo Disparity Map

Image A

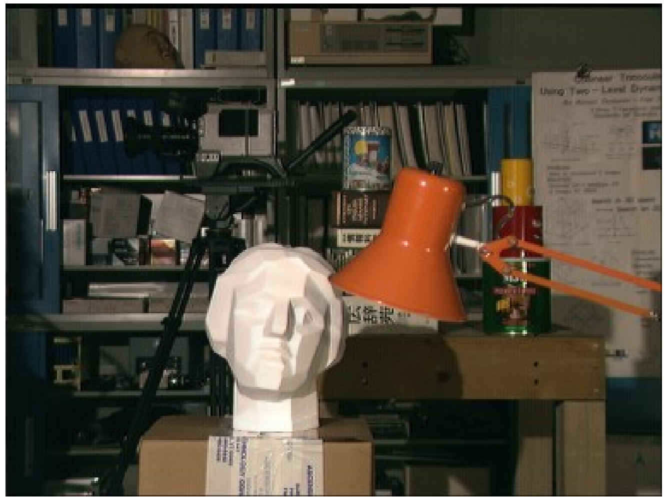
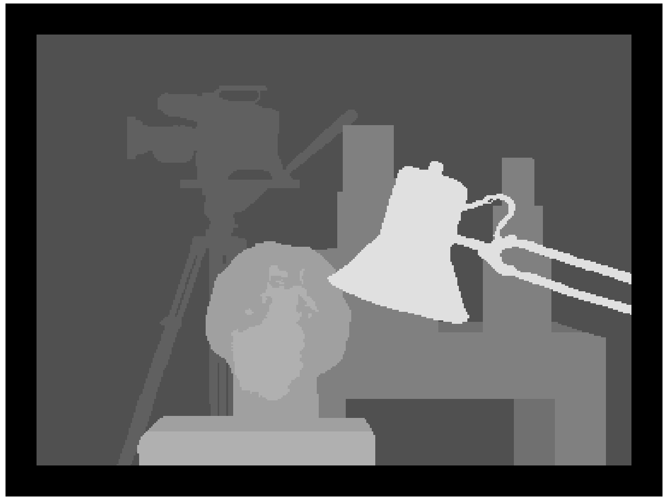


Image B


Disparity map

- A disparity map encodes for each pixel its shift from the first to the second image
  – Closer points will have higher disparity
  – Further points will have lower (or zero) disparity

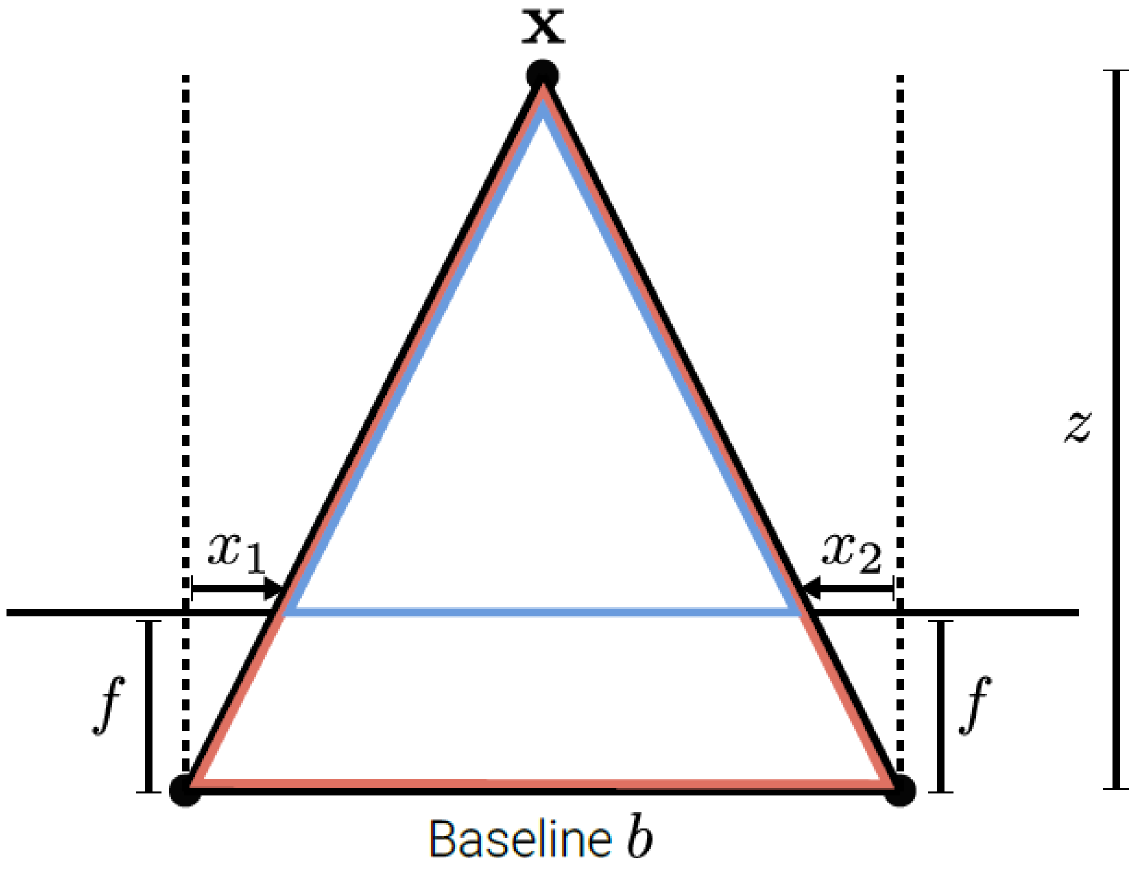
Depth from disparity

Let $d = x_1 - x_2$, then

$$\frac{Z - f}{b - d} = \frac{Z}{b}$$

$$Zb - fb = Zb - Zd$$

$$Z = \frac{fb}{d}$$

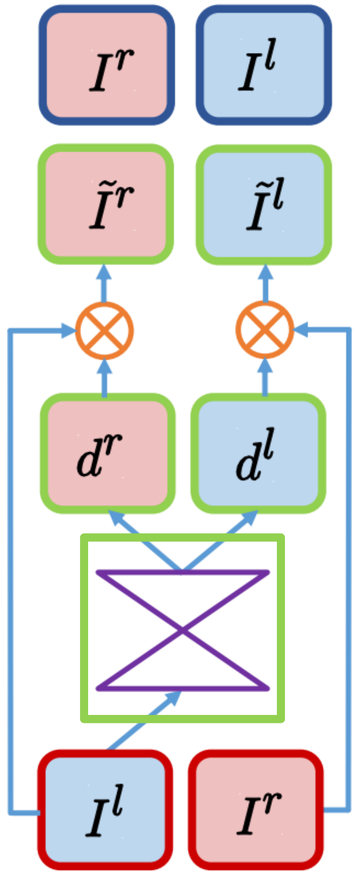


The Image Reconstruction Loss

- Given the left image (I_l), the network predicts a **disparity map** (d).
- We use this disparity to **warp** I_l to look like the right image (I_r).
- The difference between the actual right image and our synthesized version is the **Image Reconstruction Loss** (C_{ap}).

The Image Reconstruction Loss

- The used architecture is inspired by the DispNet, with fully convolutional encoder and decoder

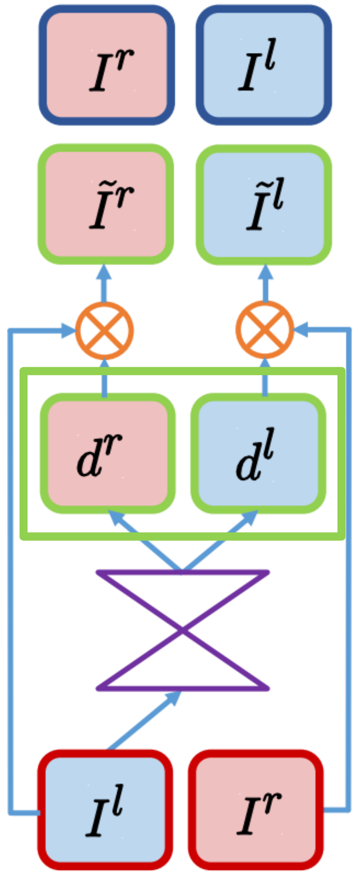


Godard et al., "Unsupervised monocular depth estimation with left-right consistency", CVPR, 2017.

The Image Reconstruction Loss

- The used architecture is inspired by the DispNet, with fully convolutional encoder and decoder

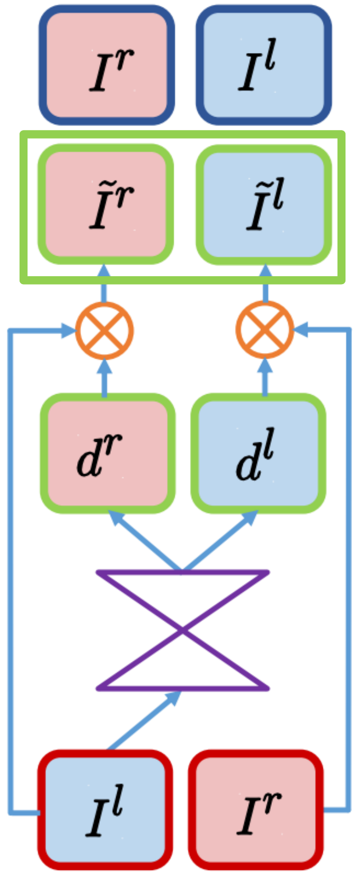
- At training time, I^l is used to produce both the **left** (d^l) and **right** (d^r) **disparity maps**



Godard et al., "Unsupervised monocular depth estimation with left-right consistency", CVPR, 2017.

The Image Reconstruction Loss

- The used architecture is inspired by the DispNet, with fully convolutional encoder and decoder
- At training time, I^l is used to produce both the **left** (d^l) and **right** (d^r) **disparity maps**
- Then the disparities are used to **reconstruct the input images**
  - $\tilde{I}^r = I^l(d^r)$
  - $\tilde{I}^l = I^r(d^l)$



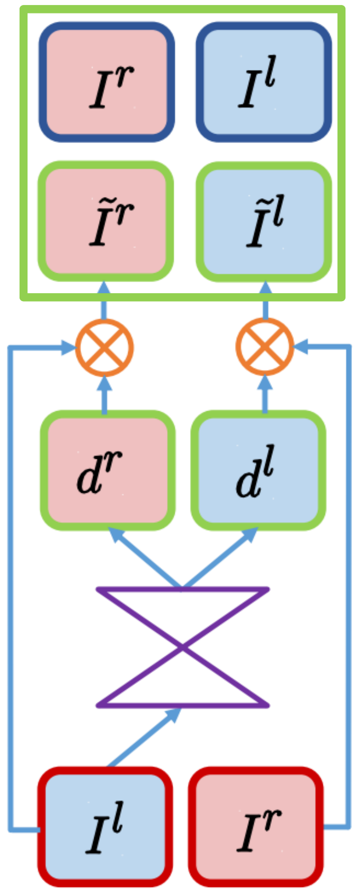
Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

The Image Reconstruction Loss

- To train the net, the reconstructed images $(\tilde{I}^l, \tilde{I}^r)$ are compared with the original ones (I^l, I^r)
- The loss function, evaluated at multiple scale, is composed by several parts

$$C = \sum_{s=1}^{4} C_s$$

$$C_s = \alpha_{ap}\left(C_{ap}^l + C_{ap}^r\right) + \alpha_{ds}\left(C_{ds}^l + C_{ds}^r\right) + \alpha_{lr}\left(C_{lr}^l + C_{lr}^r\right)$$

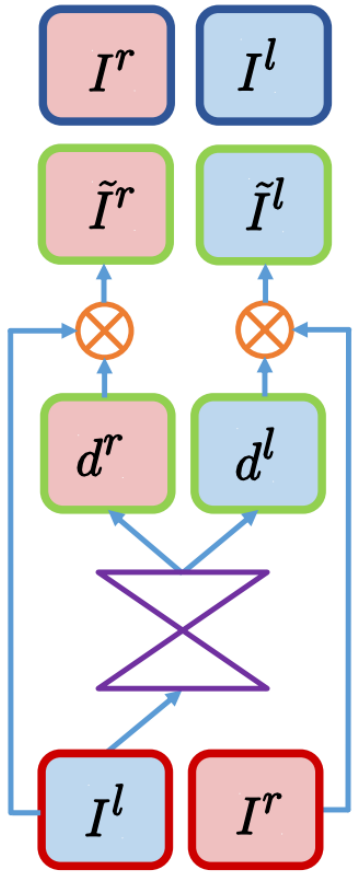


Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

The Image Reconstruction Loss

$$C_s = \boxed{\alpha_{ap}\left(C_{ap}^{l} + C_{ap}^{r}\right)} + \boxed{\alpha_{ds}\left(C_{ds}^{l} + C_{ds}^{r}\right)} + \boxed{\alpha_{lr}\left(C_{lr}^{l} + C_{lr}^{r}\right)}$$

- Appearance matching loss
  - Encourages the reconstructed images to appear similar to the input images
- Disparity smoothness loss
  - Encourages the predicted disparity maps to be smooth evaluating their gradients, weighted with the image gradient to take edges into account
- Left-right disparity consistency loss
  - To force the predicted disparity (both obtained from the left image only) to be consistent, i.e. have the left disparity equal to the projected right-view disparity



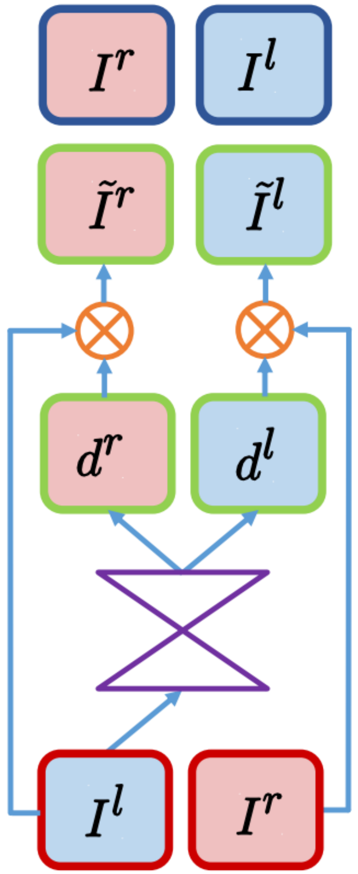
Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

The Image Reconstruction Loss

$$C_s = \boxed{\alpha_{ap}\left(C_{ap}^{l} + C_{ap}^{r}\right)} + \alpha_{ds}\left(C_{ds}^{l} + C_{ds}^{r}\right) + \alpha_{lr}\left(C_{lr}^{l} + C_{lr}^{r}\right)$$

- Appearance matching loss
  - Encourages the reconstructed images to appear similar to the input images

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



Godard et al., "Unsupervised monocular depth estimation with left-right consistency", CVPR, 2017.

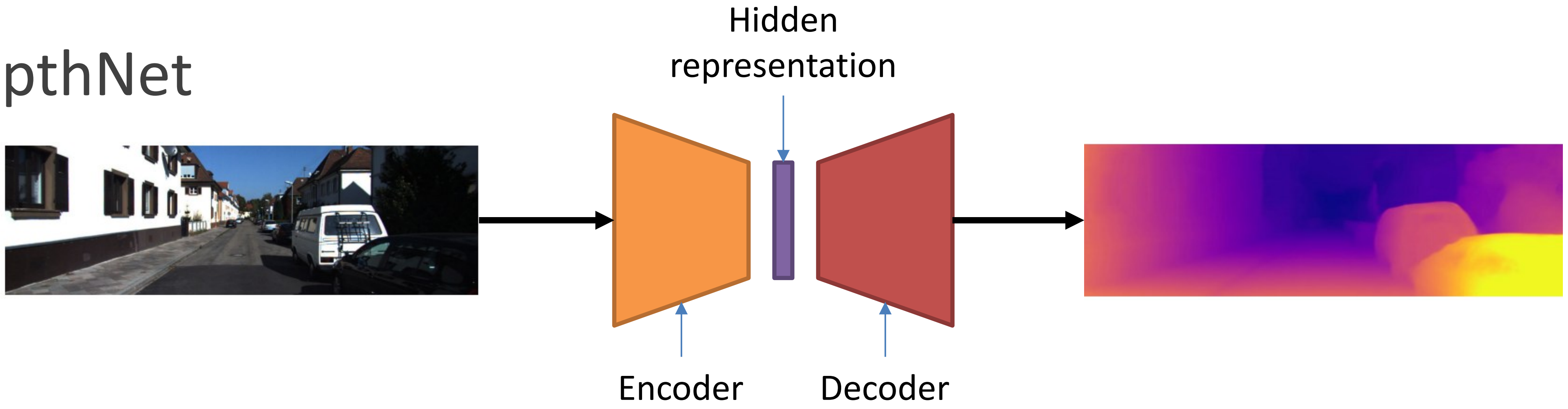
Moving toward Deep Visual Odometry

- D3VO* extends the concept of self-supervision to three levels: **Depth**, **Pose**, and **Uncertainty**.
- Instead of just using static stereo pairs, it uses **video sequences** to learn how cameras move over time.
- The network predictions are **used to boost** traditional geometric SLAM pipelines.

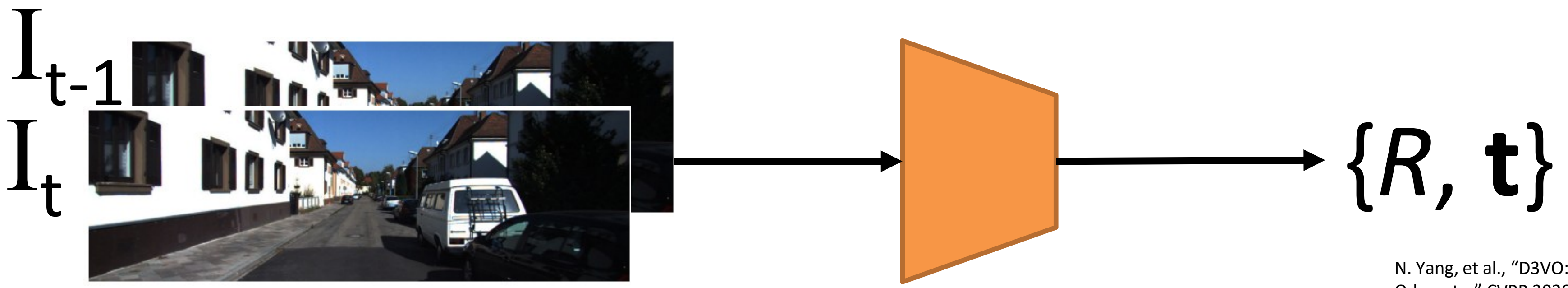
*N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

D3VO

DepthNet

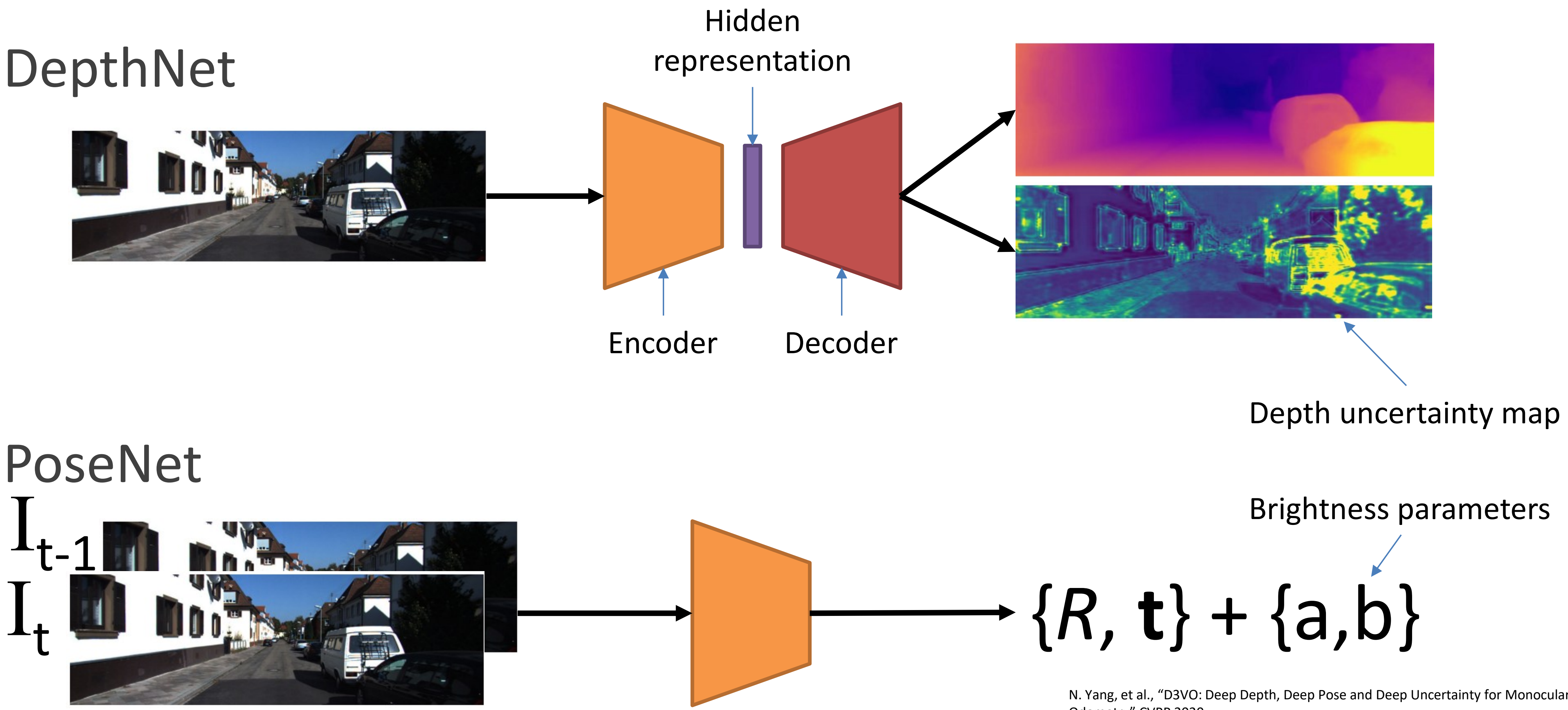


PoseNet



N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

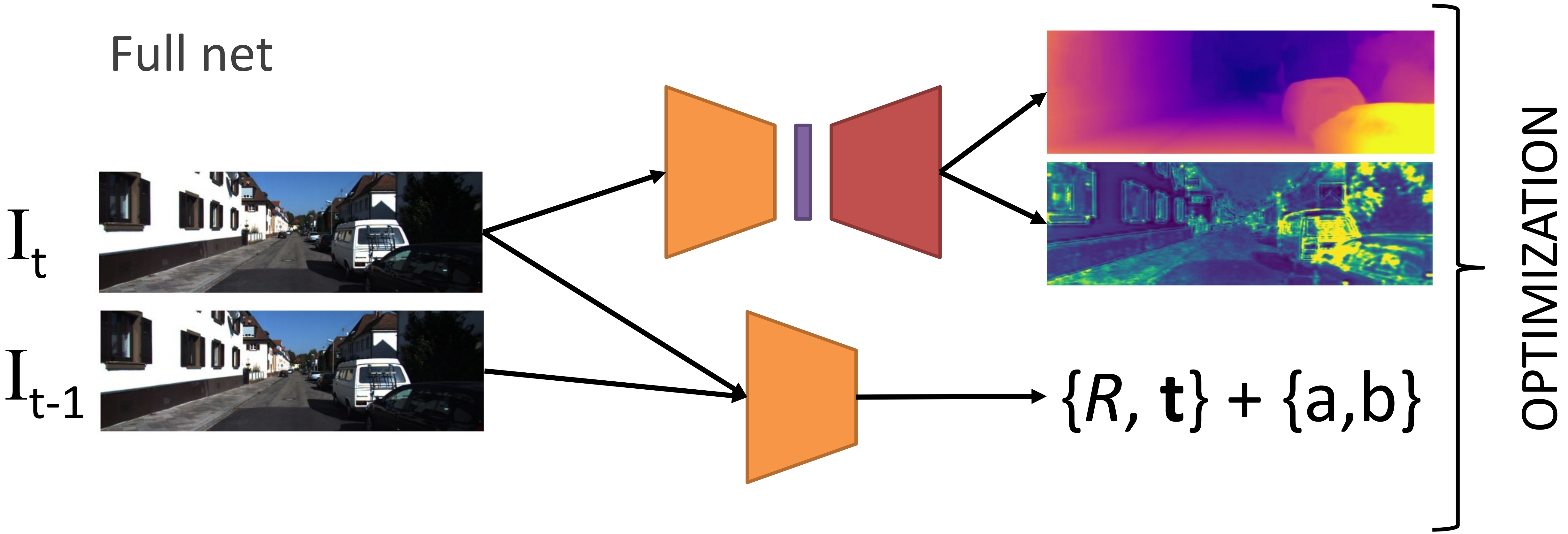
D3VO



N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

D3VO

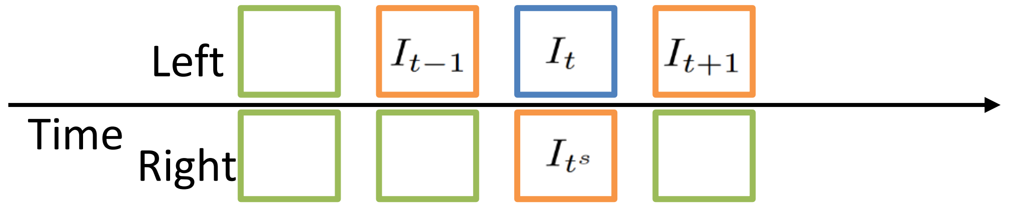
Full net



N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

D3VO

- To train the net, stereo pairs are required



- At first the loss function try to minimize the photometric re-projection error

$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \min_{t'} r(I_t, I_{t' \to t})$$

where $I_{t'} \in \{I_{t-1}, I_{t+1}, I_{t^s}\}$ and $I_{t' \to t}$ is the warping of I_t using the transformation $T_t^{t'}$ predicted by the PoseNet, and D_t, i.e. the depth map predicted by the DepthNet. Note that $T_t^{t^s}$ is known and fixed.

N. Yang, et al., “D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry” CVPR 2020.

D3VO

- The function *r* in the loss is

$$r(I_a, I_b) = \frac{\alpha}{2}(1 - SSIM(I_a, I_b)) + (1 - \alpha)\|I_a - I_b\|_1$$

where SSIM is the Structural Similarity Index. *r* is based on the **brightness constancy assumption** (BCA), that can be violated by changes in illumination.

- To better guarantee to satisfy the BCA, **brightness transformation parameter** are learned and used to adapt the illumination of I_t with $I_{t'}$ extending the loss as

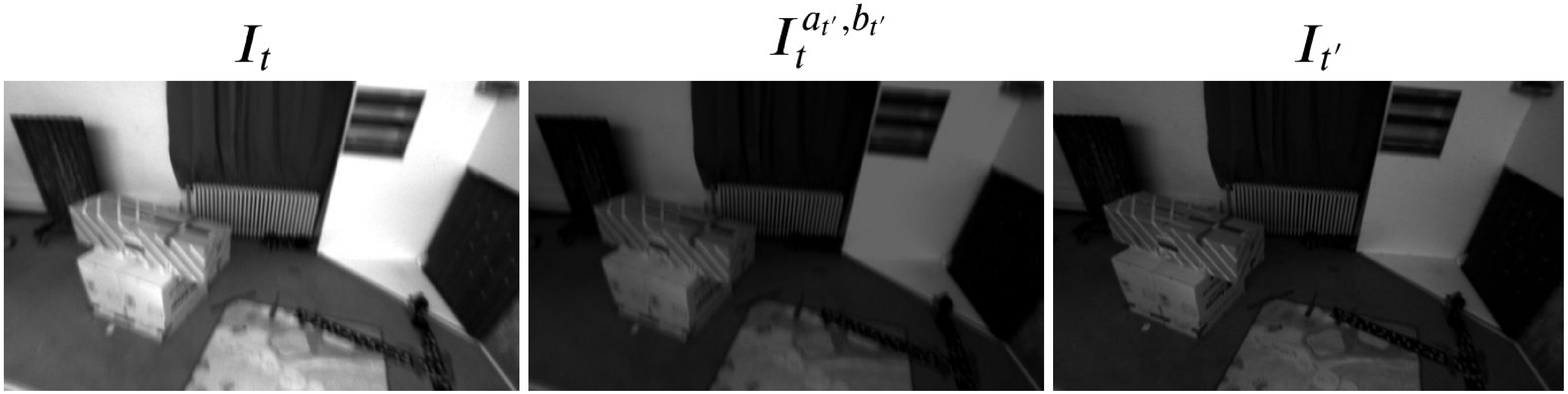
$$L_{self} = \frac{1}{|V|}\sum_{\mathbf{p}\in V} \min_{t'} r\left(I_t^{a_{t'}, b_{t'}}, I_{t' \to t}\right)$$

where $I_t^{a_{t'}, b_{t'}} = a_{t \to t'} I_t + b_{t \to t'}$

N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

D3VO

- Example result of brightness adaptation

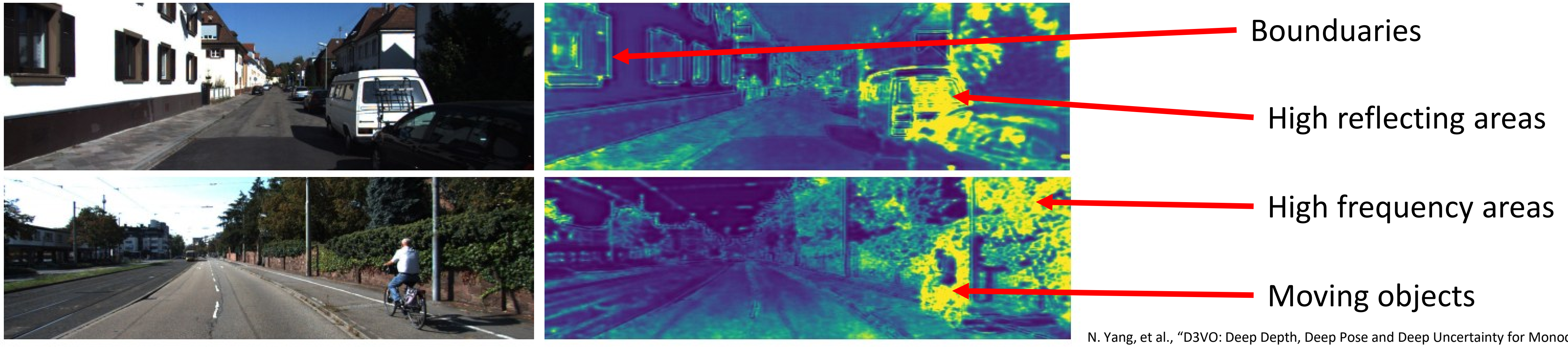


N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

D3VO

- To further improve the robustness of the system w.r.t. noisy data, the loss function is expanded to include an uncertainty map

$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \frac{\min_{t'} r\left(I_t^{a_{t'},b_{t'}}, I_{t' \to t}\right)}{\boxed{\Sigma_t}}$$



N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

D3VO

- Finally, in order to avoid degenerate solution, a regularization term is introduced, considering both the brightness parameters and the uncertainty map

$$L_{total} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \frac{\min_{t'} r\left(I_t^{a_{t'}, b_{t'}}, I_{t' \to t}\right)}{\Sigma_t} + \log \Sigma_t + \sum_{t'} (a_{t'} - 1)^2 + b_{t'}^2$$

N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

Semantic loss

Perception vs. Logic

- **Deep Learning** (*Perception*): Exceptional at recognizing faces, voices, or **patterns in high-dimensional data**. However, it is a "black box" that **lacks common sense** and requires massive amounts of data.
- **Symbolic AI** (*Reasoning*): Excellent at following **high-level rules and logical constraints**. However, it suffers from the "Symbol Grounding" problem—it doesn't know how to link abstract symbols to real-world pixels or sound.
- *Integrating logic into neural networks allows them to "see" like a computer and "reason" like a human.*

Why Logic Supports Neural Learning?

- Instead of letting a network guess randomly, we **use logic to define the *boundaries*** of the physical or logical world exploiting Domain Knowledge as a guide
- Logical rules allow the network to **learn from fewer examples** because the search space is constrained.
- Logic ensures the output is physically or **logically possible** (e.g., a person cannot be in two places at once).
- In semi-supervised settings, logic provides a *supervisory signal* for data that has no human labels.

Semi-Supervised Classification

- In many tasks, an object can belong to only one category (e.g., a digit is either a '1' or a '2', never both).
- Standard networks might predict 70% Cat and 70% Dog if they are unsure, violating logic.
- We define a Boolean sentence α that enforces the *exactly-one* rule across all output probabilities p.

The Semantic Loss Function

- The central idea is that semantic loss measures **how far the neural network is from satisfying a certain logical constraint α** on its predictions p
- Xu et al. define the semantic loss function as

$$L_s(\alpha, p) \propto -\log \sum_{x \models \alpha} \left(\prod_{i: x \models X_i} p_i \prod_{i: x \models \neg X_i} (1 - p_i) \right)$$

Xu, et al. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. Proceedings of the 35th International Conference on Machine Learning.

The Semantic Loss Function

$$L_s(\alpha, p) \propto -\log \sum_{\mathbf{x} \models \alpha} \left(\prod_{i:\mathbf{x} \models X_i} p_i \prod_{i:\mathbf{x} \models \neg X_i} (1 - p_i) \right)$$

- α logical sentence
- p vector of probability (each p_i is the output for variable X_i)
- $\Sigma_{\mathbf{x} \models \alpha}$ summation do not consider all the possible combinations but only those states $\mathbf{x}$ that make true α ($\mathbf{x}$ is any possible assignment of 0/1 to the variable X_i)
- $\prod p_i \prod (1 - p_i)$ compute the probability of a single state $\mathbf{x}$ (In practice, for every variable that in state $\mathbf{x}$ is true (=1), we take the probability p_i; for each variable that is false (=0), we take its complement $(1 - p_i)$
- $-\log$ turns the total probability of success into an error value that can be minimized

Xu, et al. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. Proceedings of the 35th International Conference on Machine Learning.

exactly-one Boolean sentence

Exactly-One constraint for multi-class classification, the Boolean sentence α is defined as follows:

- For a set of indicator variables $X = \{X_1, X_2, \dots, X_n\}$, the sentence α is the conjunction of two conditions:
  1. At least one variable must be true:
  $$(X_1 \vee X_2 \vee \cdots \vee X_n)$$
  2. At most one variable can be true:
  $$\bigwedge_{i<j} \left(\neg X_i \vee \neg X_j\right)$$
- The complete Boolean sentence α is:

$$\alpha = (X_1 \vee X_2 \vee \cdots \vee X_n) \wedge \bigwedge_{i<j} \left(\neg X_i \vee \neg X_j\right)$$

The Semantic Loss Function "Exactly-One"

- The general form of the semantic loss can be simplified in the case of "Exactly-One" problem

$$L_s(\text{exactly\_one}, p) \propto -\log \sum_{i=1}^{n} \left(p_i \prod_{j=1, j \neq i}^{n} (1 - p_j) \right)$$

- Since the constrain impose that at most one symbol should be true and at least one symbol should be true, it means that on n symbols there are only n valid states $\mathbf{x}$
- $\prod p_i$ simplify to p_i since in any valid state there is only one true variable
- $\prod(1 - p_i)$ becomes the product of all the other variable that are false

Xu, et al. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. Proceedings of the 35th International Conference on Machine Learning.

Semantic loss in training

- Semantic loss is added to the normal loss function (such as cross-entropy) as a **weighted regularization term**:

$$\text{Total Loss} = \text{Supervised Loss} + w \cdot \text{Semantic Loss}$$

- This approach allows the network to learn even from unlabelled data (**semi-supervised learning**)
- The learning signal does not derive from a *correct answer* provided by a human, but from the **intrinsic need to respect the logical rules** of the domain.

Xu, et al. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. Proceedings of the 35th International Conference on Machine Learning.

Semantic loss in training

- The Semantic Loss **forces the network to take a position** and build a safe (confident) hypothesis on all data, labelled and unlabelled.
- By encouraging the model to assign consistent classes even to unlabelled data, **the network learns to define a decision boundary** that is much more precise and physically sensible than it would achieve with just a few labelled examples.

Xu, et al. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. Proceedings of the 35th International Conference on Machine Learning.

Semantic loss in training

- The semantic loss is an intrinsic necessity because the constraint is not an optional suggestion, but a **structural rule of the domain that the network must respect** to minimize total loss
- Practical Example:
  - If we train a network on 100 labelled and 10,000 unlabelled images of handwritten digits, Semantic Loss "observes" the 10,000 unnamed images and tells the network, "*I don't know what these images are, but whatever you decide, you can't say they're both a '1' and a '7'. Choose one and be convinced*".
  - This simple consistency obligation allows to obtain results close to the state of the art (e.g. on datasets such as MNIST or CIFAR-10) using **only a minimal fraction of labelled data** (even just 1%).

Xu, et al. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. Proceedings of the 35th International Conference on Machine Learning.

The dependence on the base model and the weight w

The risk of mis-knowledge is not zero, but depends on two critical factors:

- Quality of the supervised model: Without the predictive power of an initial supervised model (even if trained on little data), **the benefits of Semantic Loss diminish**. If the initial 100 examples are totally unrepresentative, the network may actually converge towards a logically coherent but physically incorrect solution.
- The w weight of the loss: The total loss is a combination of supervised and semantic loss, weighted by w. Incorrect weight balancing w could **force the network to match logic at the expense of data reality**.

Xu, et al. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. Proceedings of the 35th International Conference on Machine Learning.

Dealing with complex constraints

- While the exactly-on constraint can be efficiently computed (simplified formula), more generally computing the probability that a complex logical rule is true is generally very hard (NP-hard)
- For intricate combinatorial objects such as paths on graphs, hierarchical classifications, or preference orderings, the number of valid states explodes exponentially

Xu, et al. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. Proceedings of the 35th International Conference on Machine Learning.