DS-GA.3001 Embodied Learning and Vision

Mengye Ren

NYU

Spring 2025

embodied-learning-vision-course.github.io

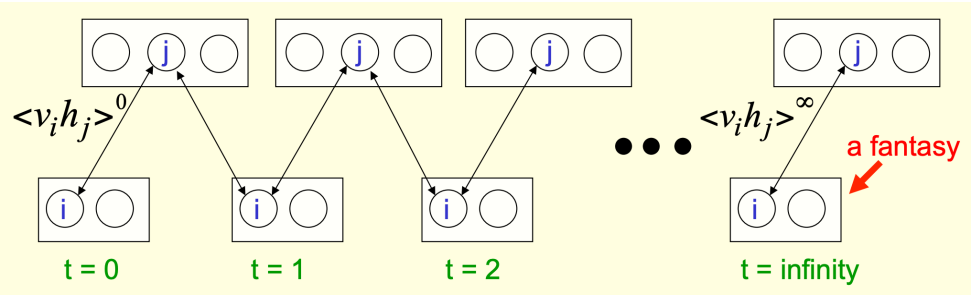
Lecture Slides for Note Taking

NYU

Energy-Based Learning

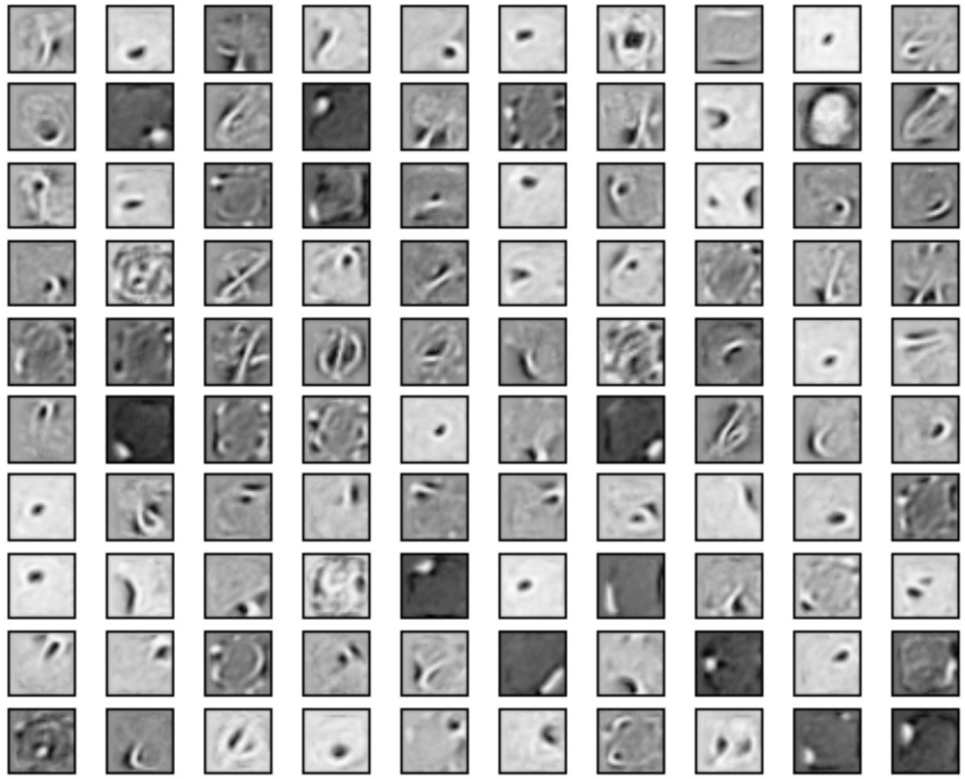
- Example: RBMs
- Energy: $E(v,h) = -\sum_{i,j} v_i h_j w_{ij}$.

$$p(h_j = 1|v_i) = \sigma(\sum_{ij} v_i w_{ij}).$$



$$\frac{\partial \log p(v)}{\partial w_{ij}} = < v_i h_j >^0 - < v_i h_j >^\infty .$$

Gibbs



Hinton. Restricted Boltzmann Machines.

NYU

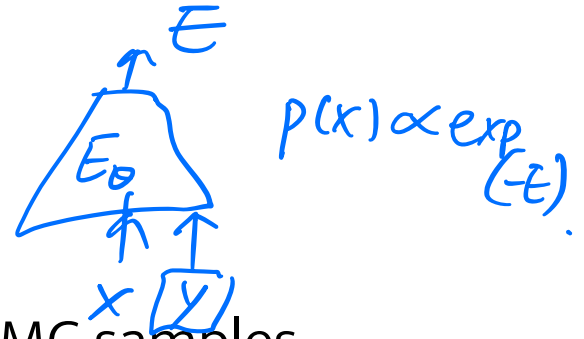
General EBMs

- Inference requires running gradient descent and MCMC samples (Langevin samples).

Du & Mordatch. Implicit Generation and Modeling with Energy-Based Models. NeurIPS 2019.

NYU

General EBMs



- Inference requires running gradient descent and MCMC samples (Langevin samples).

$$\tilde{\mathbf{x}}^k = \mathbf{x}^{k-1} - \frac{\lambda}{2}\nabla_{\mathbf{x}} E_\theta(\tilde{\mathbf{x}}^{k-1}) + \omega^k, \omega^k \sim \mathcal{N}(0, \lambda).$$

NN

noise

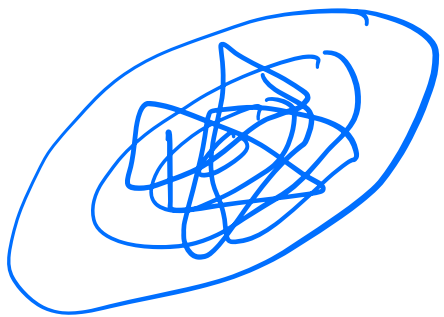
$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\mathbf{x}\sim p_D}[\nabla_\theta E_\theta(\mathbf{x}^+)] - \mathbb{E}_{\mathbf{x}\sim q_\theta}[\nabla_\theta E_\theta(\mathbf{x}^-)].$$

data.

model.

negative.

→ gradient descent on input.



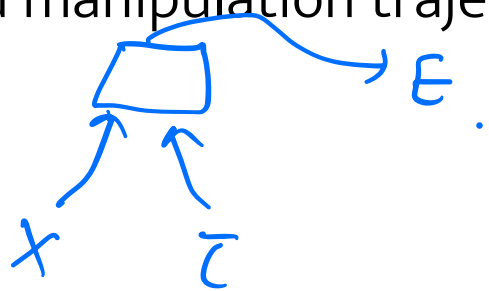
General EBMs

- Inference requires running gradient descent and MCMC samples (Langevin samples).

$$\tilde{\mathbf{x}}^k = \tilde{\mathbf{x}^{k-1}} - \frac{\lambda}{2}\nabla_{\mathbf{x}} E_\theta(\tilde{\mathbf{x}}^{k-1}) + \omega^k, \omega^k \sim \mathcal{N}(0, \lambda).$$

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\mathbf{x}\sim p_D}[\nabla_\theta E_\theta(\mathbf{x}^+)] - \mathbb{E}_{\mathbf{x}\sim q_\theta}[\nabla_\theta E_\theta(\mathbf{x}^-)].$$

- Can be applied on hand manipulation trajectory generation.



Du & Mordatch. Implicit Generation and Modeling with Energy-Based Models. NeurIPS 2019.

NYU

General EBMs

- Inference requires running gradient descent and MCMC samples (Langevin samples).

$$\tilde{\mathbf{x}}^k = \mathbf{x}^{\tilde{k-1}} - \frac{\lambda}{2}\nabla_{\mathbf{x}} E_\theta(\tilde{\mathbf{x}}^{k-1}) + \omega^k, \omega^k \sim \mathcal{N}(0, \lambda).$$

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\mathbf{x}\sim p_D}[\nabla_\theta E_\theta(\mathbf{x}^+)] - \mathbb{E}_{\mathbf{x}\sim q_\theta}[\nabla_\theta E_\theta(\mathbf{x}^-)].$$

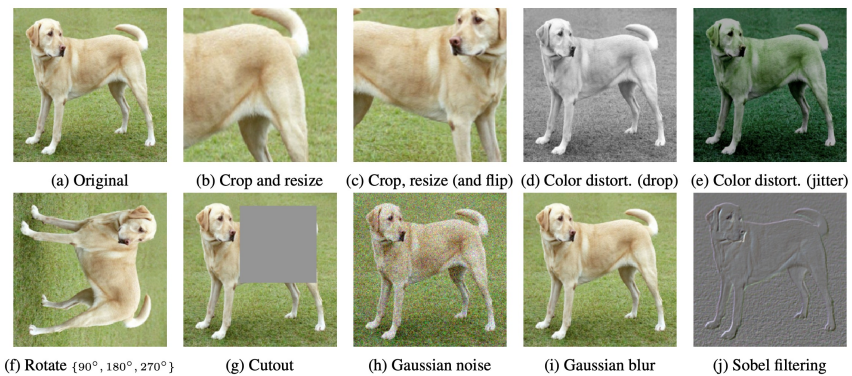
- Can be applied on hand manipulation trajectory generation.
- Good results in generation but still not a generalized representation learning algorithm.

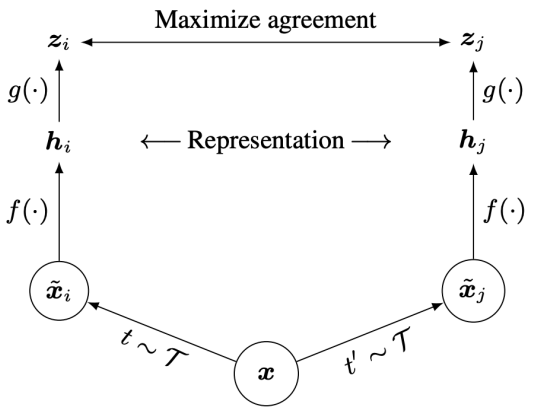
Du & Mordatch. Implicit Generation and Modeling with Energy-Based Models. NeurIPS 2019.

NYU

Self-Supervised Visual Learning

- Match the same image (with severe augmentation)

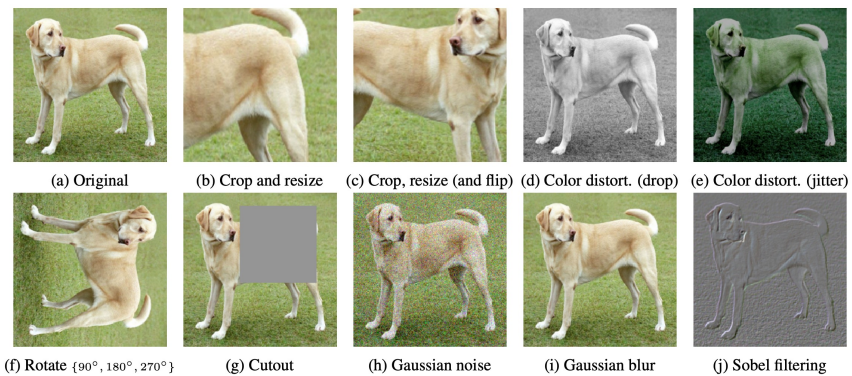


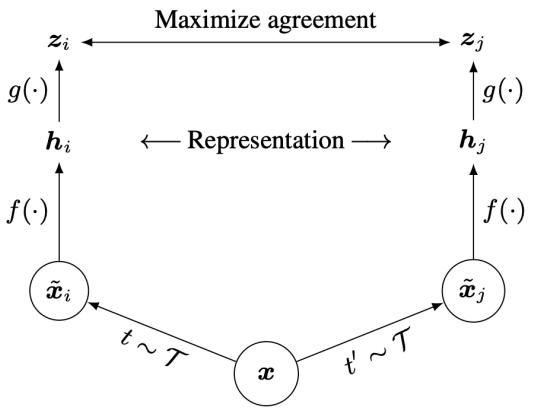


NYU

Self-Supervised Visual Learning

- Match the same image (with severe augmentation)
- Joint embedding approach: Apply loss on the embedding level.

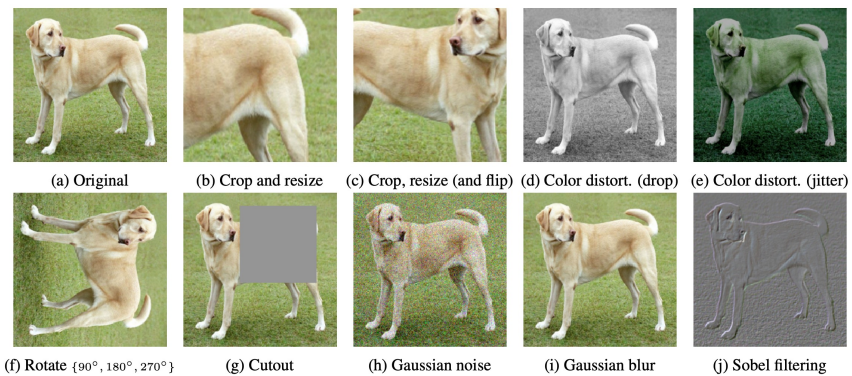


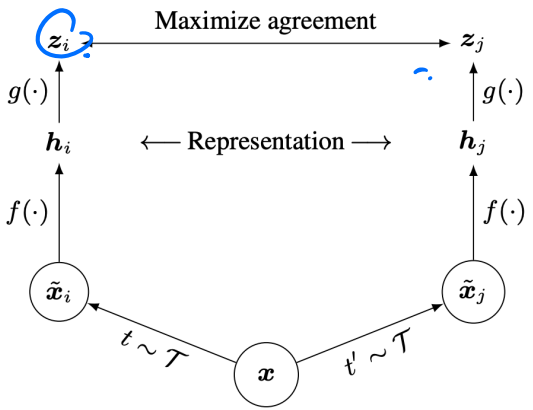


NYU

Self-Supervised Visual Learning

- Match the same image (with severe augmentation)
- Joint embedding approach: Apply loss on the embedding level.
- Use negative examples (contrastive) or not (non-contrastive).

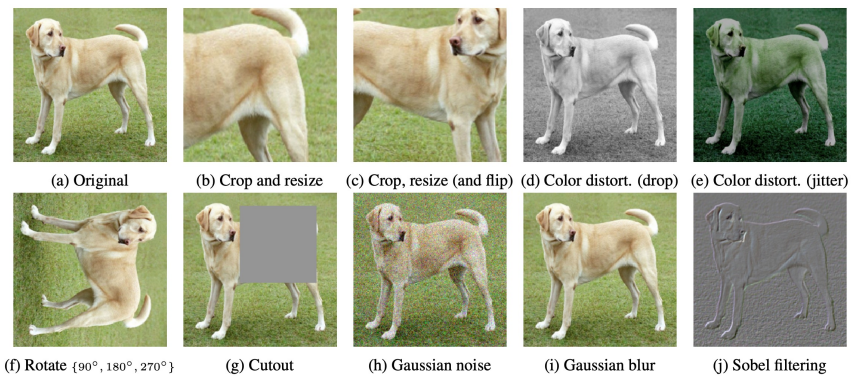


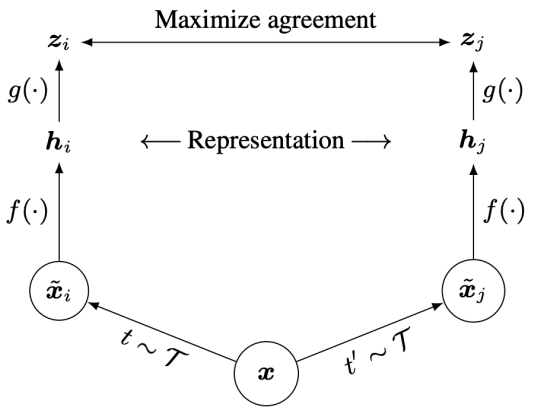


NYU

Self-Supervised Visual Learning

- Match the same image (with severe augmentation)
- Joint embedding approach: Apply loss on the embedding level.
- Use negative examples (contrastive) or not (non-contrastive).
- Energy is defined between a pair of images.



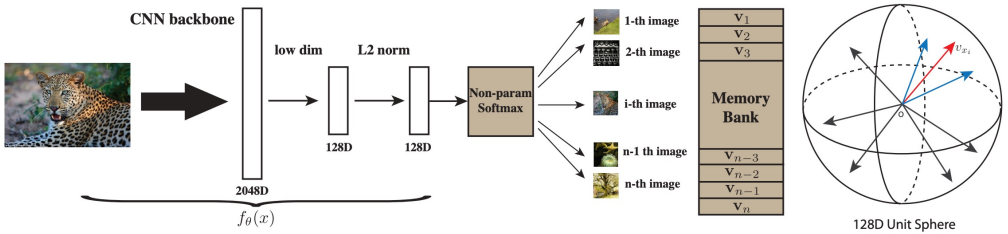


NYU

Several Embedding Loss Formulations

Wu et al., 2018

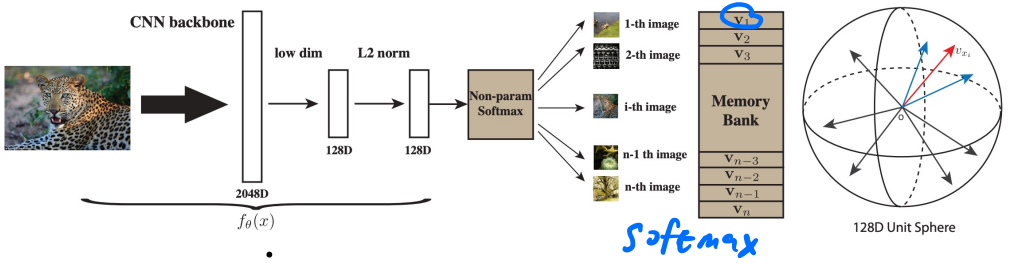
- Instance Classification:



Several Embedding Loss Formulations

Wu et al., 2018

- Instance Classification:



- Contrastive Learning: Cross entropy on pairs

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}.$$

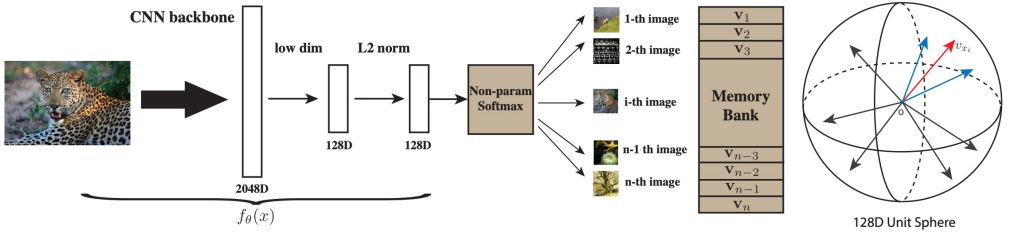
Chen et al. 2020

NYU

Several Embedding Loss Formulations

- Instance Classification:

Wu et al., 2018

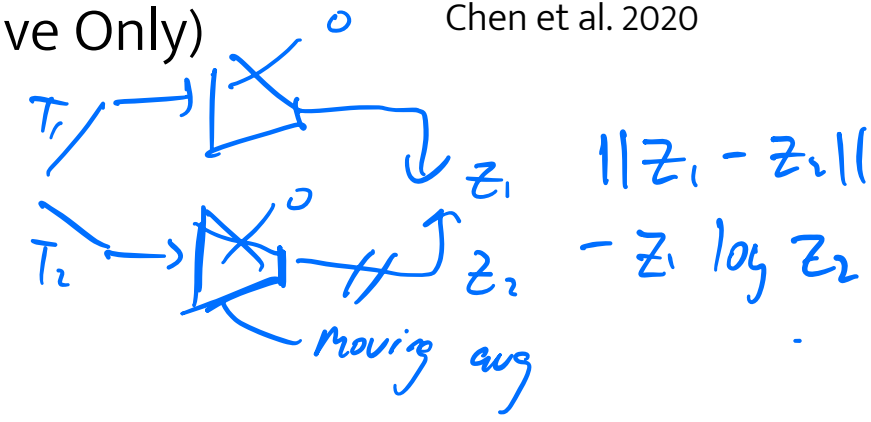


- Contrastive Learning: Cross entropy on pairs

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}.$$

Chen et al. 2020

- Non-contrastive Learning (Positive Only)
  - Moving Average [Grill et al., 2020]
  - Stop Gradient [Chen & He, 2020]

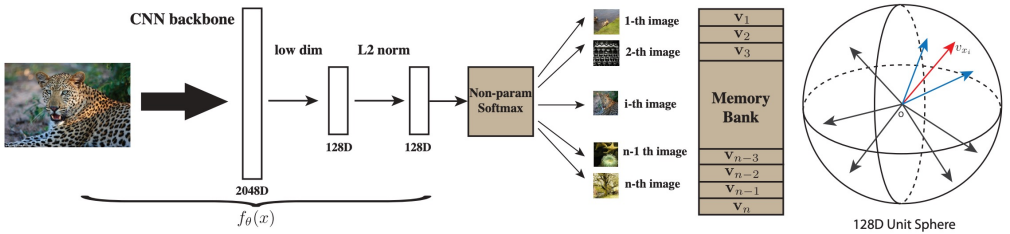


NYU

Several Embedding Loss Formulations

- Instance Classification:

Wu et al., 2018



- Contrastive Learning: Cross entropy on pairs

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}.$$

Chen et al. 2020

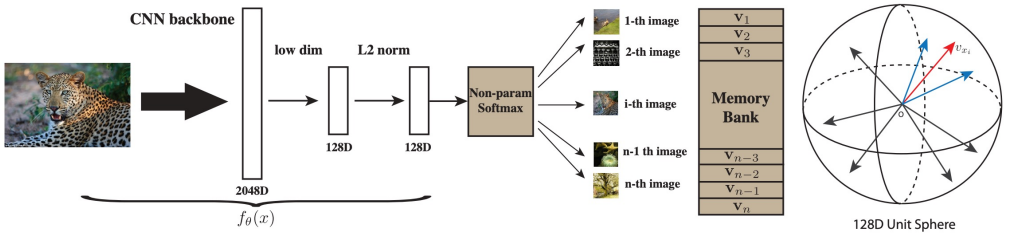
- Non-contrastive Learning (Positive Only)
  - Moving Average [Grill et al., 2020]
  - Stop Gradient [Chen & He, 2020]
- Use of projectors and predictors

NYU

Several Embedding Loss Formulations

- Instance Classification:

Wu et al., 2018

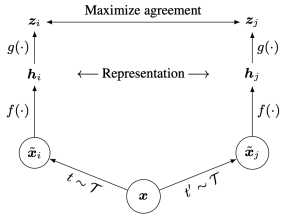


- Contrastive Learning: Cross entropy on pairs

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}.$$

Chen et al. 2020

- Non-contrastive Learning (Positive Only)
  - Moving Average [Grill et al., 2020]
  - Stop Gradient [Chen & He, 2020]
- Use of projectors and predictors



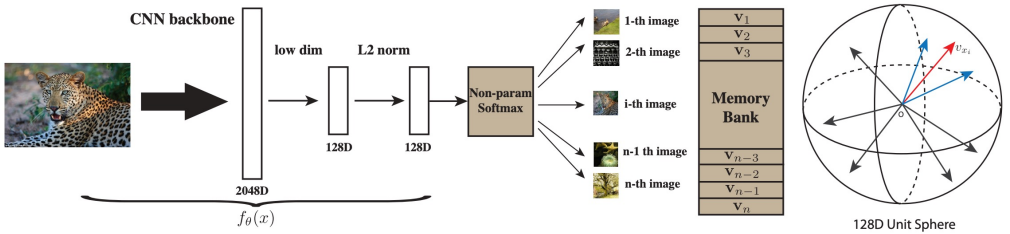
Chen et al. 2020

NYU

Several Embedding Loss Formulations

- Instance Classification:

Wu et al., 2018

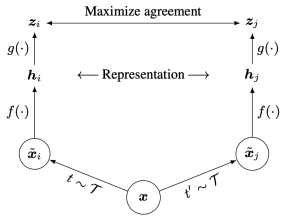


- Contrastive Learning: Cross entropy on pairs

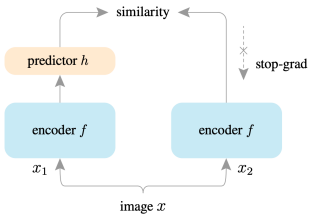
$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}.$$

Chen et al. 2020

- Non-contrastive Learning (Positive Only)
  - Moving Average [Grill et al., 2020]
  - Stop Gradient [Chen & He, 2020]
- Use of projectors and predictors



Chen et al. 2020



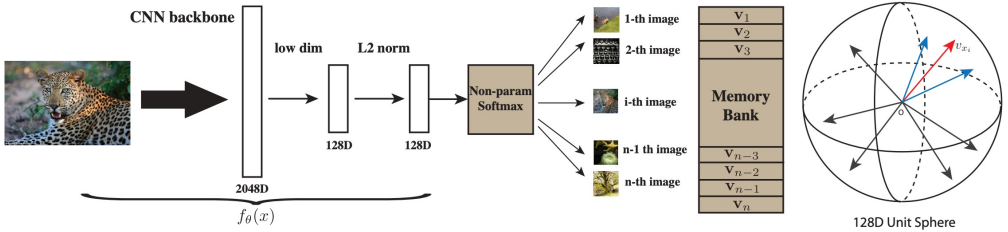
Chen & He, 2021

NYU

Several Embedding Loss Formulations

- Instance Classification:

Wu et al., 2018

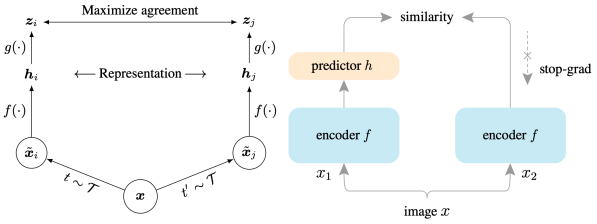


- Contrastive Learning: Cross entropy on pairs

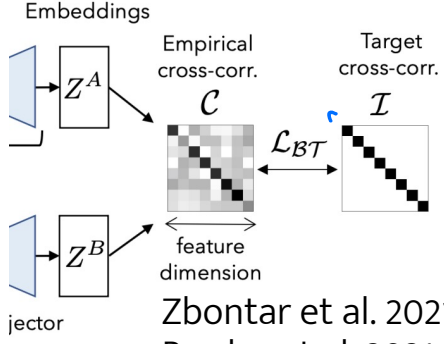
$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}.$$

Chen et al. 2020

- Non-contrastive Learning (Positive Only)
  - Moving Average [Grill et al., 2020]
  - Stop Gradient [Chen & He, 2020]
- Use of projectors and predictors
- Use of co-variance regularization



Chen et al. 2020 Chen & He, 2021

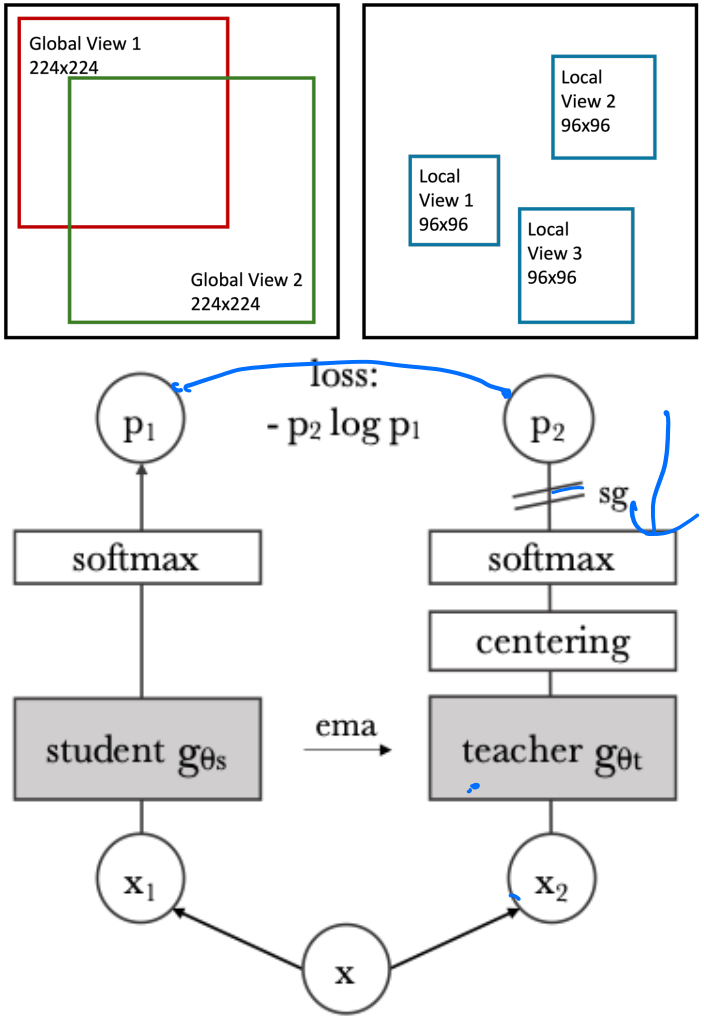


Zbontar et al. 2021
Bardes et al. 2021

NYU

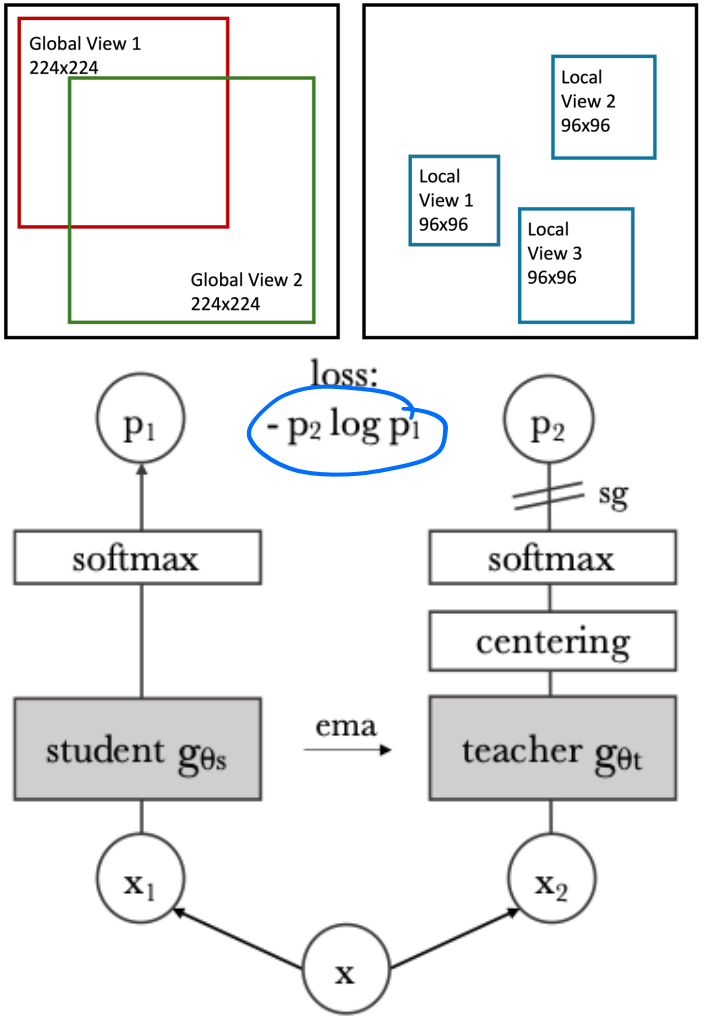
DINO

- Knowledge distillation between a student and a teacher network.



DINO

- Knowledge distillation between a student and a teacher network.
- Student: $p_s(x) = \frac{\exp(g_{\theta_s}(x)_i / \tau_s)}{\sum_k \exp(g_{\theta_s}(x)_k / \tau_s)}$.

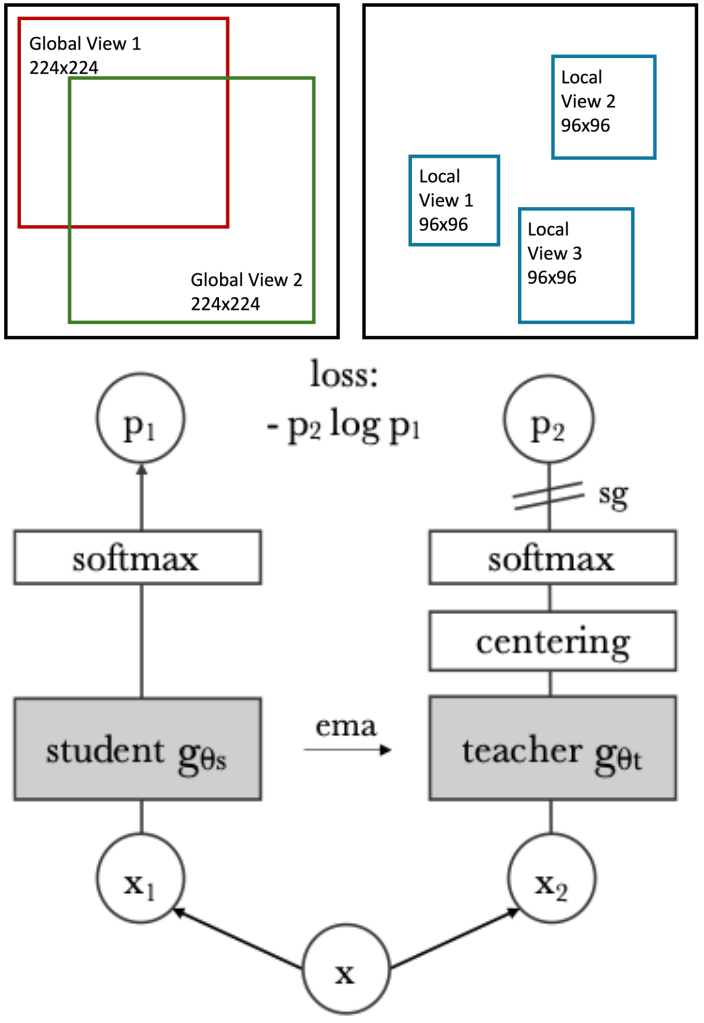


Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021.

NYU

DINO

- Knowledge distillation between a student and a teacher network.
- Student: $p_s(x) = \frac{\exp(g_{\theta_s}(x)_i/\tau_s)}{\sum_k \exp(g_{\theta_s}(x)_k/\tau_s)}$.
- Minimize CE: $\min_\theta H(p_t(x), p_s(x))$.

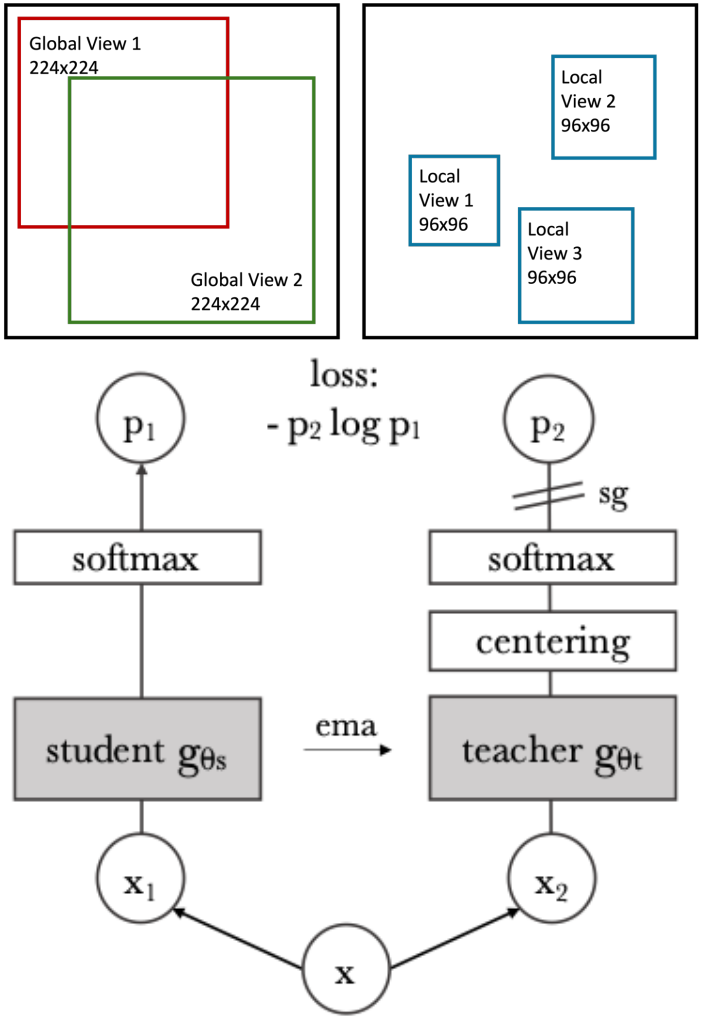


Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021.

NYU

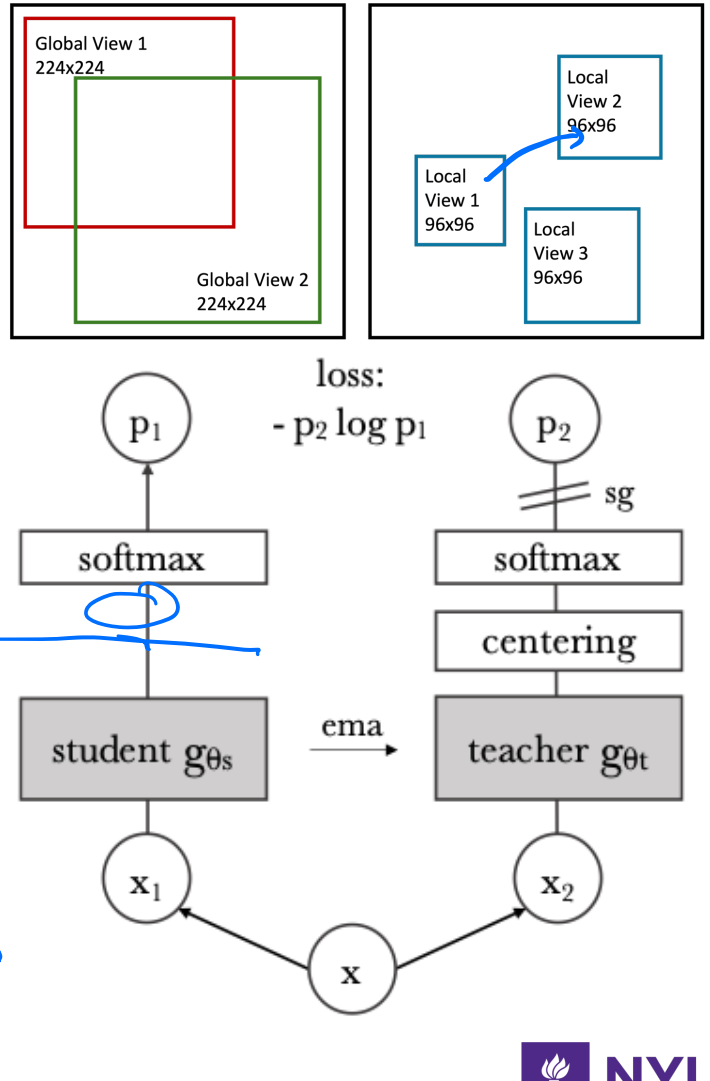
DINO

- Knowledge distillation between a student and a teacher network.
- Student: $p_s(x) = \frac{\exp(g_{\theta_s}(x)_i/\tau_s)}{\sum_k \exp(g_{\theta_s}(x)_k/\tau_s)}$.
- Minimize CE: $\min_\theta H(p_t(x), p_s(x))$.
- Stop gradient on the teacher (no true label).



DINO

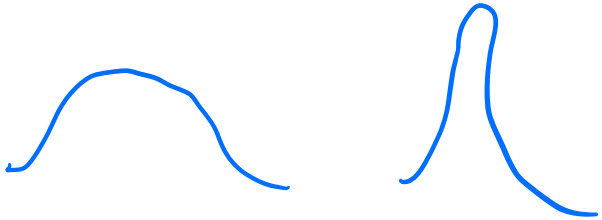
- Knowledge distillation between a student and a teacher network.
- Student: $p_s(x) = \frac{\exp(g_{\theta_s}(x)_i/\tau_s)}{\sum_k \exp(g_{\theta_s}(x)_k/\tau_s)}$.
- Minimize CE: $\min_\theta H(p_t(x), p_s(x))$.
- Stop gradient on the teacher (no true label).
- Teacher network has EMA weights copied from student (prevent collapse).



cls.

Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021.

NYU

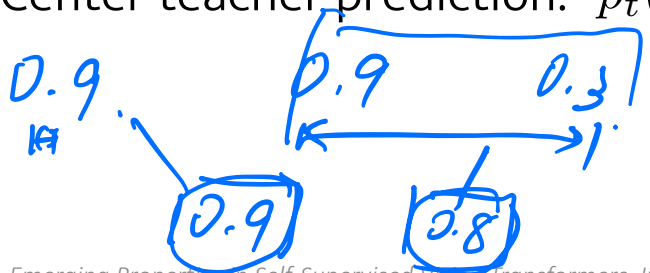


Preventing Collapse

- Cross entropy objective can make both sides collapse to uniform distribution.
  - Apply sharpening, apply a temperature term on both teacher and student.
  - $\mathrm{softmax}(g/\tau)$ The higher the temperature, the more uniform.

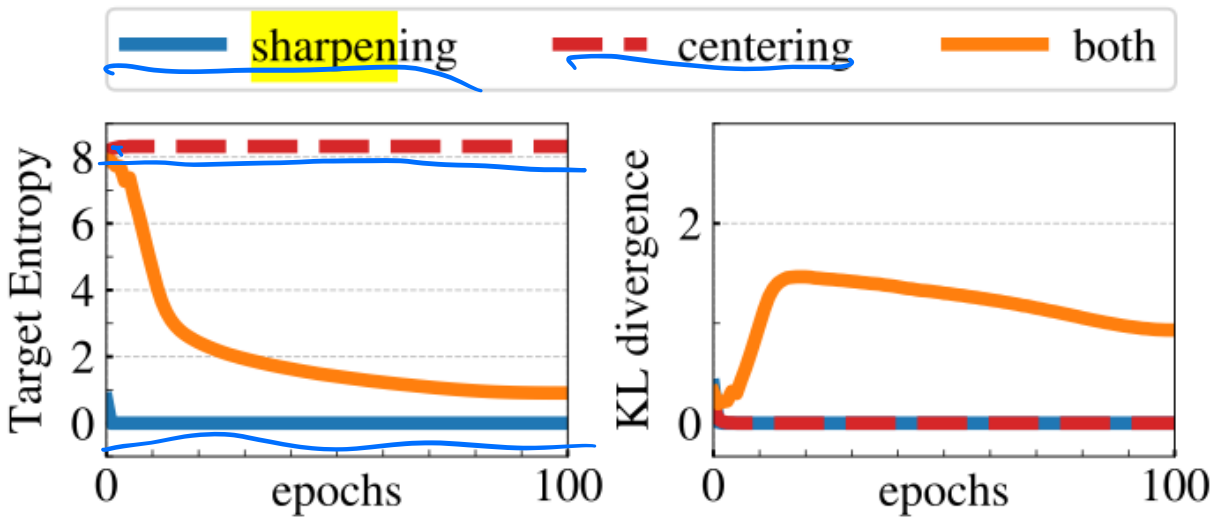
Preventing Collapse

- Cross entropy objective can make both sides collapse to uniform distribution.
  - Apply sharpening, apply a temperature term on both teacher and student.
  - $\mathrm{softmax}(g/\tau)$ The higher the temperature, the more uniform.
- It can also collapse into always activating a single unit.
  - Mean statistics: $c_t = mc_{t-1} + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i)$
  - Center teacher prediction: $p_t(x) = \frac{\exp((g_{\theta_t}(x)_i - c_t)/\tau_t)}{\sum_k \exp((g_{\theta_t}(x)_k - c_t)/\tau_t)}$.



Centering and Sharpening

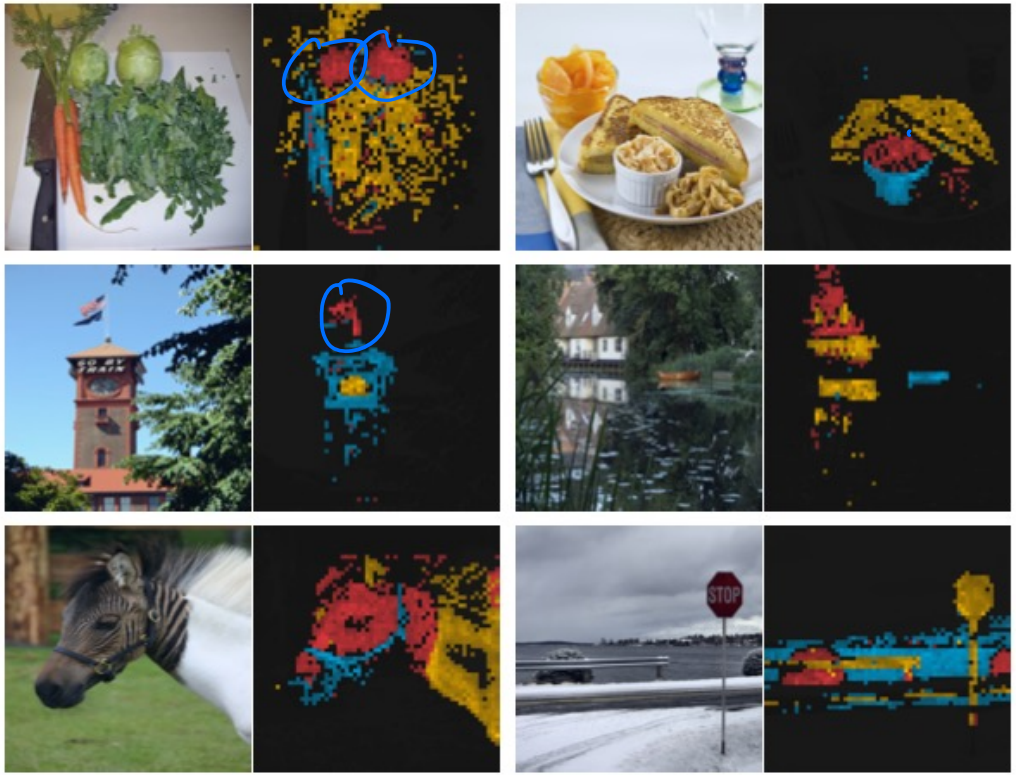
- Only centering: Always uniform distribution, high entropy, easy to guess.
- Only sharpening: Collapsed into one unit, easy to guess, low loss, but no real learning.



NYU

Visualizing Attention

- The [CLS] token is an extra token added to summarize the whole image into a vector.

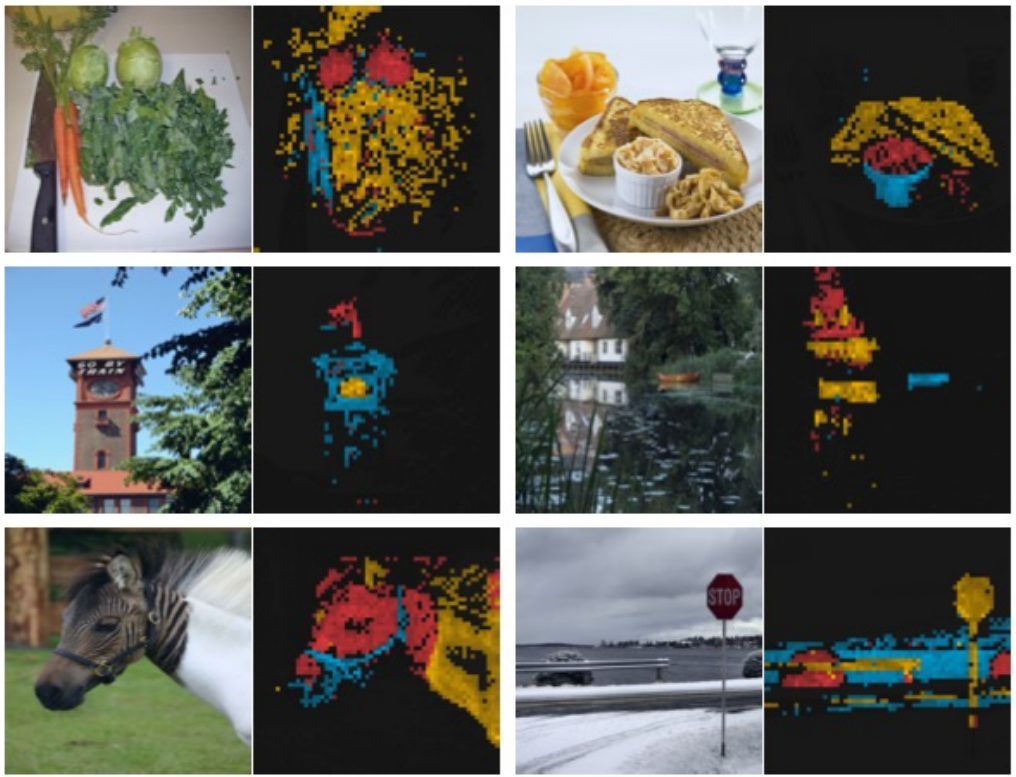


Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021.

NYU

Visualizing Attention

- The [CLS] token is an extra token added to summarize the whole image into a vector.
- Visualize the attention map of different attention heads using different colors.

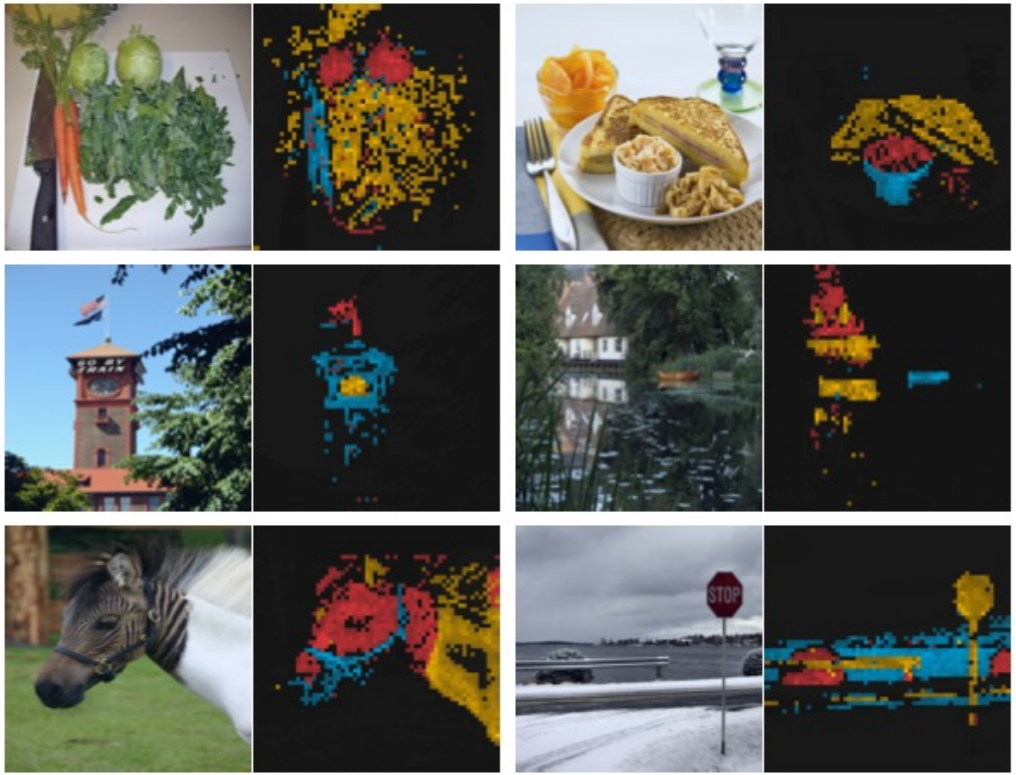


Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021.

NYU

Visualizing Attention

- The [CLS] token is an extra token added to summarize the whole image into a vector.
- Visualize the attention map of different attention heads using different colors.
- Showing understanding of different objects and parts.

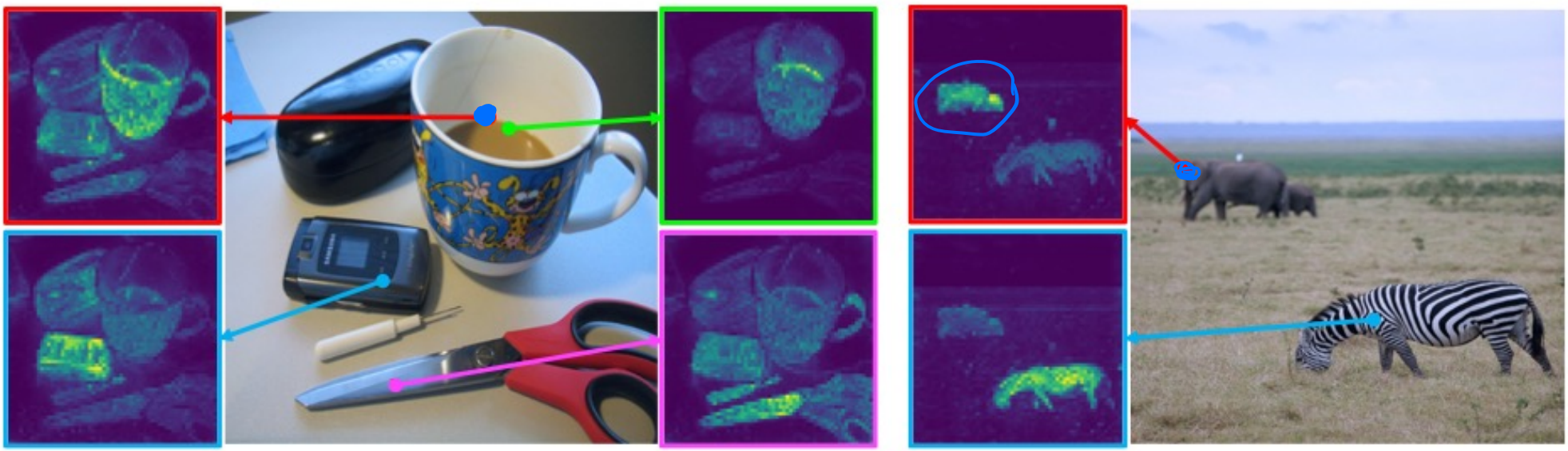


Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021.

NYU

Visualizing Attention

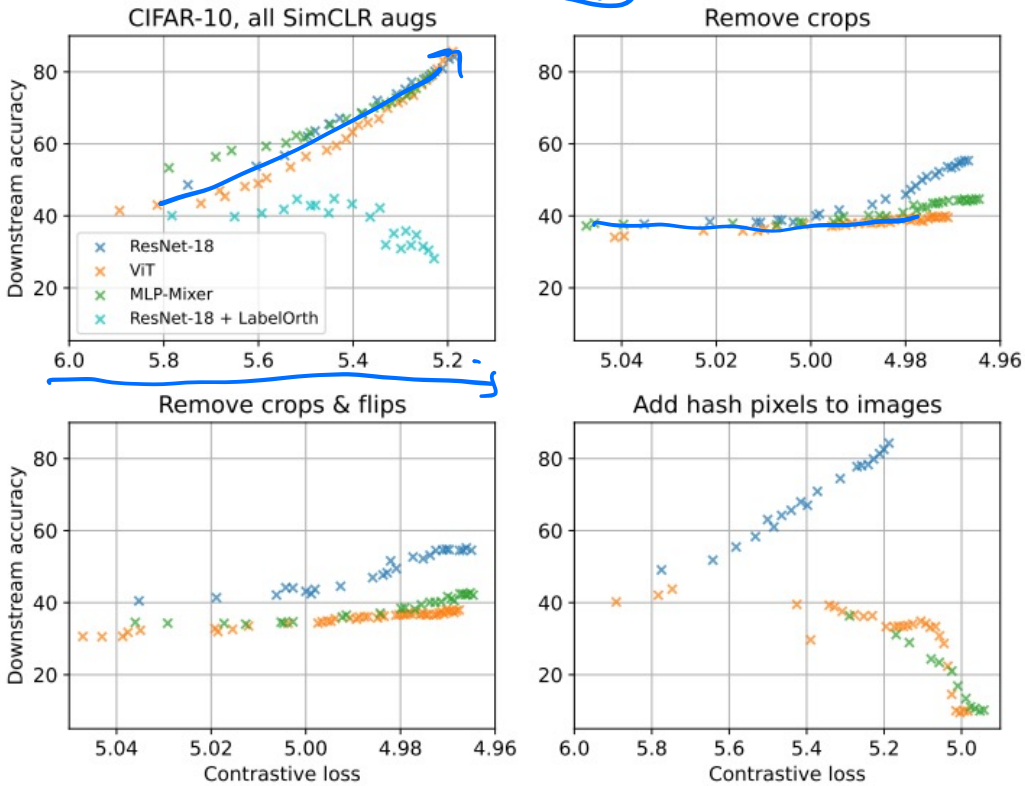
- We can also visualize the attention by querying from a location.
- Weak separation of objects.



NYU

Why Does SSL Work?

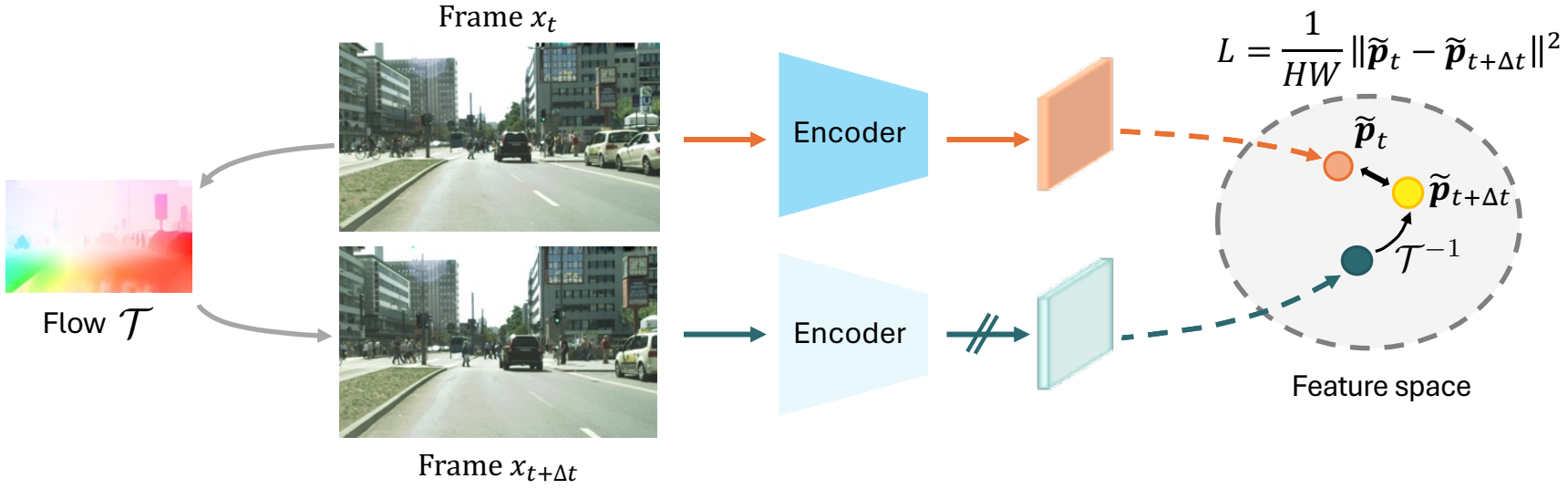
- The unsupervised loss is a surrogate. If an image belongs to a similarity class, it also belongs to the same semantic class.
- The choice of similarity class matters.



Aurora et al. A Theoretical Analysis of Contrastive Unsupervised Representation Learning.
Saunshi et al. Understanding Contrastive Learning Requires Incorporating Inductive Biases. 2022.

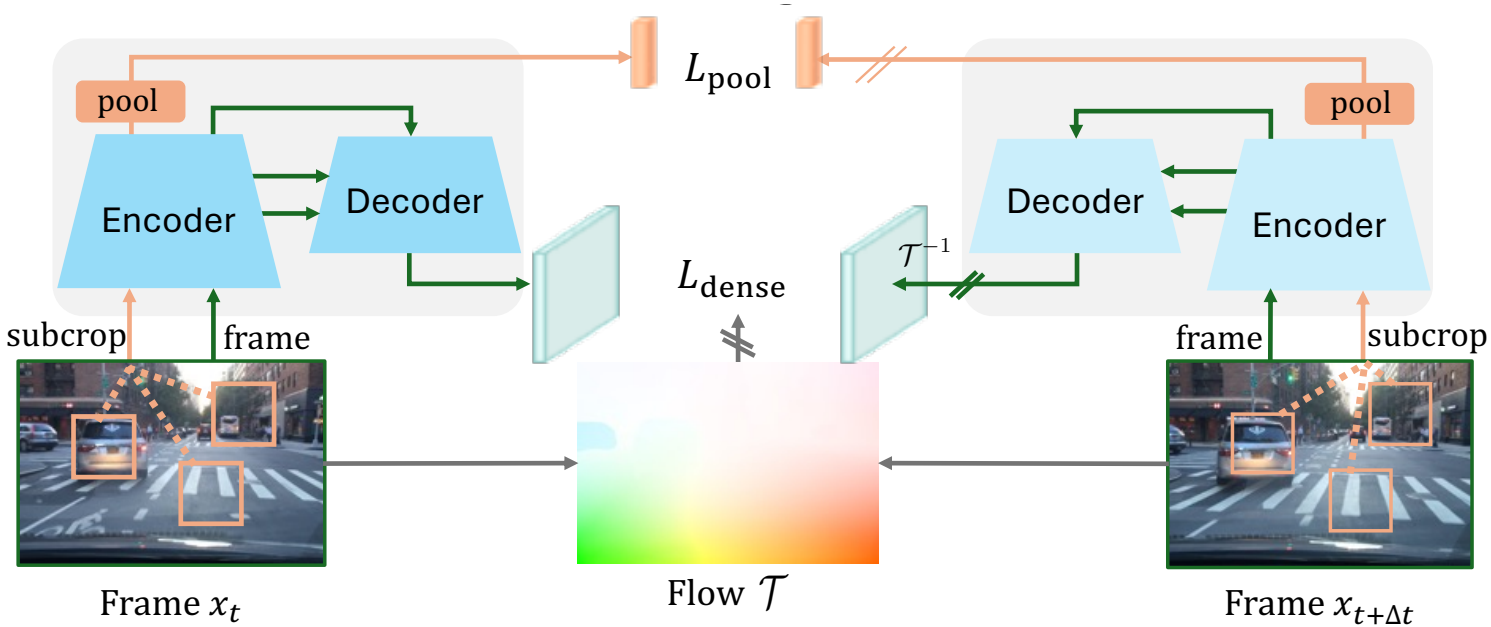
SSL with Motion

- Can we use adjacent frames as self-supervision?
- Objects move densely throughout the image.



SSL with Motion

- Perform SSL in multiple scales (small objects vs. big regions).

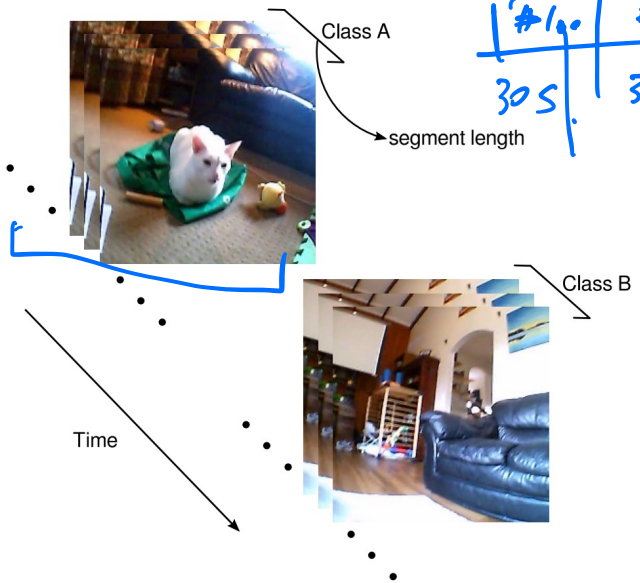


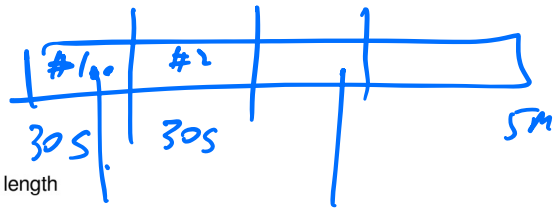
Wang et al. PooDLe: Pooled and Dense Self-Supervised Learning from Naturalistic Videos. ICLR 2025.

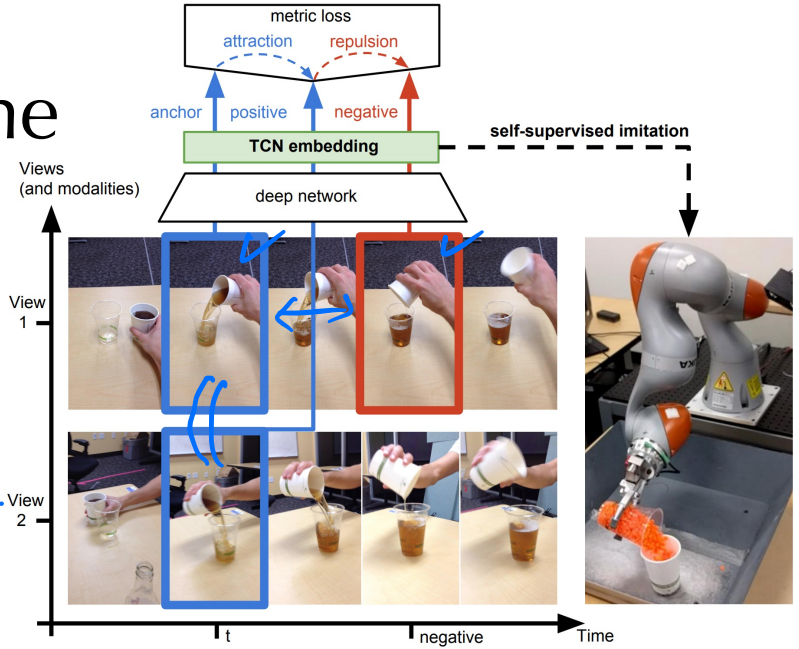
NYU

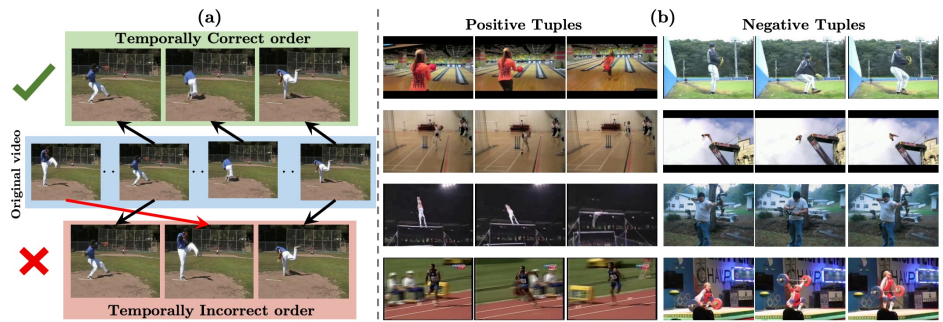
SSL with Time

- Use time as an additional source of supervision.





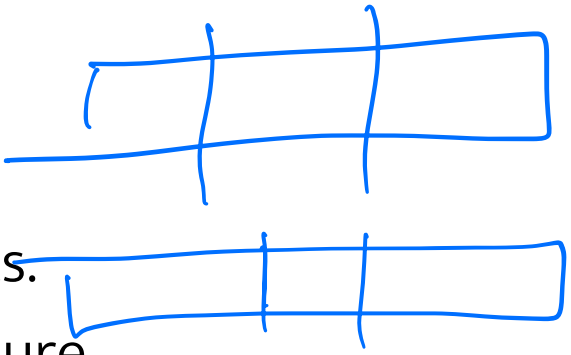




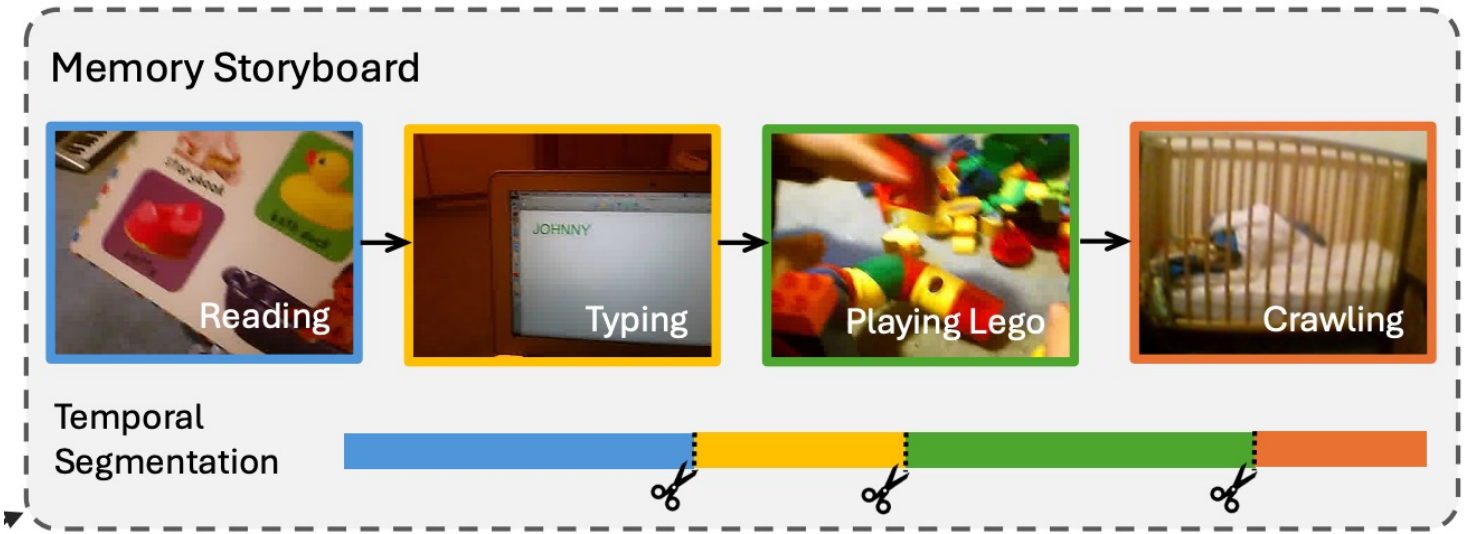
Misra et al. Shuffle and Learn: Unsupervised Learning using Temporal Order Verification. ECCV 2016.
Sermanet et al. Time-Contrastive Networks: Self-Supervised Learning from Video. ICRA 2018.
Orhan et al. Self-Supervised Learning through the Eyes of a Child. NeurIPS 2020.

NYU

SSL with Time



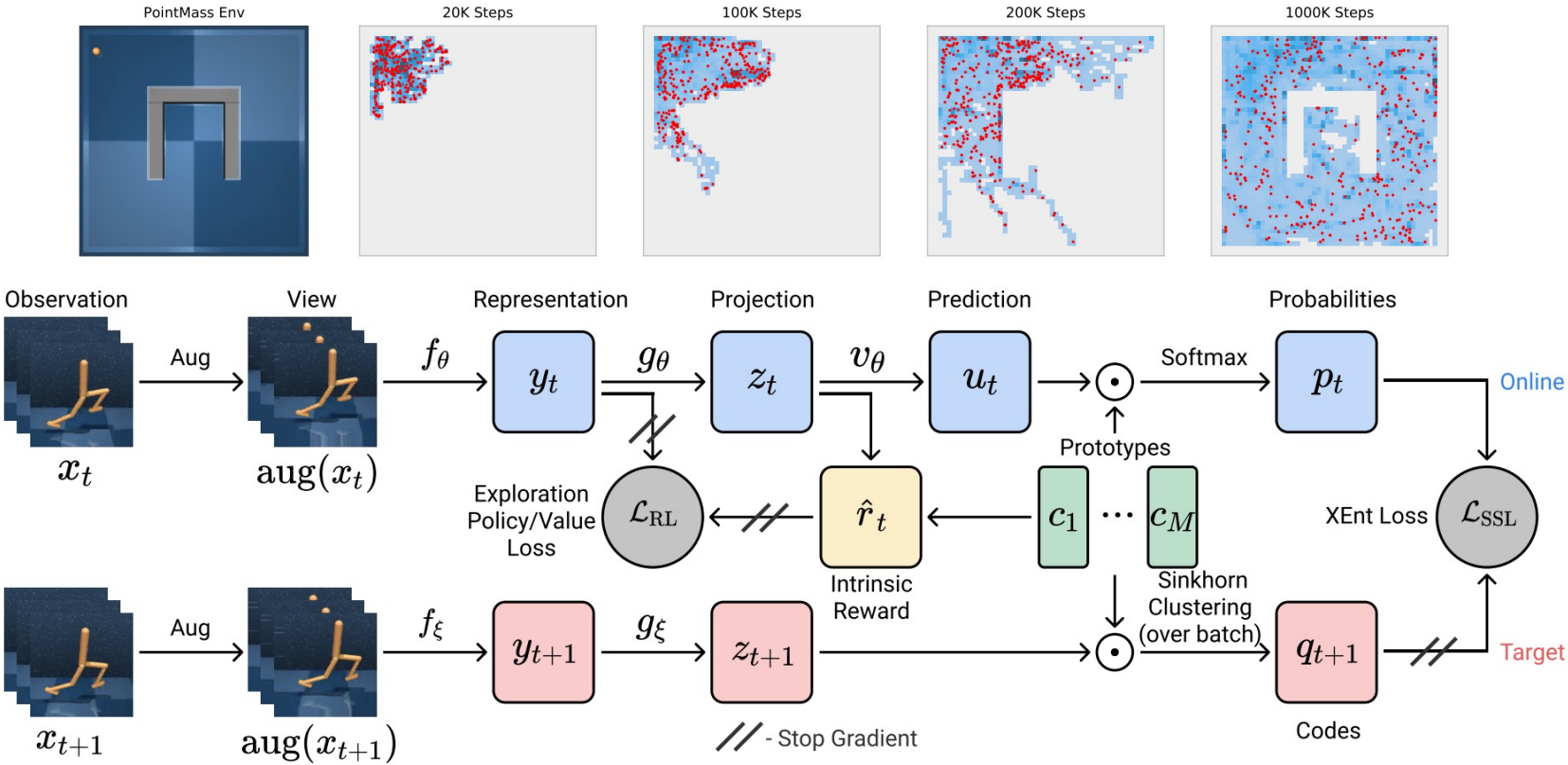
- We can segment videos into meaningful events.
- Leverage the spatiotemporal continuity structure.



Yang & Ren. Memory Storyboard: Leveraging Temporal Segmentation for Streaming Self-Supervised Learning from Egocentric Videos. arXiv 2025.

NYU

SSL for Visual Control

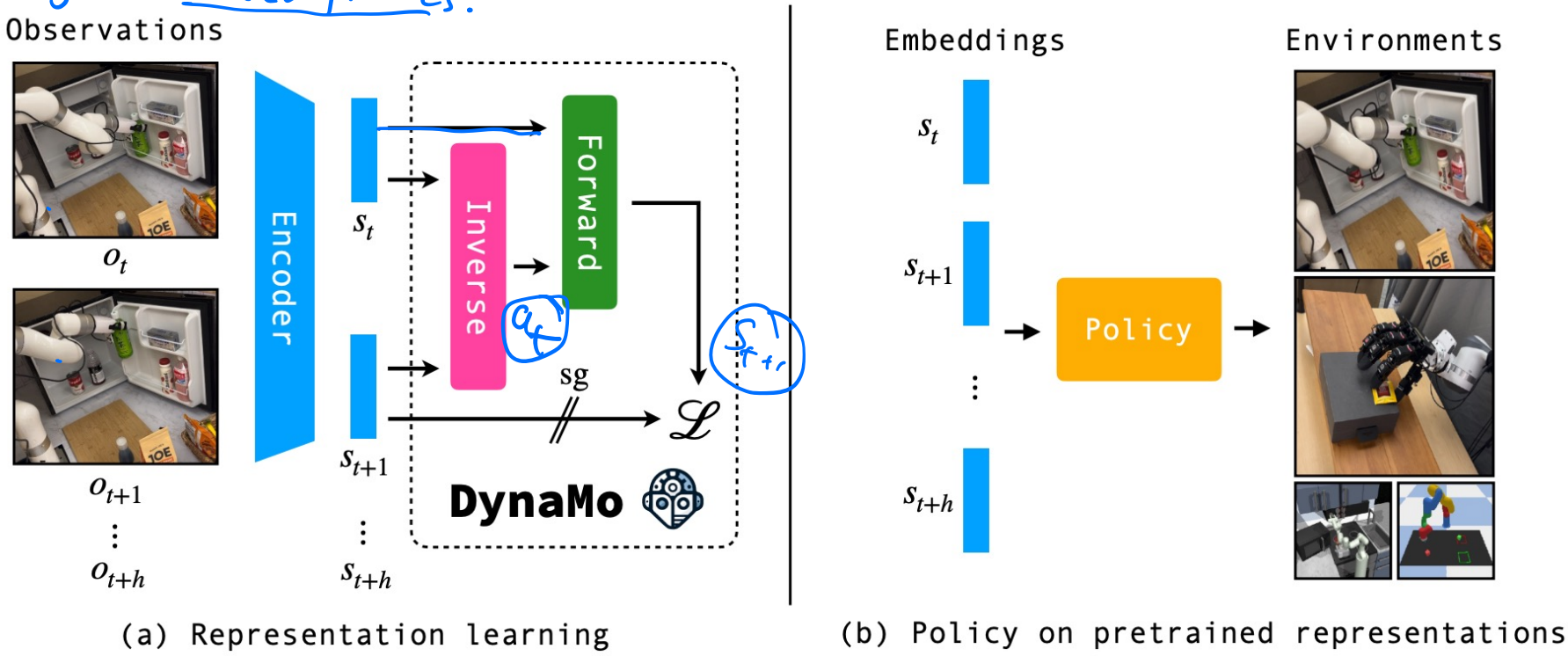


Yarats. Reinforcement Learning with Prototypical Representations. ICML 2021.

NYU

SSL for Visual Control

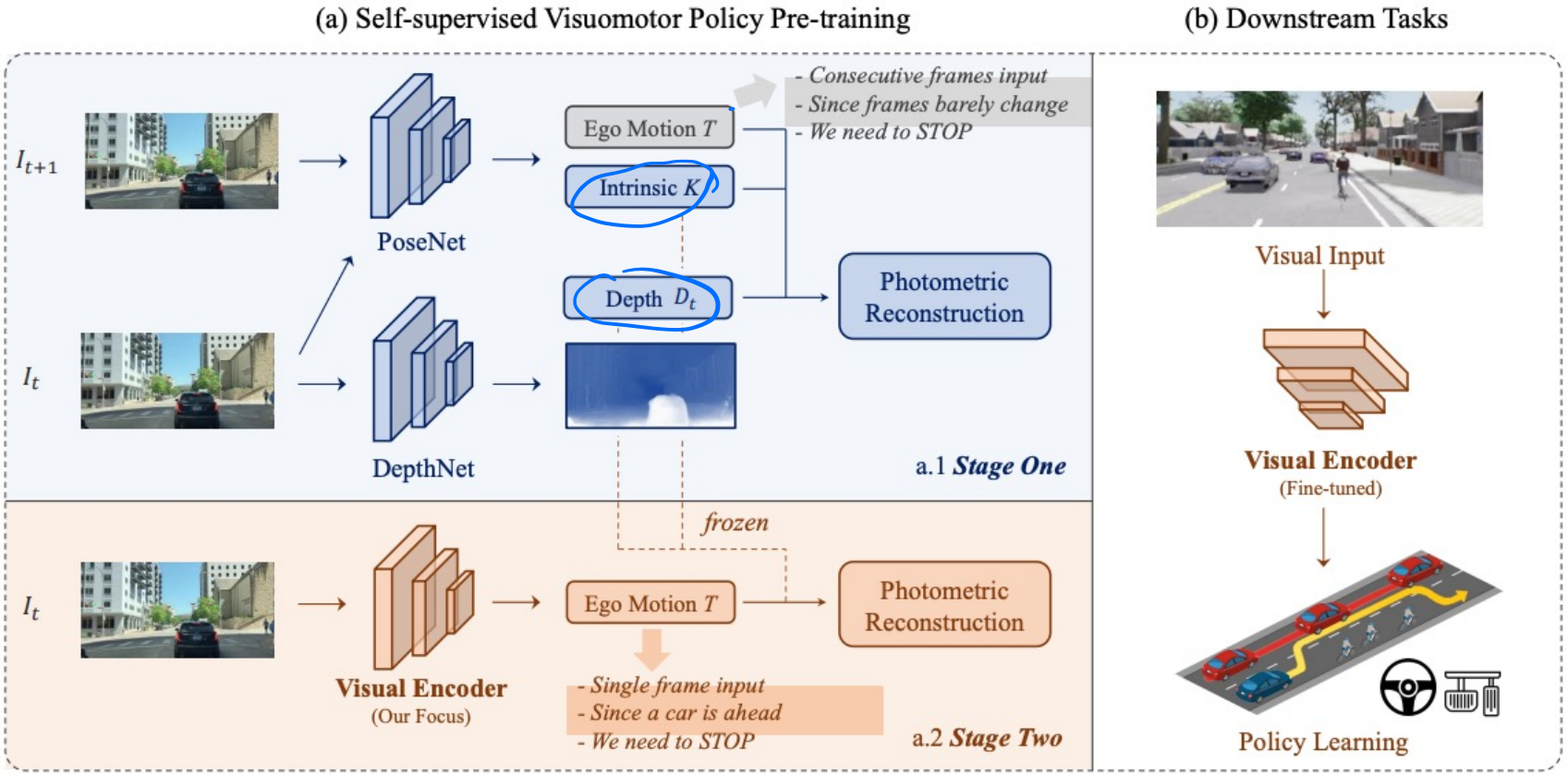
Not augment just video frames.



Cui et al. DynaMo: In-Domain Dynamics Pretraining for Visuo-Motor Control. NeurIPS 2024.

NYU

SSL for Visual Control

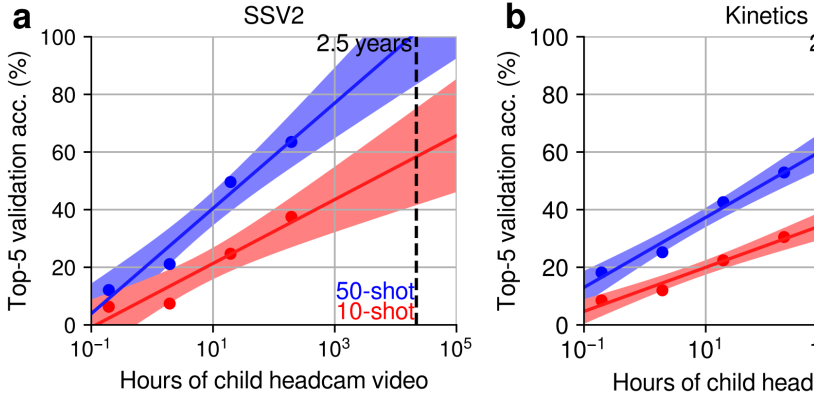


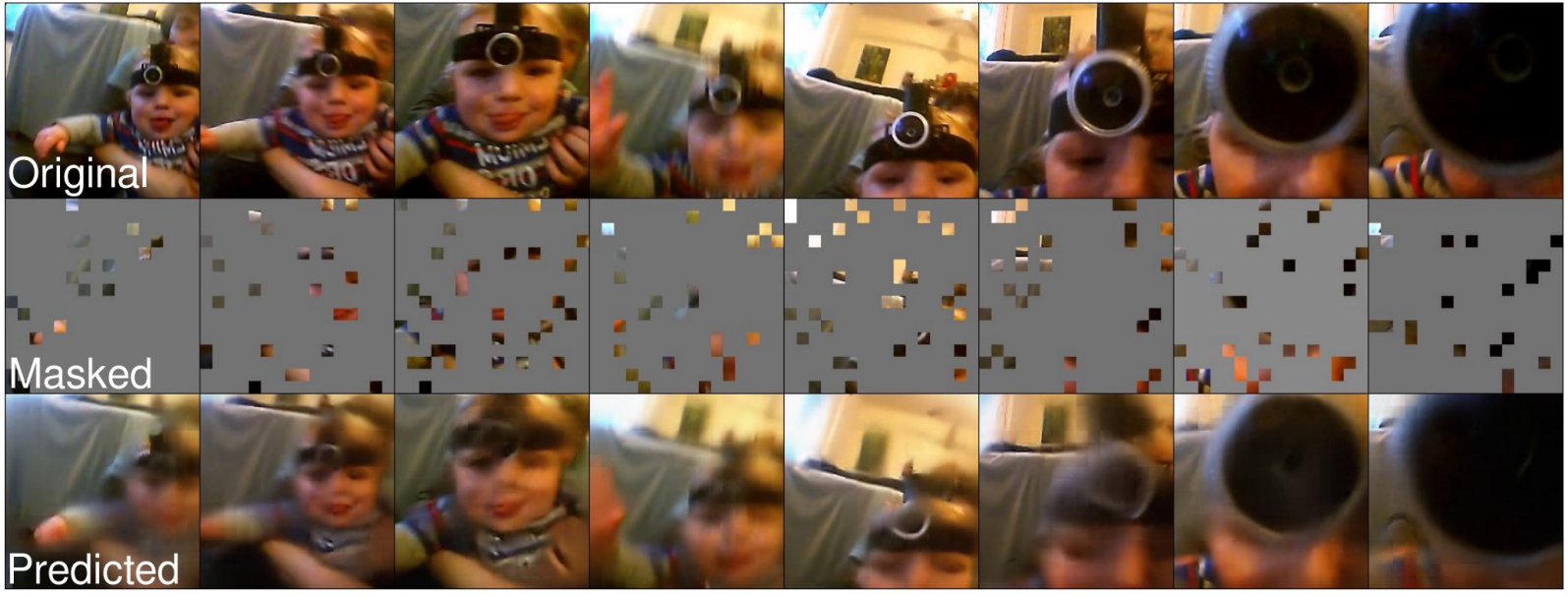
Wu et al. Policy Pre-training for Autonomous Driving via Self-supervised Geometric Modeling. ICLR 2023.

NYU

BabyCam

- Run visual learning algorithms on baby headcam videos.





NYU

Summary

- Representation learning leverage the information in unlabeled data.

NYU

Summary

- Representation learning leverage the information in unlabeled data.
- A foundation for sensorimotor learning.

NYU

Summary

- Representation learning leverage the information in unlabeled data.
- A foundation for sensorimotor learning.
- Inductive biases matter.

NYU

Summary

- Representation learning leverage the information in unlabeled data.
- A foundation for sensorimotor learning.
- Inductive biases matter.
- Possible learning objectives for egocentric videos.

NYU

Summary

- Representation learning leverage the information in unlabeled data.
- A foundation for sensorimotor learning.
- Inductive biases matter.
- Possible learning objectives for egocentric videos.
- Incorporate 3D vision and actions for downstream planning.

NYU

Emergent Attention, Object Discovery

- SSL representations show awareness of object classes and instance identities.

NYU

Emergent Attention, Object Discovery

- SSL representations show awareness of object classes and instance identities.
- Why does attention show awareness of objects?

NYU

Emergent Attention, Object Discovery

- SSL representations show awareness of object classes and instance identities.
- Why does attention show awareness of objects?
- The network is encouraged to associate different parts of the objects together in order to identify whether two inputs belong to the same image or not.

NYU

Emergent Attention, Object Discovery

- SSL representations show awareness of object classes and instance identities.
- Why does attention show awareness of objects?
- The network is encouraged to associate different parts of the objects together in order to identify whether two inputs belong to the same image or not.
- Attending to semantically similar parts facilitates the process.

NYU

Emergent Attention, Object Discovery

- SSL representations show awareness of object classes and instance identities.
- Why does attention show awareness of objects?
- The network is encouraged to associate different parts of the objects together in order to identify whether two inputs belong to the same image or not.
- Attending to semantically similar parts facilitates the process.
- The network is a hierarchical information processing pipeline – Lower layers integrate more granular and smaller neighborhood.

Weak-to-Strong Supervision

- General idea: Use self-supervised learning to learn good features, which allow us to generate low-quality masks.

NYU

Weak-to-Strong Supervision

- General idea: Use self-supervised learning to learn good features, which allow us to generate low-quality masks.
- Then use these masks as pseudo labels and supervise the network to predict these low-quality masks.

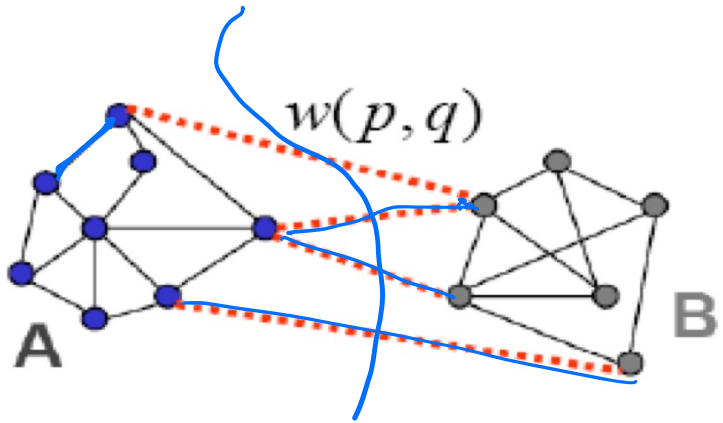
NYU

Weak-to-Strong Supervision

- General idea: Use self-supervised learning to learn good features, which allow us to generate low-quality masks.
- Then use these masks as pseudo labels and supervise the network to predict these low-quality masks.
- Question: how do we come up with masks? What loss is used to supervise the network?

Graph Cut

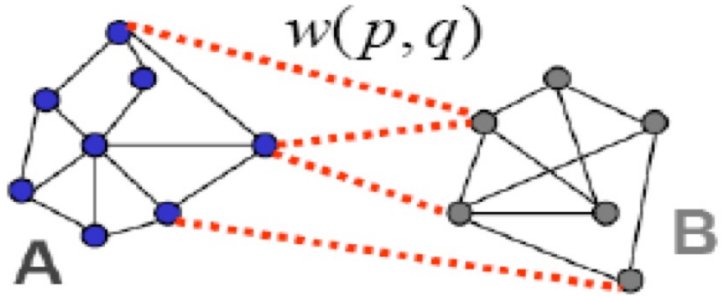
- Segmentation is essentially a clustering problem.



$$cut\,(A, B) = \sum_{p \in A, q \in B} w(p, q)$$

Graph Cut

- Segmentation is essentially a clustering problem.
- We can transform the clustering problem with the graph cut problem.

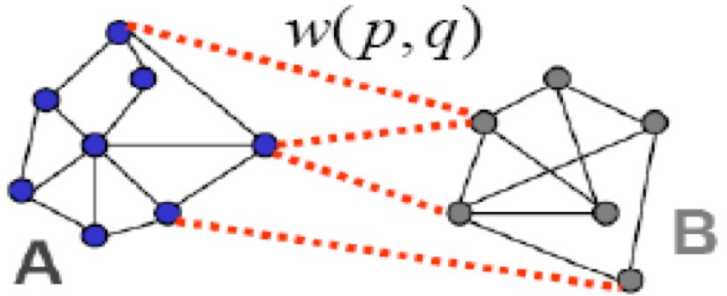


$$cut\,(A, B) = \sum_{p \in A, q \in B} w(p, q)$$

NYU

Graph Cut

- Segmentation is essentially a clustering problem.
- We can transform the clustering problem with the graph cut problem.
- Pixel = node.

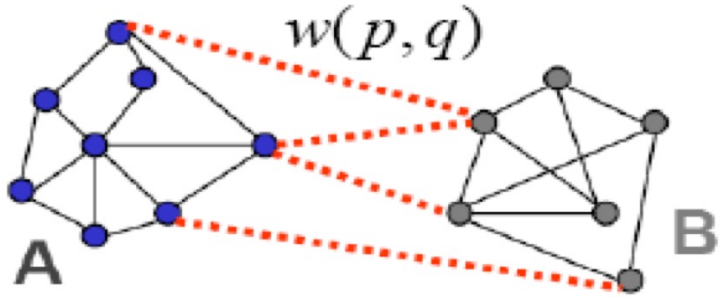


$$cut\,(A, B) = \sum_{p \in A, q \in B} w(p, q)$$

NYU

Graph Cut

- Segmentation is essentially a clustering problem.
- We can transform the clustering problem with the graph cut problem.
- Pixel = node.
- Affinity between the two pixels = edge value (flow).

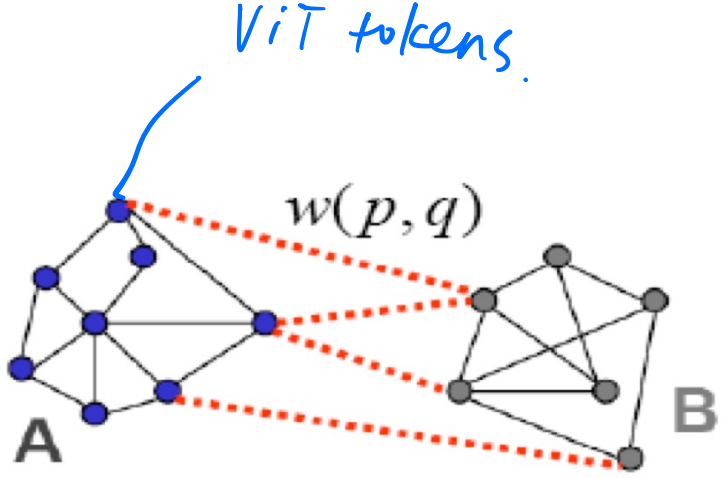


$$cut\,(A, B) = \sum_{p \in A, q \in B} w(p, q)$$

NYU

Graph Cut

- Segmentation is essentially a clustering problem.
- We can transform the clustering problem with the graph cut problem.
- Pixel = node.
- Affinity between the two pixels = edge value (flow).
- Objective: Cut the graph into disconnected components with a minimum sum of edge values.

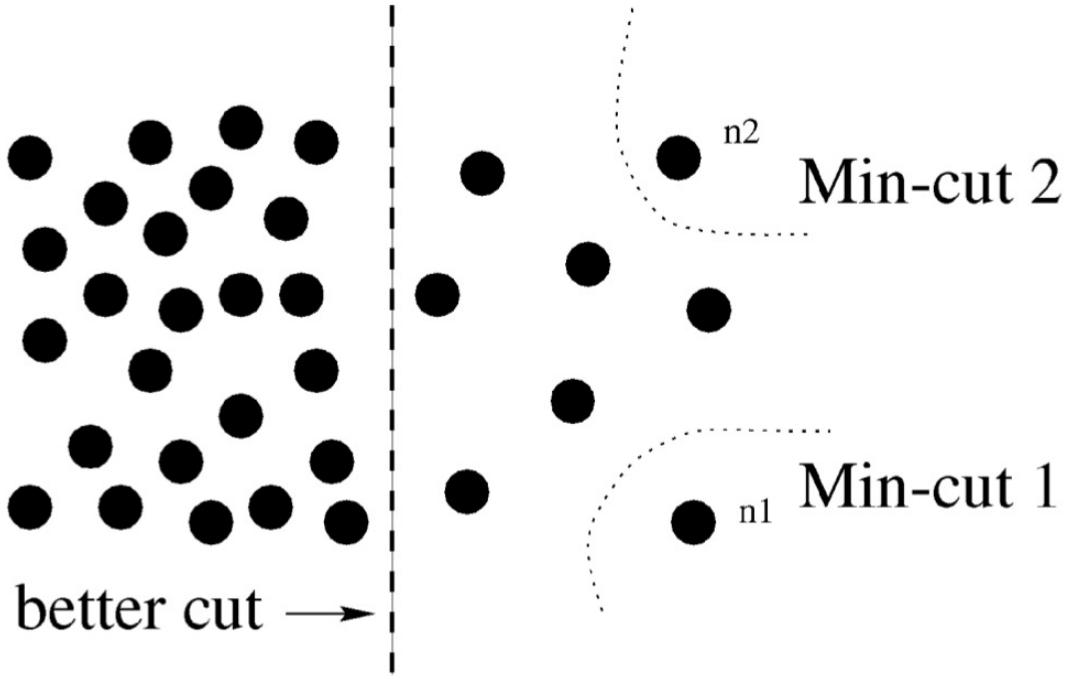


$$cut(A, B) = \sum_{p \in A, q \in B} w(p, q)$$

NYU

Normalized Graph Cut (NCut)

- How to prevent cutting small isolated nodes?



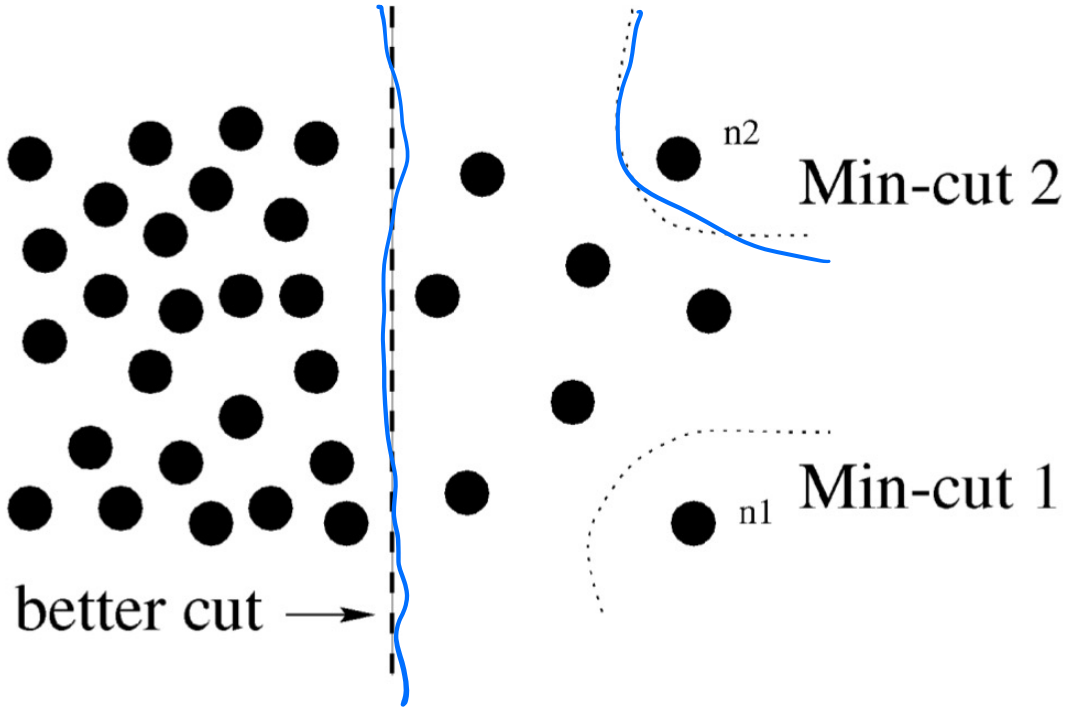
Shi and Malik. Normalized Cuts and Image Segmentation. TPAMI 2000.

NYU

Normalized Graph Cut (NCut)

- How to prevent cutting small isolated nodes?
- Normalize by the total edge connections of a group to all the nodes.

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$



Shi and Malik. Normalized Cuts and Image Segmentation. TPAMI 2000.

NYU

NCut Details (Optional)

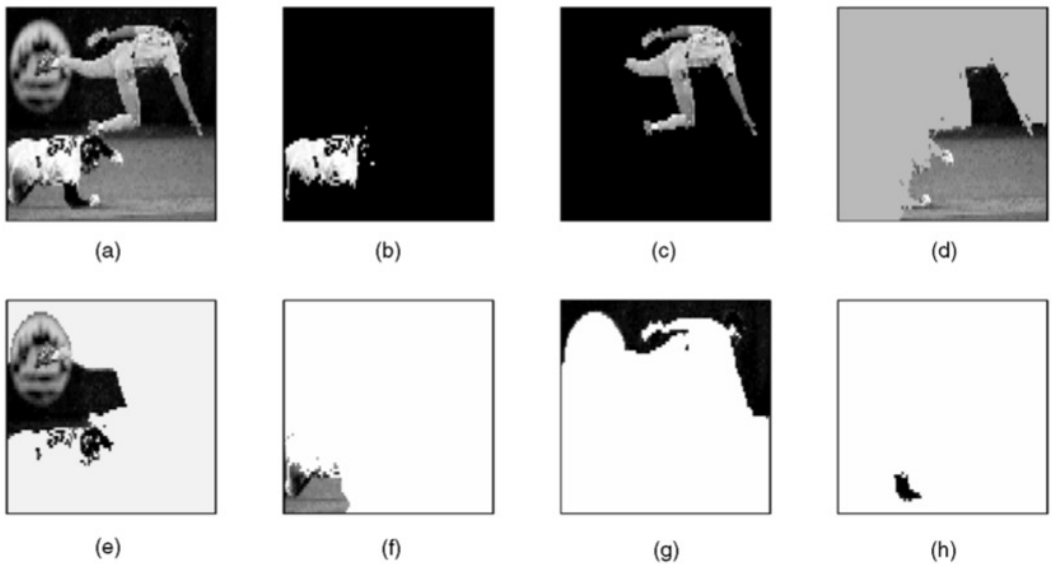
- A form of spectral clustering.
- Degree matrix D $N \times N$ with d_i on the diagonal.
- Weight matrix W $N \times N$ symmetric w_{ij}.
- Selection vector $x_i = 1$ if $i \in A$ otherwise -1.
- Solve the minimization: $\min_y \frac{y^\top (D-W)y}{y^\top Dy}$ $y = (1+x) - \frac{\sum_{i|x_i>0} d_i}{\sum_{i|x_i<0} d_i}(1-x)$.
- Generalized eigenvalue system: $(D-W)y = \lambda Dy$.
- Let $z = D^{1/2}y$ $D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}z = \lambda z$.

Shi and Malik. Normalized Cuts and Image Segmentation. TPAMI 2000.

NYU

NCut

- Sort the eigenvectors from the smallest to the largest.

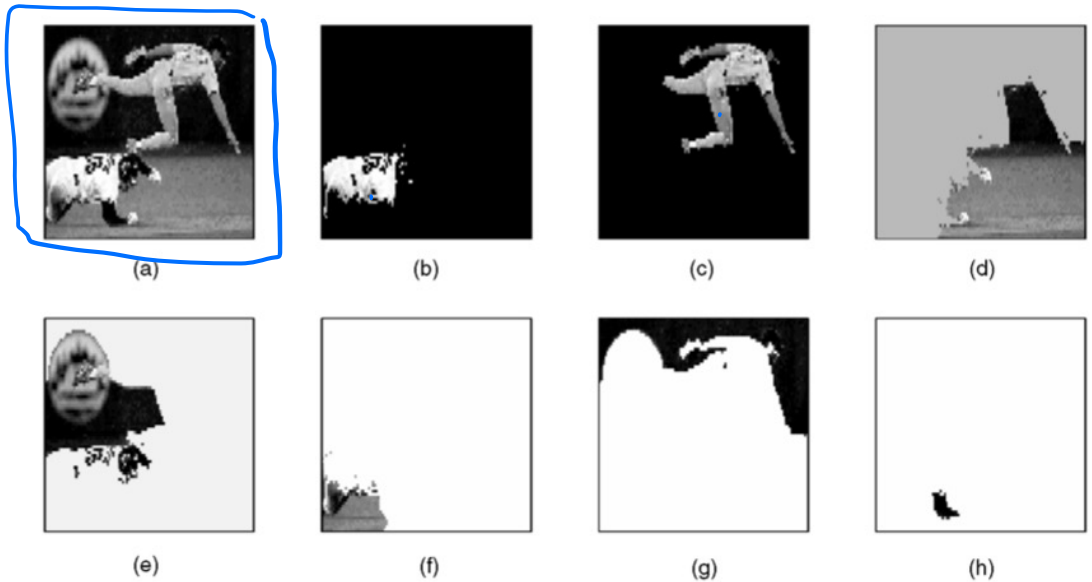


Shi and Malik. Normalized Cuts and Image Segmentation. TPAMI 2000.

NYU

NCut

- Sort the eigenvectors from the smallest to the largest.
- This was a classic image segmentation technique operating directly on image intensity.

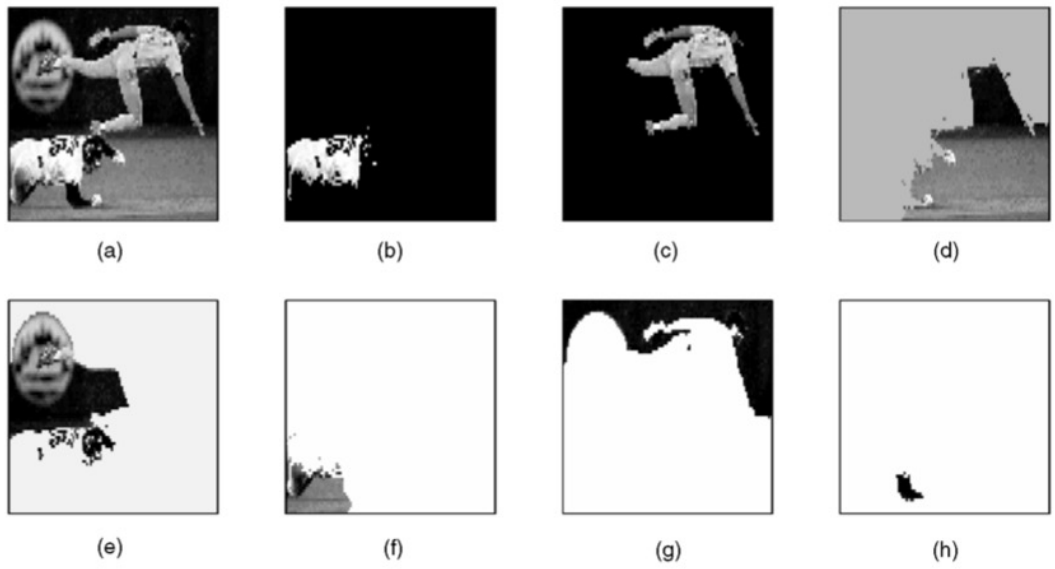


Shi and Malik. Normalized Cuts and Image Segmentation. TPAMI 2000.

NYU

NCut

- Sort the eigenvectors from the smallest to the largest.
- This was a classic image segmentation technique operating directly on image intensity.
- Now, instead of segmenting pixels, we can directly segment semantically meaningful representations from self-supervision.

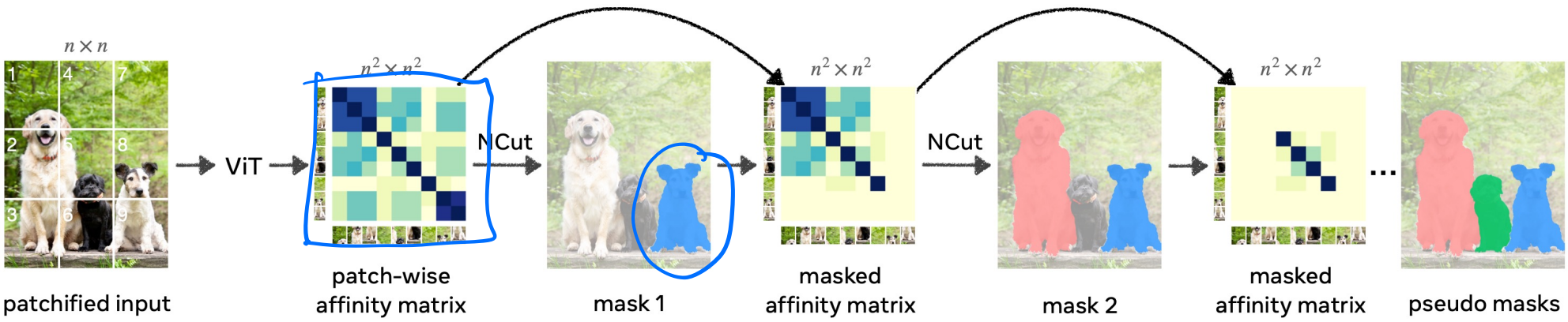


Shi and Malik. Normalized Cuts and Image Segmentation. TPAMI 2000.

NYU

MaskCut

- Use a pretrained DINO ViT network.

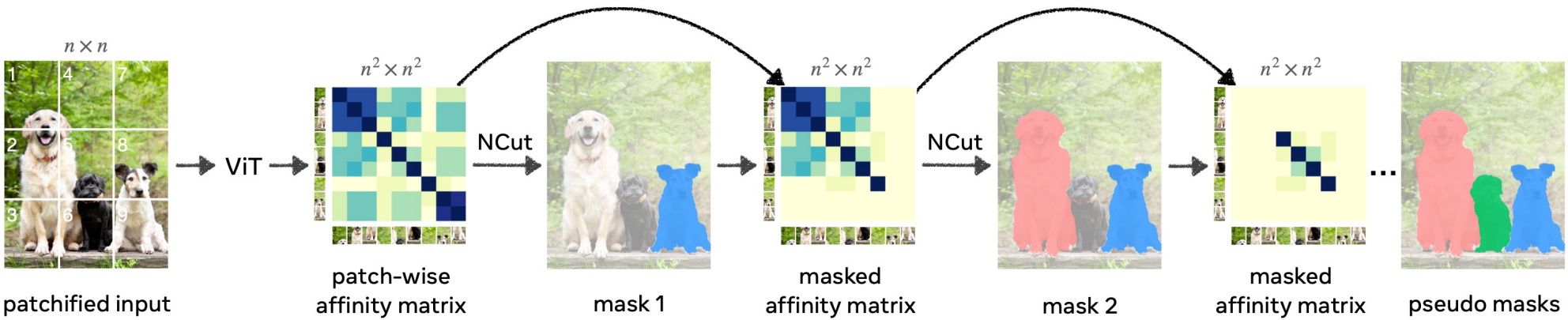


Wang et al. Self-supervised transformers for unsupervised object discovery using normalized cut. CVPR 2022.
Wang et al. Cut and Learn for Unsupervised Object Detection and Instance Segmentation. CVPR 2023.

NYU

MaskCut

- Use a pretrained DINO ViT network.
- Use the "key" features from the last attention layer: $W_{ij} = \frac{K_i K_j}{\|K_i\|_2 \|K_j\|_2}$

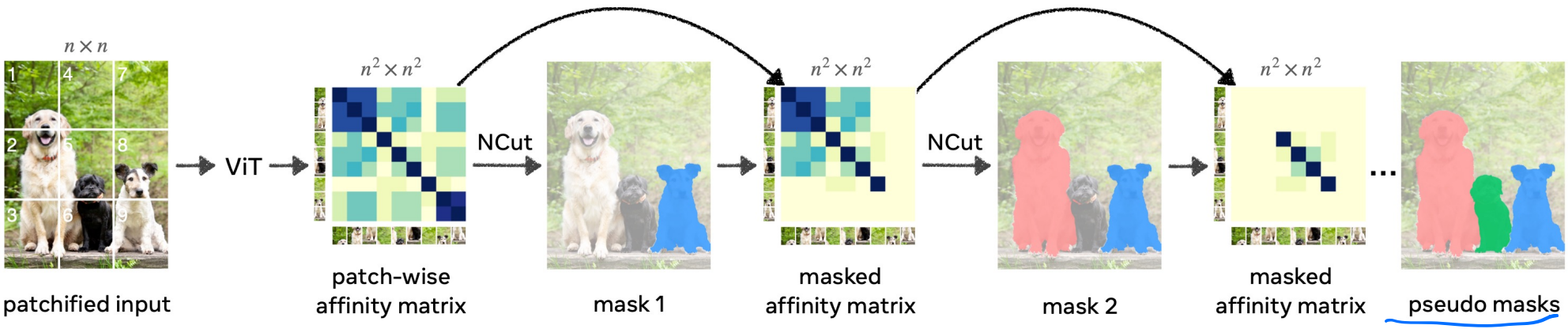


Wang et al. Self-supervised transformers for unsupervised object discovery using normalized cut. CVPR 2022.
Wang et al. Cut and Learn for Unsupervised Object Detection and Instance Segmentation. CVPR 2023.

NYU

MaskCut

- Use a pretrained DINO ViT network.
- Use the "key" features from the last attention layer: $W_{ij} = \frac{K_i K_j}{\|K_i\|_2 \|K_j\|_2}$
- Iterative NCut on the pairwise matrix by masking out the regions from previous stages.

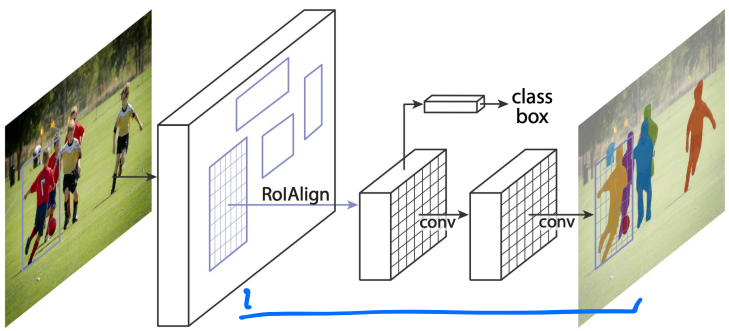


Wang et al. Self-supervised transformers for unsupervised object discovery using normalized cut. CVPR 2022.
Wang et al. Cut and Learn for Unsupervised Object Detection and Instance Segmentation. CVPR 2023.

NYU

Iterative Self-Training

- Now add a MaskRCNN structure on top of the pretrained network.

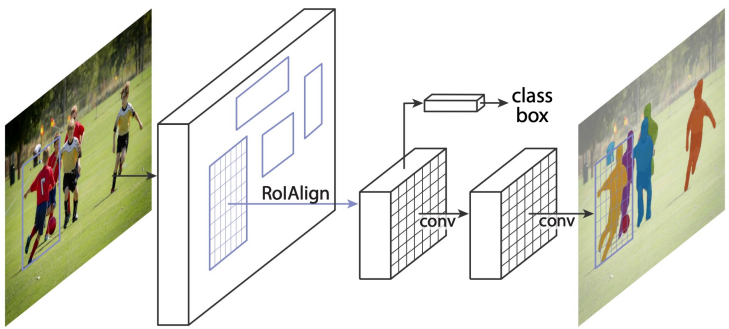


Wang et al. Cut and Learn for Unsupervised Object Detection and Instance Segmentation. CVPR 2023.

NYU

Iterative Self-Training

- Now add a MaskRCNN structure on top of the pretrained network.
- Select the predictions with the highest confidence score and use them as labels.

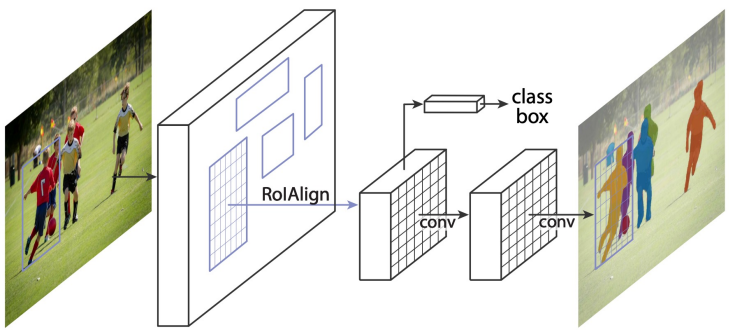


Wang et al. Cut and Learn for Unsupervised Object Detection and Instance Segmentation. CVPR 2023.

NYU

Iterative Self-Training

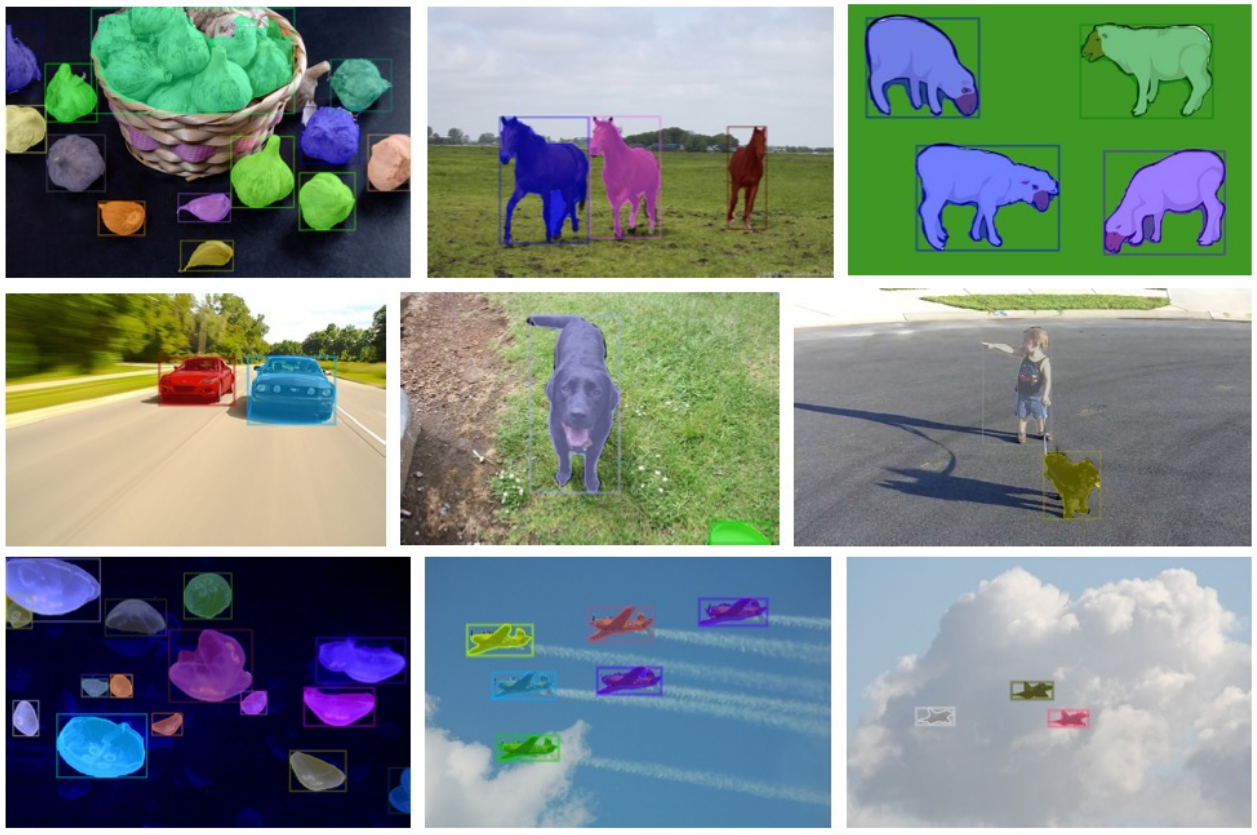
- Now add a MaskRCNN structure on top of the pretrained network.
- Select the predictions with the highest confidence score and use them as labels.
- Neural networks can learn from the noisy labels and output smoother predictions.



Wang et al. Cut and Learn for Unsupervised Object Detection and Instance Segmentation. CVPR 2023.

NYU

More Visualization

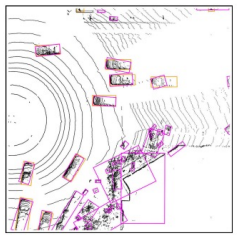


Wang et al. Cut and Learn for Unsupervised Object Detection and Instance Segmentation. CVPR 2023.

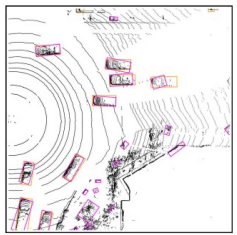
NYU

Pseudo Labels in 3D

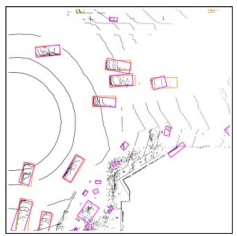
1 Point clustering pseudo-labels



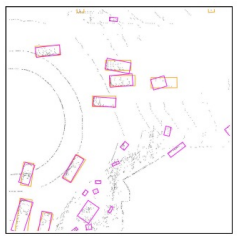
2 Filter out temporally inconsistent tracks



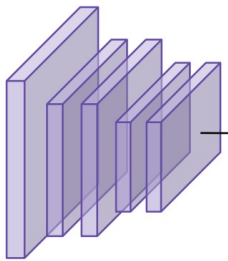
3 Randomly drop lidar beams



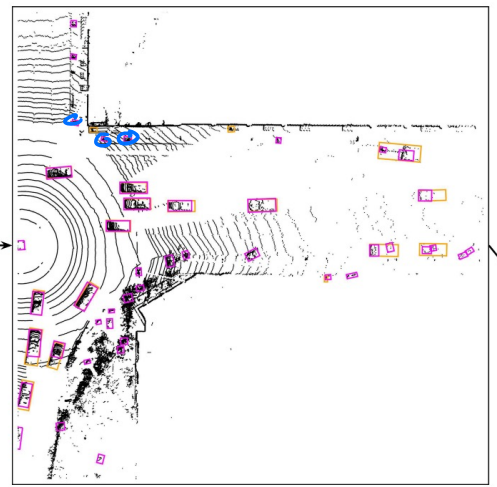
4 Randomly drop spherical rows/cols



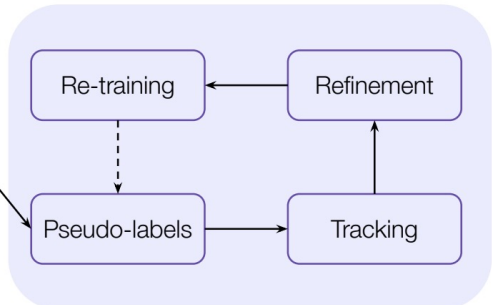
5 Train CNN in short range



6 Zero-shot generalization to long-range



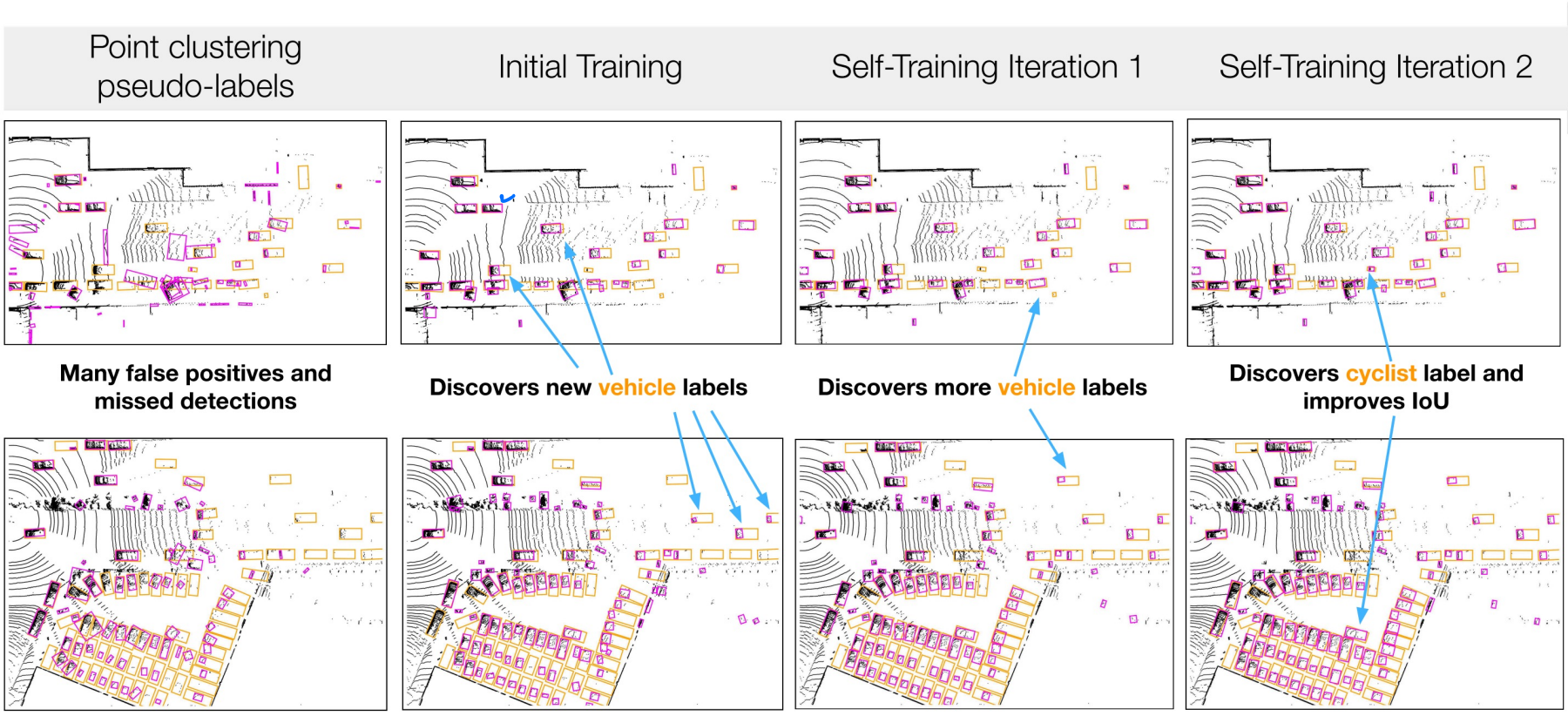
7 Self-training in long-range



Zhang et al. Towards Unsupervised Object Detection from LiDAR Point Clouds. CVPR 2023.

NYU

Iterative Refinement of Pseudo Labels

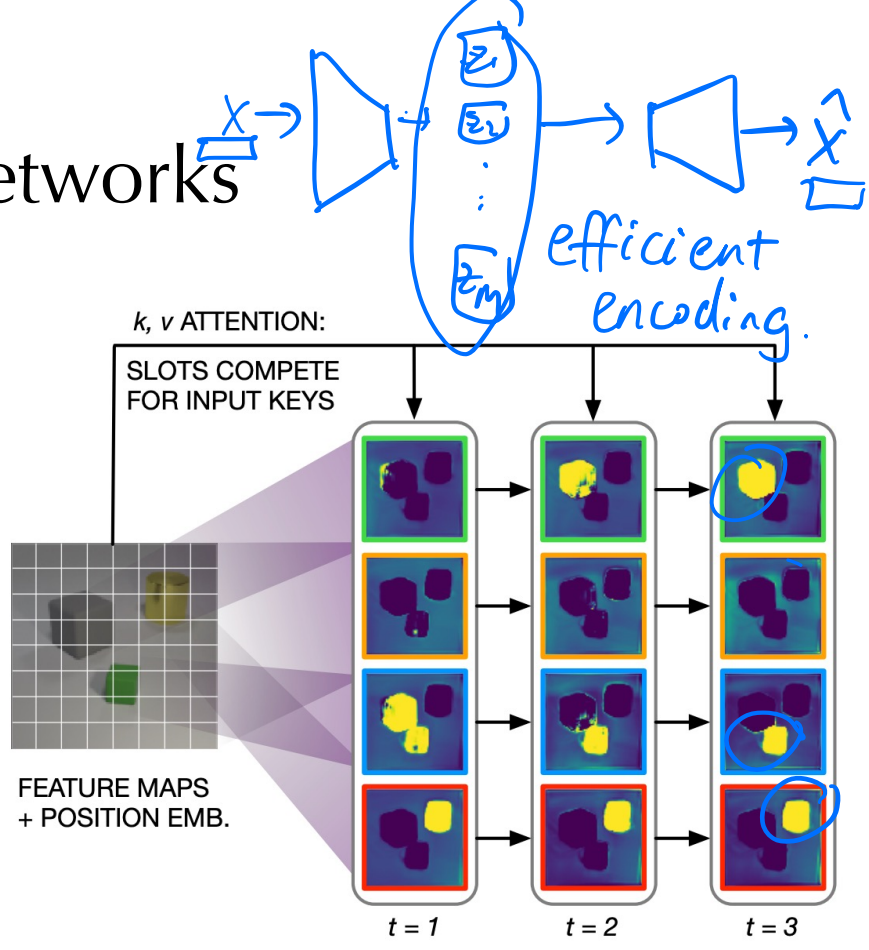


Zhang et al. Towards Unsupervised Object Detection from LiDAR Point Clouds. CVPR 2023.

NYU

Slot Attention Networks

- Can we learn clustering as an end-to-end operation?

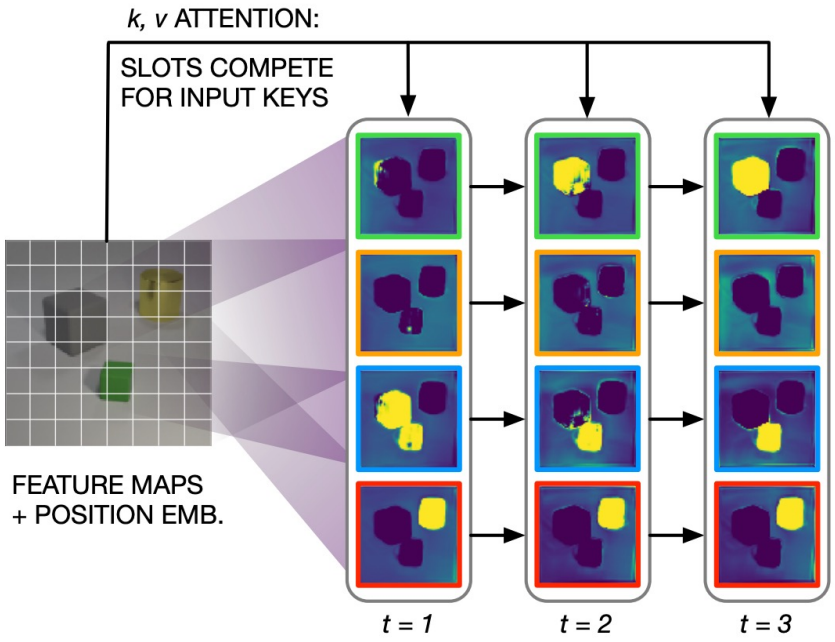


(a) Slot Attention module.

NYU

Slot Attention Networks

- Can we learn clustering as an end-to-end operation?
- Slot attention is inspired by the success of the attention mechanism.



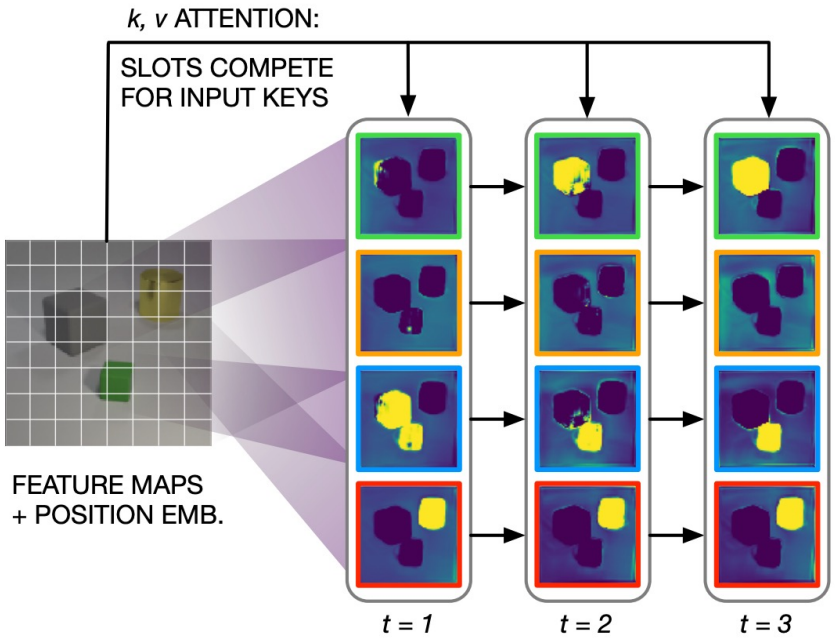
(a) Slot Attention module.

Locatello et al. Object-Centric Learning with Slot Attention. NeurIPS 2020.

NYU

Slot Attention Networks

- Can we learn clustering as an end-to-end operation?
- Slot attention is inspired by the success of the attention mechanism.
- Each “slot” attends to a region of the image and stores an object centric representation.



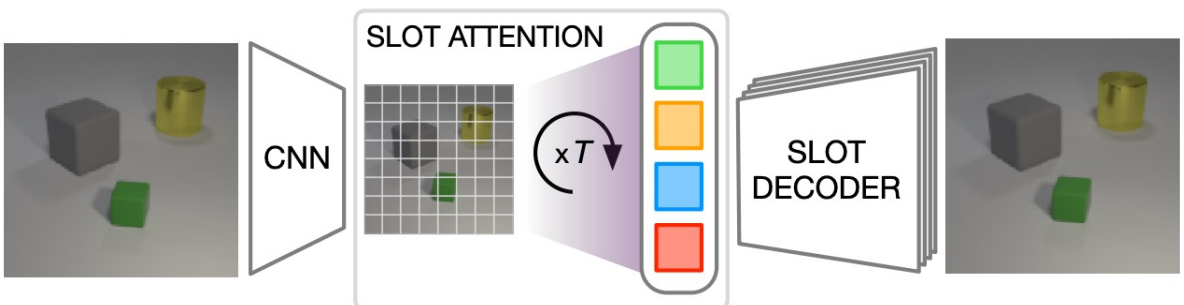
(a) Slot Attention module.

Locatello et al. Object-Centric Learning with Slot Attention. NeurIPS 2020.

NYU

Slot Attention Networks

- Goal: Reconstruct the image with a concise slot-based representation.

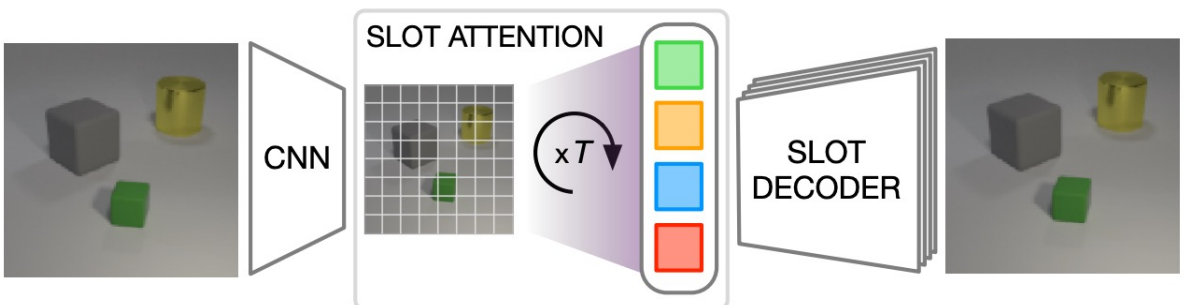


Locatello et al. Object-Centric Learning with Slot Attention. NeurIPS 2020.

NYU

Slot Attention Networks

- Goal: Reconstruct the image with a concise slot-based representation.
- Input: $x \in \mathbb{R}^{N \times D}$ (after encoder), Slots: $m \in \mathbb{R}^{M \times D}$. Normalize: $\tilde{m}_{t-1} = LN(m_{t-1})$.

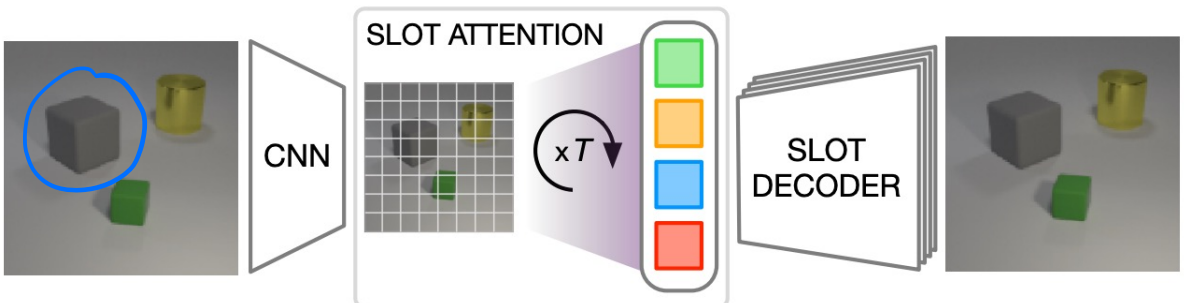


Locatello et al. Object-Centric Learning with Slot Attention. NeurIPS 2020.

NYU

Slot Attention Networks

- Goal: Reconstruct the image with a concise slot-based representation.
- Input: $x \in \mathbb{R}^{N \times D}$ (after encoder), Slots: $m \in \mathbb{R}^{M \times D}$. Normalize: $\tilde{m}_{t-1} = LN(m_{t-1})$.
- Attention over slots: $a_{t,i,j} = \frac{\frac{1}{\sqrt{D}} k(x_i) \cdot q(\tilde{m}_j)^\top}{\sum_j \frac{1}{\sqrt{D}} k(x_i) \cdot q(\tilde{m}_j)^\top}$.

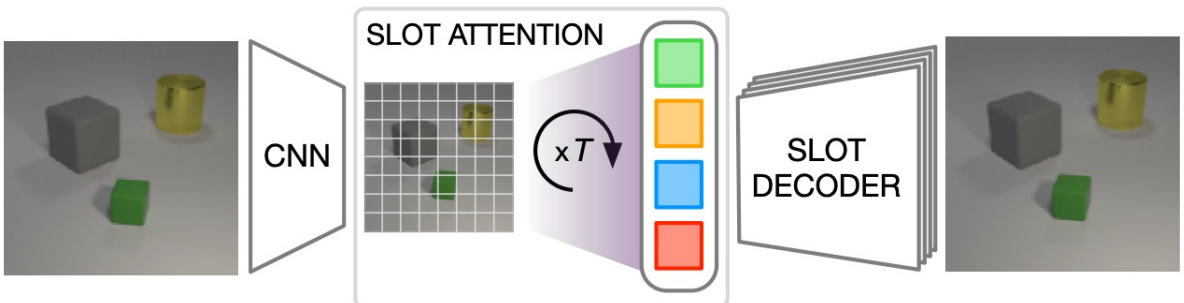


Locatello et al. Object-Centric Learning with Slot Attention. NeurIPS 2020.

NYU

Slot Attention Networks

- Goal: Reconstruct the image with a concise slot-based representation.
- Input: $x \in \mathbb{R}^{N \times D}$ (after encoder), Slots: $m \in \mathbb{R}^{M \times D}$. Normalize: $\tilde{m}_{t-1} = LN(m_{t-1})$.
- Attention over slots: $a_{t,i,j} = \frac{\frac{1}{\sqrt{D}} k(x_i) \cdot q(\tilde{m}_j)^\top}{\sum_j \frac{1}{\sqrt{D}} k(x_i) \cdot q(\tilde{m}_j)^\top}$.
- Updates: $u_{tj} = \sum_i a_{tij} v(x_i)$.



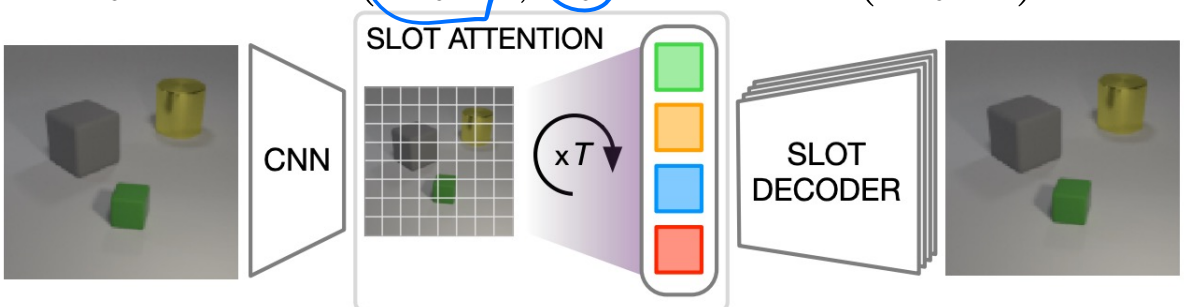
Locatello et al. Object-Centric Learning with Slot Attention. NeurIPS 2020.

NYU

Slot Attention Networks

- Goal: Reconstruct the image with a concise slot-based representation.
- Input: $x \in \mathbb{R}^{N \times D}$ (after encoder), Slots: $m \in \mathbb{R}^{M \times D}$. Normalize: $\tilde{m}_{t-1} = LN(m_{t-1})$.
- Attention over slots: $a_{t,i,j} = \frac{\frac{1}{\sqrt{D}} k(x_i) \cdot q(\tilde{m}_j)^\top}{\sum_j \frac{1}{\sqrt{D}} k(x_i) \cdot q(\tilde{m}_j)^\top}$.
- Updates: $u_{tj} = \sum_i a_{tij} v(x_i)$.
- Write into slots: $m_t = GRU(m_{t-1}, u_t) + MLP(\tilde{m}_{t-1})$.

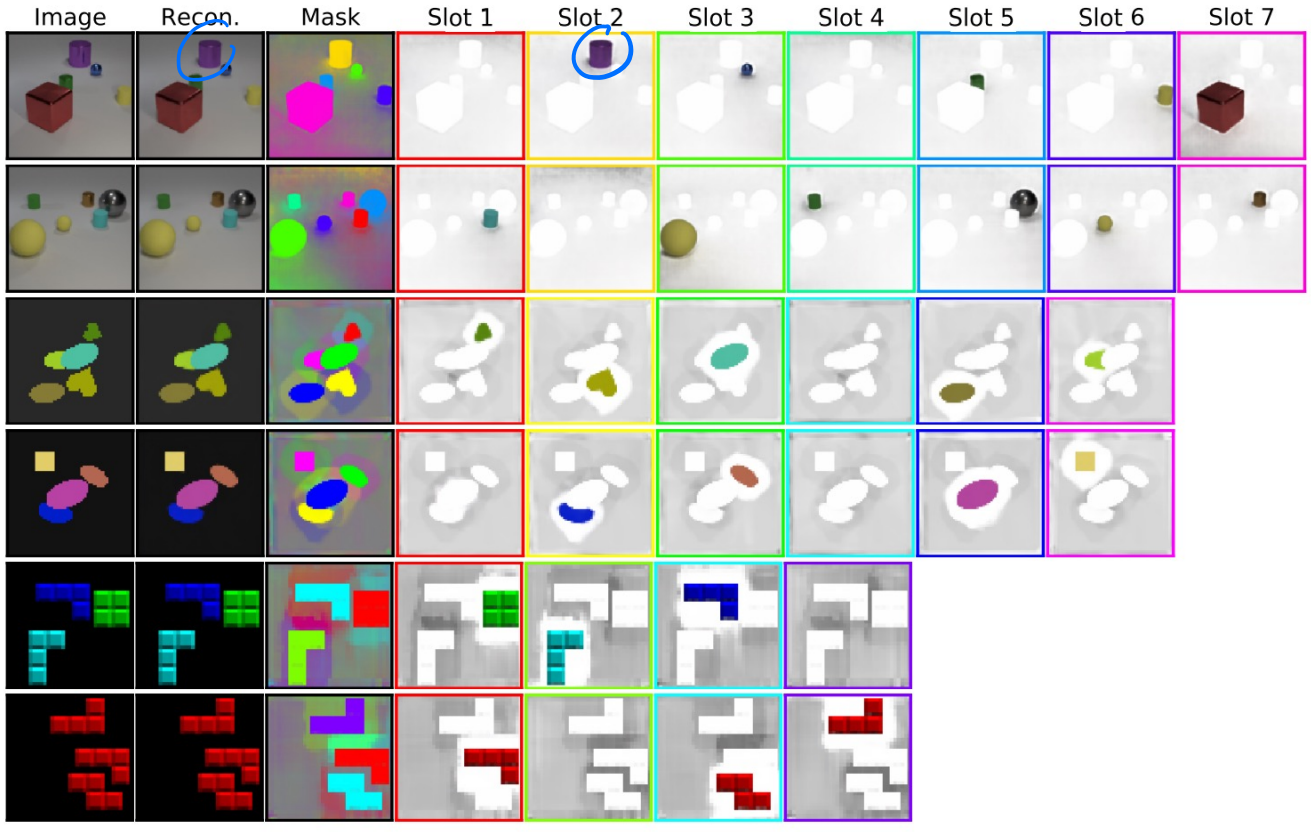
prev mem updates



Locatello et al. Object-Centric Learning with Slot Attention. NeurIPS 2020.

NYU

Slot Attention Networks

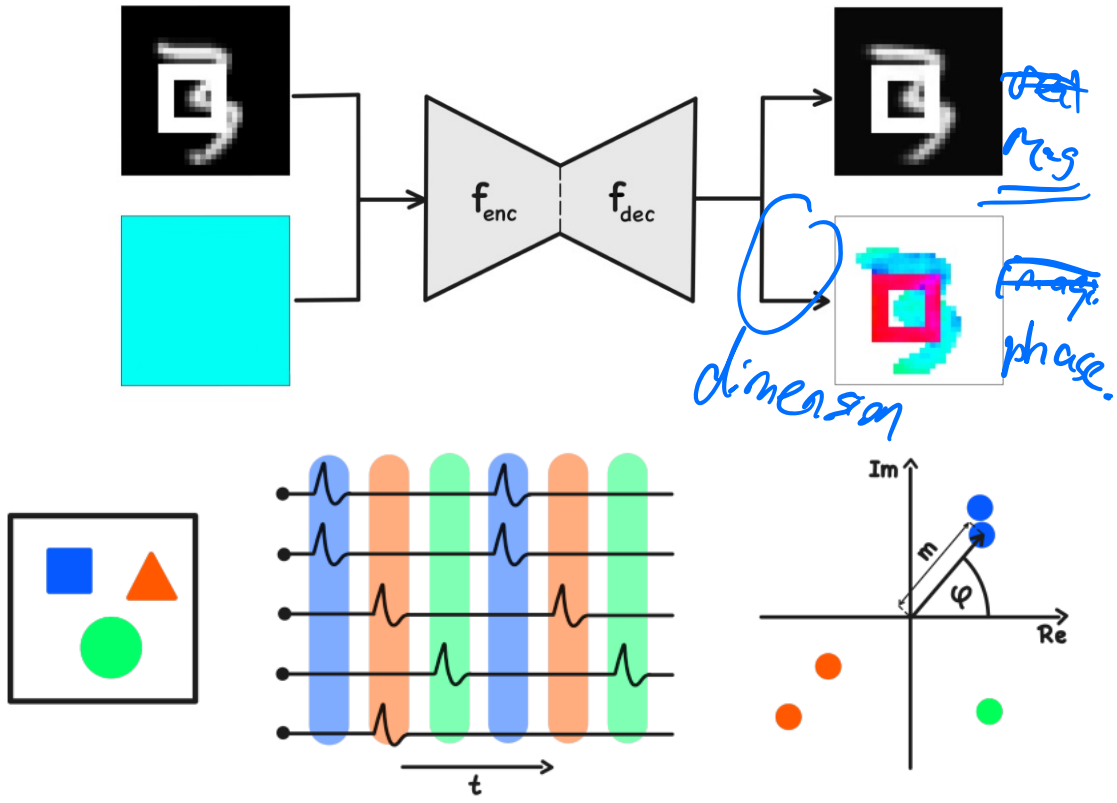


Locatello et al. Object-Centric Learning with Slot Attention. NeurIPS 2020.

NYU

Complex-Valued Autoencoders (CAEs)

- The complex number can represent magnitude and phase: $z = m \cdot e^{i\varphi} \in \mathbb{C}$.

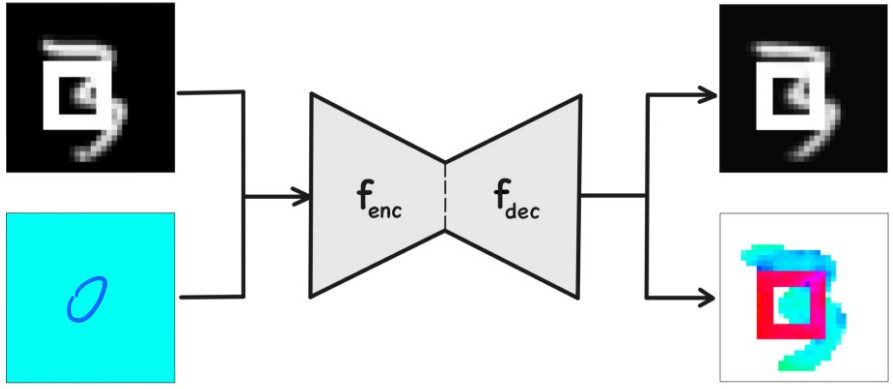


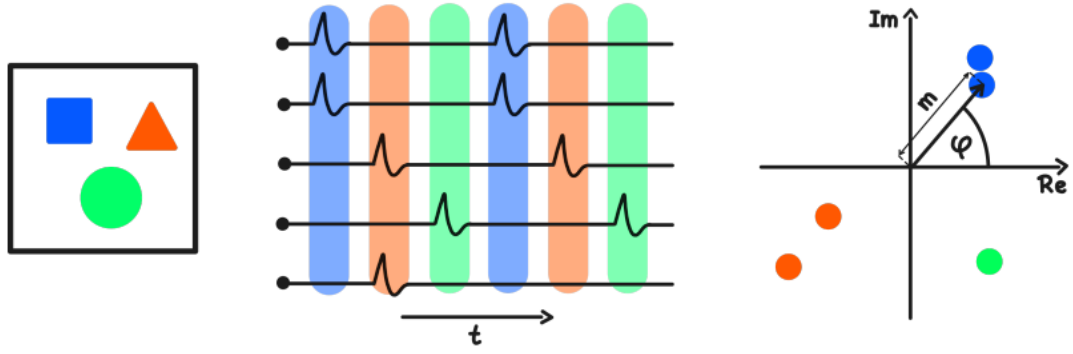
Löwe et al. Complex-Valued Autoencoders for Object Discovery. TMLR 2022.

NYU

Complex-Valued Autoencoders (CAEs)

- The complex number can represent magnitude and phase: $z = m \cdot e^{i\varphi} \in \mathbb{C}$.
- Each pixel starts with an initial phase 0.



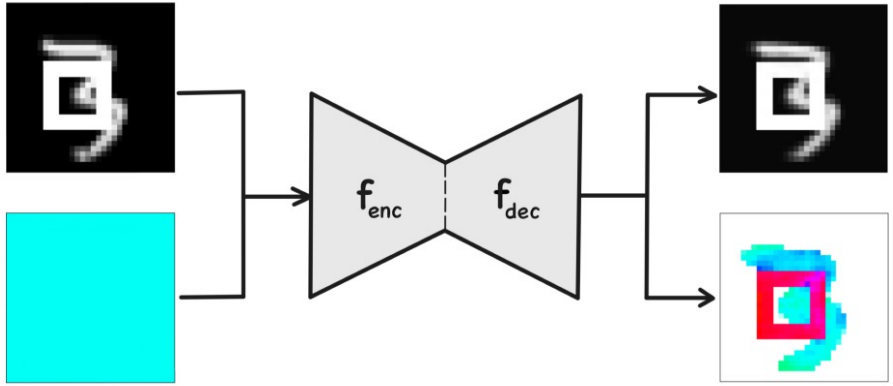


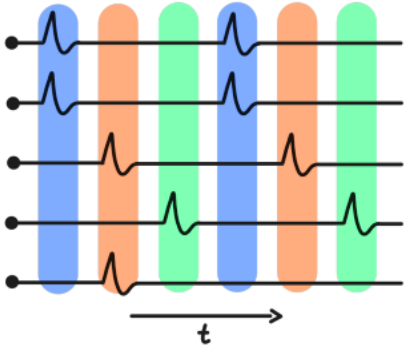
NYU

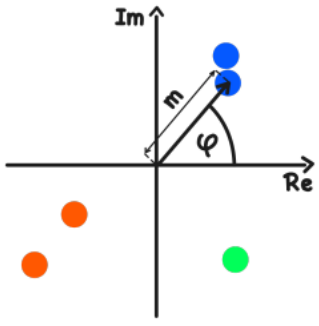
Complex-Valued Autoencoders (CAEs)

- The complex number can represent magnitude and phase: $z = m \cdot e^{i\varphi} \in \mathbb{C}$.
- Each pixel starts with an initial phase 0.

$$\hat{\mathbf{z}} = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x})) \in \mathbb{C}^{h \times w}.$$







Löwe et al. Complex-Valued Autoencoders for Object Discovery. TMLR 2022.

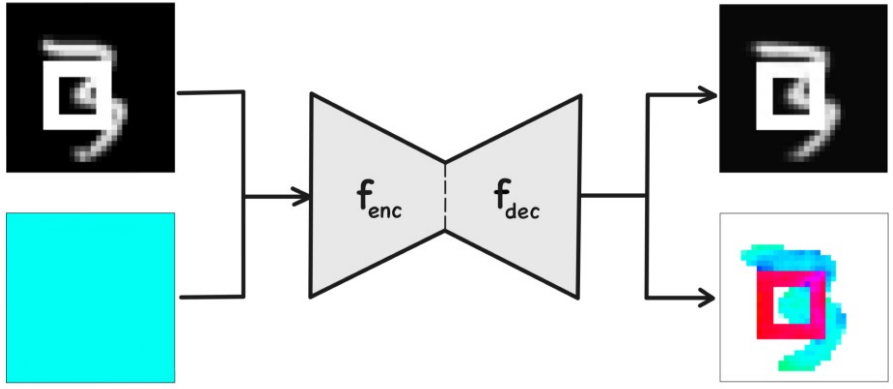
NYU

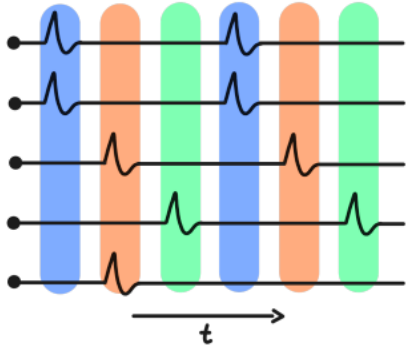
Complex-Valued Autoencoders (CAEs)

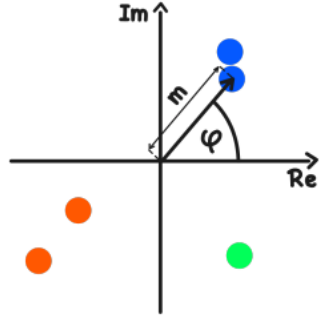
- The complex number can represent magnitude and phase: $z = m \cdot e^{i\varphi} \in \mathbb{C}$.
- Each pixel starts with an initial phase 0.

$$\hat{\mathbf{z}} = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x})) \in \mathbb{C}^{h \times w}.$$

$$\hat{\mathbf{x}} = f_{\text{out}}(\hat{\mathbf{z}}) \in \mathbb{R}^{h \times w}.$$







Löwe et al. Complex-Valued Autoencoders for Object Discovery. TMLR 2022.

NYU

CAE: More Details

- Apply weights separately to real and imaginary:

$$\boldsymbol{\psi} = f_{\mathbf{w}}(\mathbf{z}) = f_{\mathbf{w}}(\mathrm{Re}(\mathbf{z})) + f_{\mathbf{w}}(\mathrm{Im}(\mathbf{z})) \cdot i \quad \in \mathbb{C}_{d_{\mathrm{out}}}$$

Löwe et al. Complex-Valued Autoencoders for Object Discovery. TMLR 2022.
Neuronal Synchrony in Complex-Valued Deep Networks. ICLR 2014.

CAE: More Details

- Apply weights separately to real and imaginary:

$$\boldsymbol{\psi} = f_{\mathbf{w}}(\mathbf{z}) = f_{\mathbf{w}}(\mathrm{Re}(\mathbf{z})) + f_{\mathbf{w}}(\mathrm{Im}(\mathbf{z})) \cdot i \quad \in \mathbb{C}_{d_{\text{out}}}$$

- Bias on magnitude and phase:

$$\boldsymbol{m}_{\boldsymbol{\psi}} = |\boldsymbol{\psi}| + \boldsymbol{b}_{\boldsymbol{m}} \quad \in \mathbb{R}^{d_{\text{out}}} \quad \boldsymbol{\varphi}_{\boldsymbol{\psi}} = \arg(\psi) + \boldsymbol{b}_{\boldsymbol{\varphi}} \quad \in \mathbb{R}^{d_{\text{out}}}$$

Löwe et al. Complex-Valued Autoencoders for Object Discovery. TMLR 2022.
Neuronal Synchrony in Complex-Valued Deep Networks. ICLR 2014.

CAE: More Details

- Apply weights separately to real and imaginary:

$$\boldsymbol{\psi} = f_{\mathbf{w}}(\mathbf{z}) = f_{\mathbf{w}}(\mathrm{Re}(\mathbf{z})) + f_{\mathbf{w}}(\mathrm{Im}(\mathbf{z})) \cdot i \quad \in \mathbb{C}_{d_{\text{out}}}$$

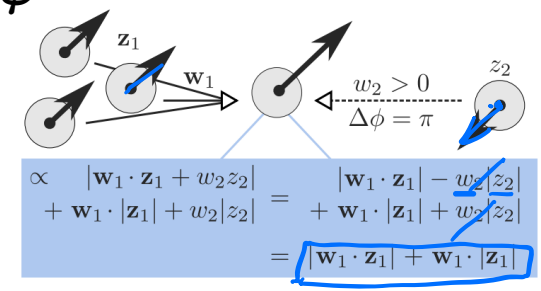
- Bias on magnitude and phase:

$$\boldsymbol{m}_{\boldsymbol{\psi}} = |\boldsymbol{\psi}| + \boldsymbol{b}_{\boldsymbol{m}} \quad \in \mathbb{R}^{d_{\text{out}}} \quad \boldsymbol{\varphi}_{\boldsymbol{\psi}} = \arg(\psi) + \boldsymbol{b}_{\boldsymbol{\varphi}} \quad \in \mathbb{R}^{d_{\text{out}}}$$

- Gating:

$$\boldsymbol{\chi} = f_{\mathbf{w}}(|\mathbf{z}|) + \boldsymbol{b}_{\boldsymbol{m}} \in \mathbb{R}^{d_{\text{out}}}$$

$$\boldsymbol{m}_{\mathbf{z}} = 0.5 \cdot \boldsymbol{m}_{\boldsymbol{\psi}} + 0.5 \cdot \boldsymbol{\chi} \in \mathbb{R}^{d_{\text{out}}}$$



Löwe et al. Complex-Valued Autoencoders for Object Discovery. TMLR 2022.
Neuronal Synchrony in Complex-Valued Deep Networks. ICLR 2014.

NYU

CAE: More Details

- Apply weights separately to real and imaginary:

$$\boldsymbol{\psi} = f_{\mathbf{w}}(\mathbf{z}) = f_{\mathbf{w}}(\mathrm{Re}(\mathbf{z})) + f_{\mathbf{w}}(\mathrm{Im}(\mathbf{z})) \cdot i \quad \in \mathbb{C}_{d_{\text{out}}}$$

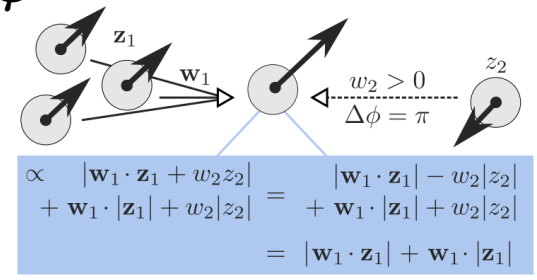
- Bias on magnitude and phase:

$$\boldsymbol{m}_{\boldsymbol{\psi}} = |\boldsymbol{\psi}| + \boldsymbol{b}_{\boldsymbol{m}} \quad \in \mathbb{R}^{d_{\text{out}}} \quad \boldsymbol{\varphi}_{\boldsymbol{\psi}} = \arg(\psi) + \boldsymbol{b}_{\boldsymbol{\varphi}} \quad \in \mathbb{R}^{d_{\text{out}}}$$

- Gating:

$$\boldsymbol{\chi} = f_{\mathbf{w}}(|\mathbf{z}|) + \boldsymbol{b}_{\boldsymbol{m}} \in \mathbb{R}^{d_{\text{out}}}$$

$$\boldsymbol{m}_{\mathbf{z}} = 0.5 \cdot \boldsymbol{m}_{\boldsymbol{\psi}} + 0.5 \cdot \boldsymbol{\chi} \in \mathbb{R}^{d_{\text{out}}}$$



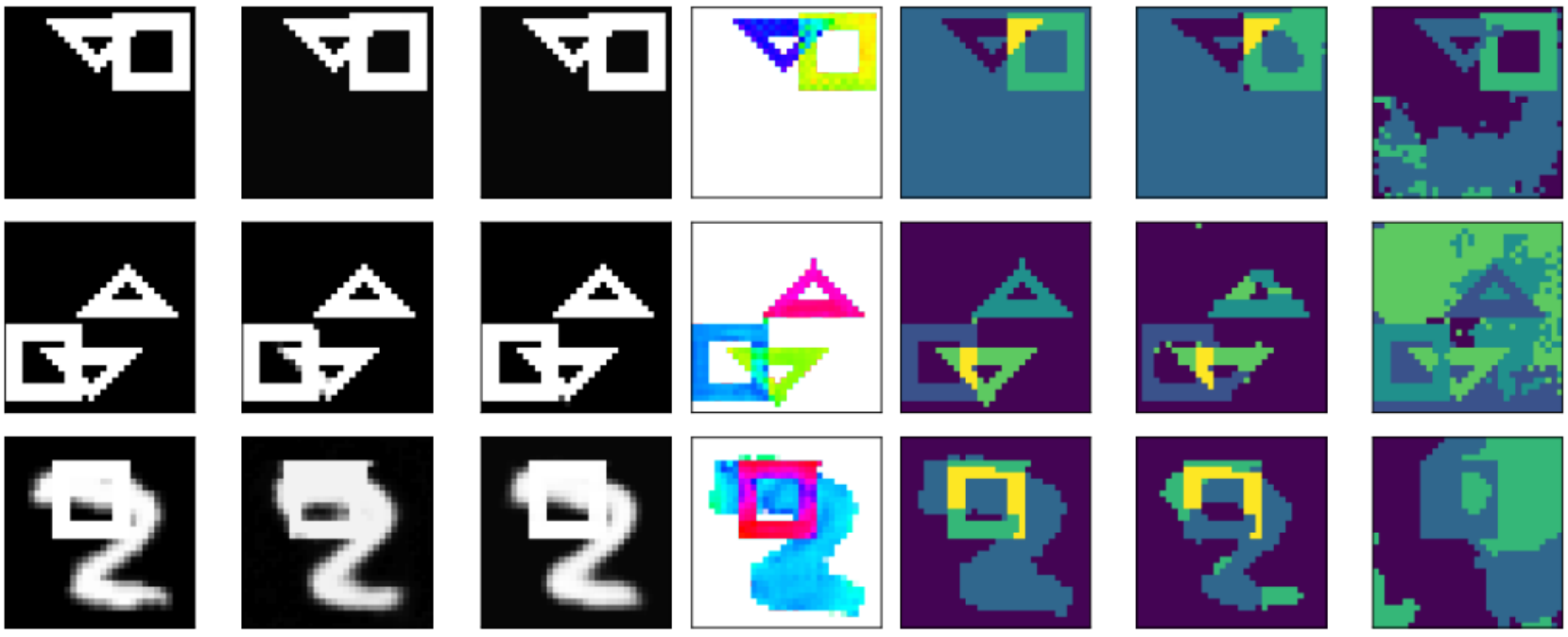
- Activation: $\mathbf{z}' = \mathrm{ReLU}(\mathrm{BatchNorm}(\boldsymbol{m}_{\mathbf{z}})) \cdot e^{i\boldsymbol{\varphi}_{\boldsymbol{\psi}}} \quad \in \mathbb{C}^{d_{\text{out}}}$

mag, phase

Löwe et al. Complex-Valued Autoencoders for Object Discovery. TMLR 2022.
Neuronal Synchrony in Complex-Valued Deep Networks. ICLR 2014.

NYU

Complex-Valued Autoencoders



Löwe et al. Complex-Valued Autoencoders for Object Discovery. TMLR 2022.

NYU

Summary: Object Discovery

- Combine deep features with clustering algorithms.

NYU

Summary: Object Discovery

- Combine deep features with clustering algorithms.
- Pseudo-labels to train detector networks.

NYU

Summary: Object Discovery

- Combine deep features with clustering algorithms.
- Pseudo-labels to train detector networks.
- Creative end-to-end learning-based solutions exist, but there are still plenty room for improvement.
  - Possible to train from scratch!

NYU

Summary: Object Discovery

- Combine deep features with clustering algorithms.
- Pseudo-labels to train detector networks.
- Creative end-to-end learning-based solutions exist, but there are still plenty room for improvement.
  - Possible to train from scratch!
- What do we make use of the discovered objects? Is it better to keep the awareness in the latent space?

NYU

Module 4:
World Models and End-to-End Planning

NYU

World Models: Predicting Future

World Models: Predicting Future

- There is a debate whether predicting future is necessary.

NYU

World Models: Predicting Future

- There is a debate whether predicting future is necessary.
- Data efficiency
  - Predicting future can “simulate” roll out without querying for outcomes.
  - Leveraging the massive amount of data in past experiences (not just the final reward).

NYU

World Models: Predicting Future

- There is a debate whether predicting future is necessary.
- Data efficiency
  - Predicting future can “simulate” roll out without querying for outcomes.
  - Leveraging the massive amount of data in past experiences (not just the final reward).
- Long-horizon planning: Predicting high-level future steps is needed.

NYU

World Models: Predicting Future

- There is a debate whether predicting future is necessary.
- Data efficiency
  - Predicting future can “simulate” roll out without querying for outcomes.
  - Leveraging the massive amount of data in past experiences (not just the final reward).
- Long-horizon planning: Predicting high-level future steps is needed.
- Can representations learned from SSL help us build better prediction?

NYU

Model-Predictive Control (MPC)

- A classic example of world models.

Rawlings et al. Model Predictive Control: Theory, Computation, and Design.

NYU

Model-Predictive Control (MPC)

- A classic example of world models.
- Analytical forms, complete knowledge of the dynamical system.

Rawlings et al. Model Predictive Control: Theory, Computation, and Design.

NYU

Model-Predictive Control (MPC)

- A classic example of world models.
- Analytical forms, complete knowledge of the dynamical system.

General Form:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u}, t)$$

$$\mathbf{y} = h(\mathbf{x}, \mathbf{u}, t)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0$$

state control

Rawlings et al. Model Predictive Control: Theory, Computation, and Design.

NYU

Model-Predictive Control (MPC)

- A classic example of world models.
- Analytical forms, complete knowledge of the dynamical system.

General Form:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u}, t)$$
$$\mathbf{y} = h(\mathbf{x}, \mathbf{u}, t)$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Linear Form:

$$\frac{dx}{dt} = A\mathbf{x} + B\mathbf{u}$$
$$\mathbf{y} = C\mathbf{x} + D\mathbf{u}$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Rawlings et al. Model Predictive Control: Theory, Computation, and Design.

NYU

Model-Predictive Control (MPC)

- A classic example of world models.
- Analytical forms, complete knowledge of the dynamical system.

General Form:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u}, t)$$
$$\mathbf{y} = h(\mathbf{x}, \mathbf{u}, t)$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Linear Form:

$$\frac{dx}{dt} = A\mathbf{x} + B\mathbf{u}$$
$$\mathbf{y} = C\mathbf{x} + D\mathbf{u}$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Optimize Value/Cost Function:

$$\min_{\mathbf{u}} J(\mathbf{x}(0), \mathbf{u})$$

Rawlings et al. Model Predictive Control: Theory, Computation, and Design.

NYU

Model-Predictive Control (MPC)

- A classic example of world models.
- Analytical forms, complete knowledge of the dynamical system.

General Form:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u}, t)$$
$$\mathbf{y} = h(\mathbf{x}, \mathbf{u}, t)$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Linear Form:

$$\frac{dx}{dt} = A\mathbf{x} + B\mathbf{u}$$
$$\mathbf{y} = C\mathbf{x} + D\mathbf{u}$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Optimize Value/Cost Function:

$$\min_{\mathbf{u}} J(\mathbf{x}(0), \mathbf{u})$$

Quadratic Form (LQR):

$$J = \int_0^\infty \mathbf{x}^\top Q\mathbf{x} + \mathbf{u}^\top R\mathbf{u}\, dt.$$

reach ? smooth ?

Rawlings et al. Model Predictive Control: Theory, Computation, and Design.

NYU

Model-Predictive Control (MPC)

- A classic example of world models.
- Analytical forms, complete knowledge of the dynamical system.

General Form:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u}, t)$$
$$\mathbf{y} = h(\mathbf{x}, \mathbf{u}, t)$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Linear Form:

$$\frac{dx}{dt} = A\mathbf{x} + B\mathbf{u}$$
$$\mathbf{y} = C\mathbf{x} + D\mathbf{u}$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Optimize Value/Cost Function:

$$\min_{\mathbf{u}} J(\mathbf{x}(0), \mathbf{u})$$

Quadratic Form (LQR):

$$J = \int_0^\infty \mathbf{x}^\top Q\mathbf{x} + \mathbf{u}^\top R\mathbf{u}\, dt.$$

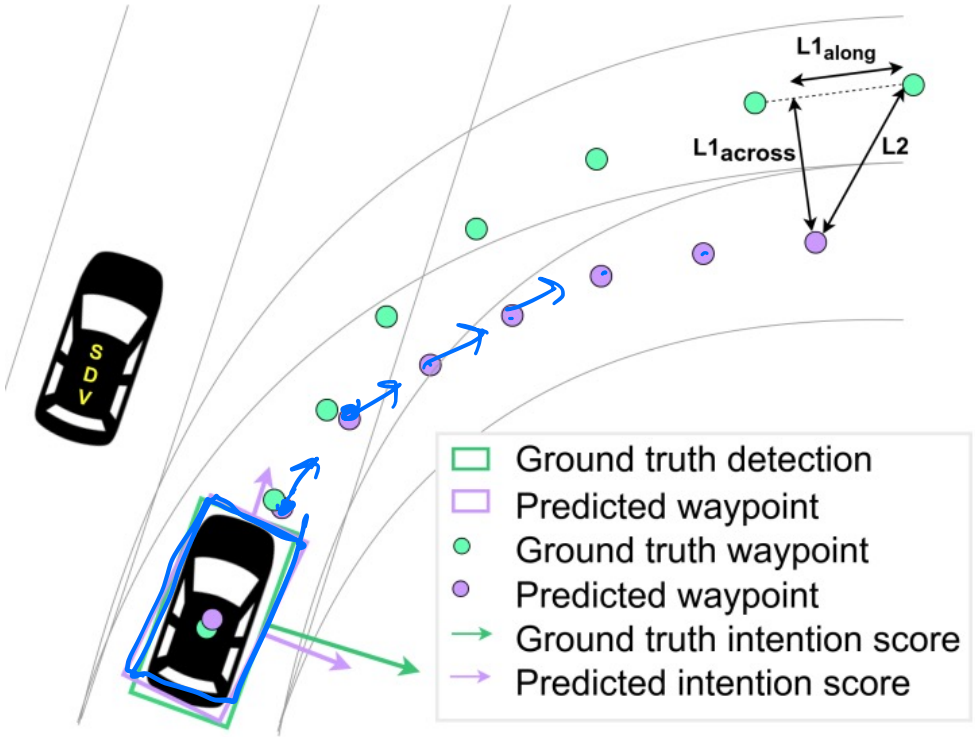
- Example: Cars: x: position velocity angle angular velocity; u: jerk and angular accel.;

Rawlings et al. Model Predictive Control: Theory, Computation, and Design.

NYU

Trajectory Prediction as Object Detection

- We also need to predict external dynamic objects.

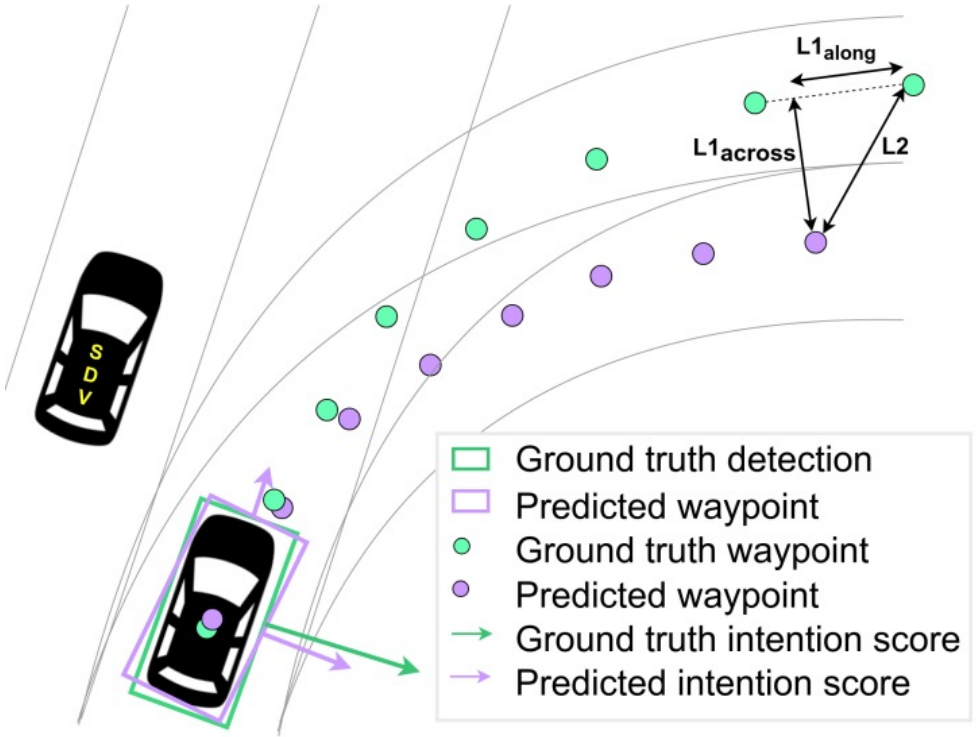


Casas et al. IntentNet: Learning to Predict Intention from Raw Sensor Data. CoRL 2018.

NYU

Trajectory Prediction as Object Detection

- We also need to predict external dynamic objects.
- How to capture multiple modes?

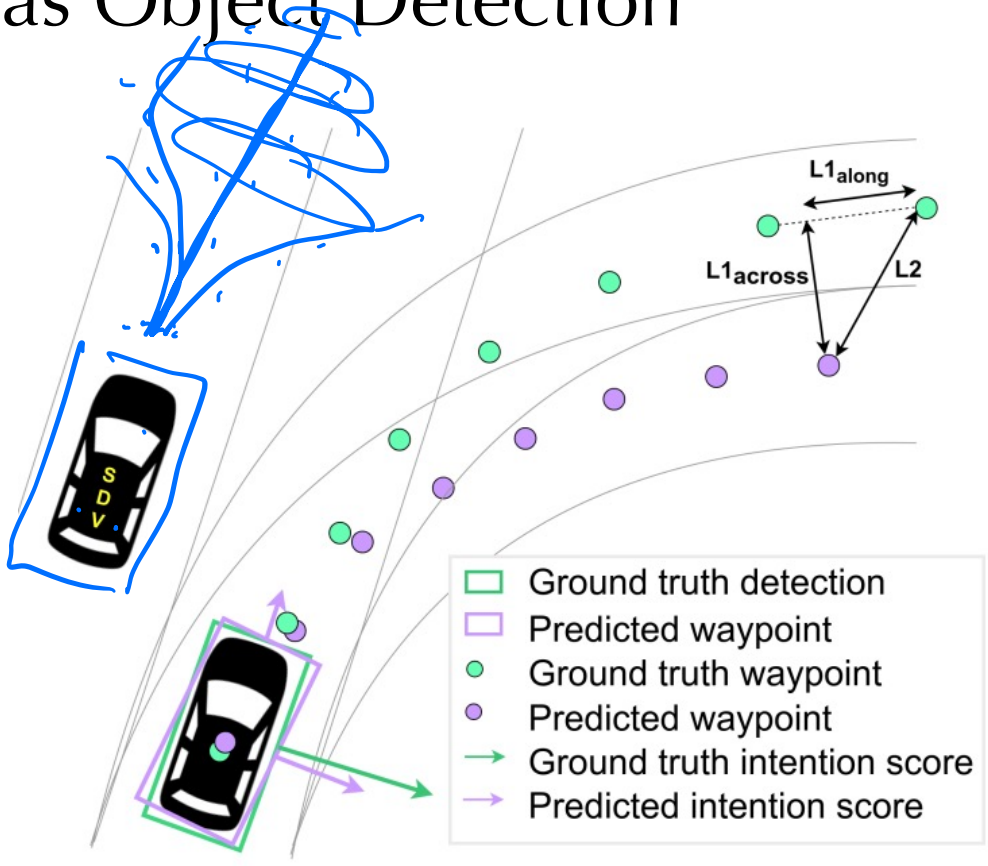


Casas et al. IntentNet: Learning to Predict Intention from Raw Sensor Data. CoRL 2018.

NYU

Trajectory Prediction as Object Detection

- We also need to predict external dynamic objects.
- How to capture multiple modes?
- Discrete intention prediction problem:
  - keep lane, turn left/right, left/right change lane, stopping, parked, etc.

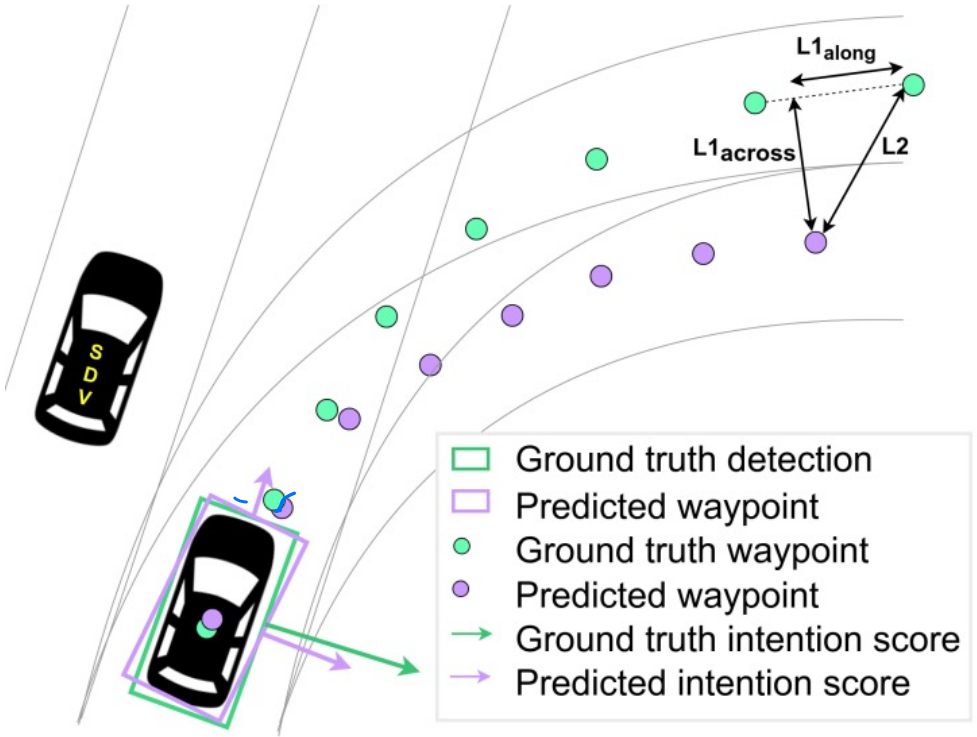


Casas et al. IntentNet: Learning to Predict Intention from Raw Sensor Data. CoRL 2018.

NYU

Trajectory Prediction as Object Detection

- We also need to predict external dynamic objects.
- How to capture multiple modes?
- Discrete intention prediction problem:
  - keep lane, turn left/right, left/right change lane, stopping, parked, etc.
- Output multiple trajectories

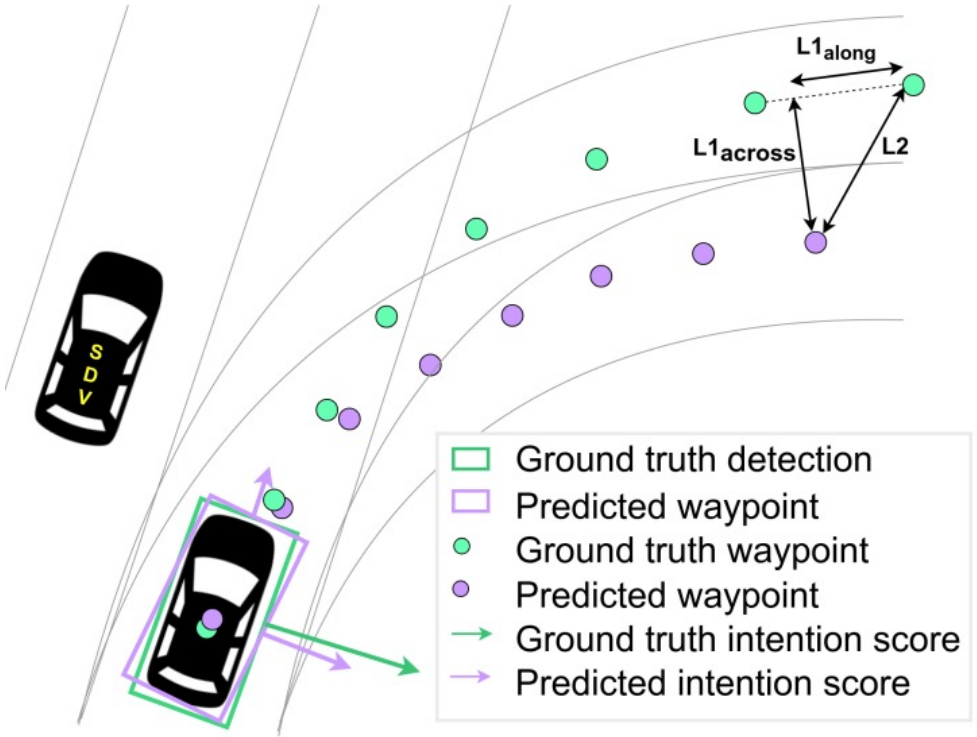


Casas et al. IntentNet: Learning to Predict Intention from Raw Sensor Data. CoRL 2018.

NYU

Trajectory Prediction as Object Detection

- We also need to predict external dynamic objects.
- How to capture multiple modes?
- Discrete intention prediction problem:
  - keep lane, turn left/right, left/right change lane, stopping, parked, etc.
- Output multiple trajectories
- Requires high-level action labels.



Casas et al. IntentNet: Learning to Predict Intention from Raw Sensor Data. CoRL 2018.

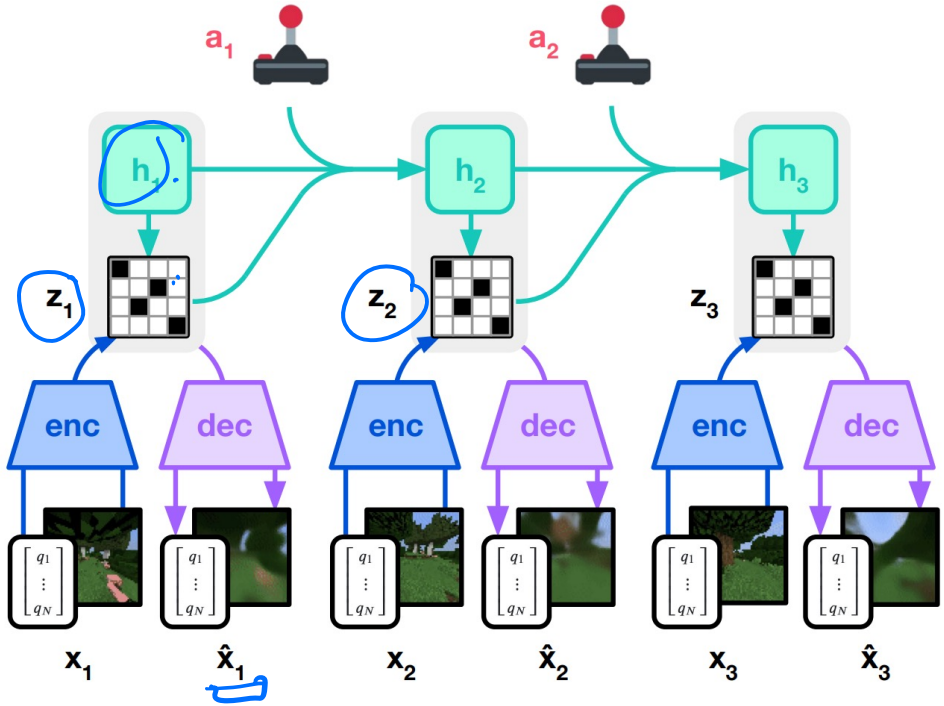
NYU

Latent Sequence World Model for RL

- Autoencoder to ensure the latent representations are meaningful.

$$z_t \sim q_\phi(z_t \mid h_t, x_t)$$

$$\hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t)$$



Hafner et al. Mastering Diverse Domains through World Models. 2023.

NYU

Latent Sequence World Model for RL

- Autoencoder to ensure the latent representations are meaningful.

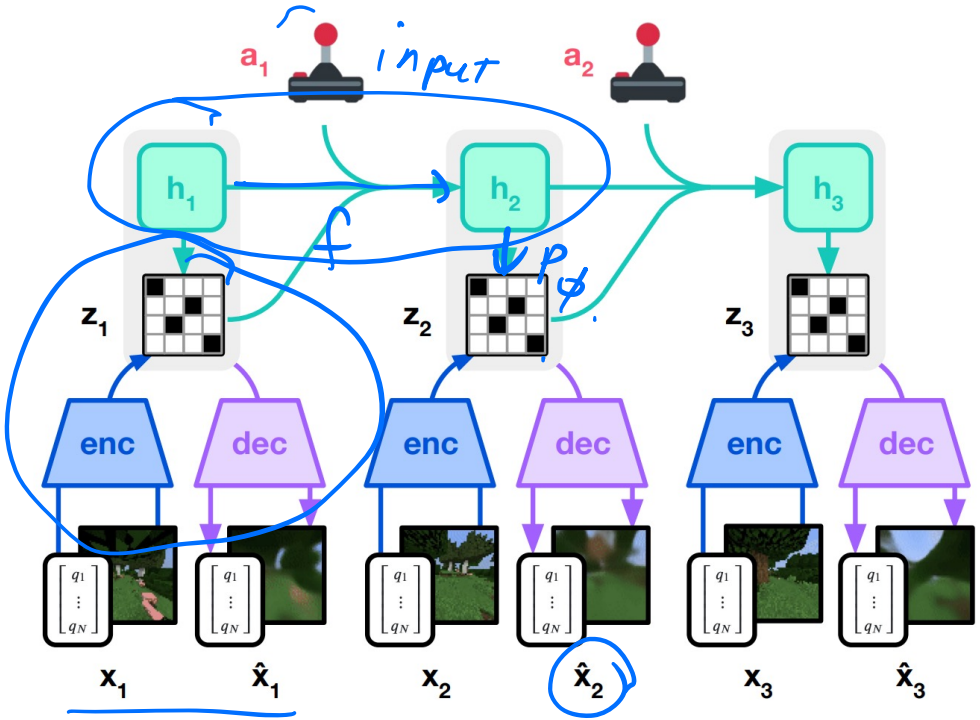
$$z_t \sim q_\phi(z_t \mid h_t, x_t)$$

$$\hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t)$$

- Learn a sequence model to predict the latent conditioned on previous action.

$$h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1})$$

$$\hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t)$$



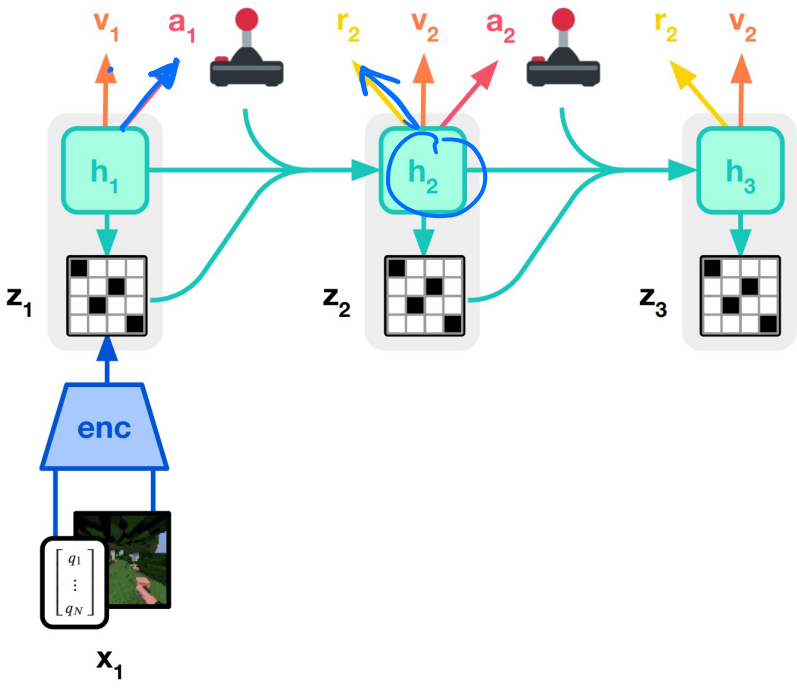
Hafner et al. Mastering Diverse Domains through World Models. 2023.

NYU

Incorporating Rewards

- Predicting reward

$$\hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t)$$



Hafner et al. Mastering Diverse Domains through World Models. 2023.

NYU

Incorporating Rewards

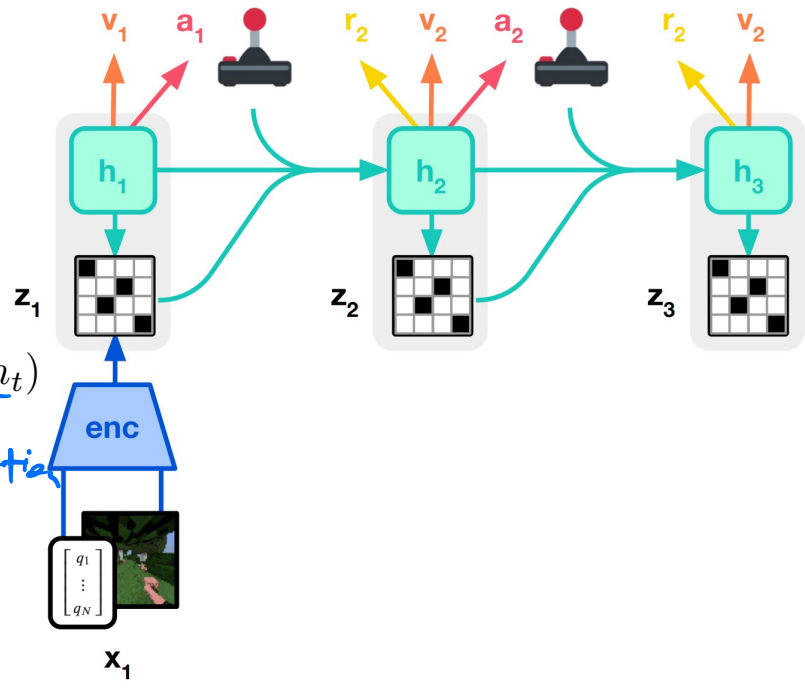
- Predicting reward

$$\hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t)$$

- Reconstruction, reward, continue

$$\mathcal{L}_{\text{pred}}(\phi) = -\log p_\phi(x_t \mid z_t, h_t) - \log p_\phi(r_t \mid z_t, h_t) - \log p_\phi(c_t \mid z_t, h_t)$$

reconstruction reward continuation



NYU

Incorporating Rewards

- Predicting reward

$$\hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t)$$

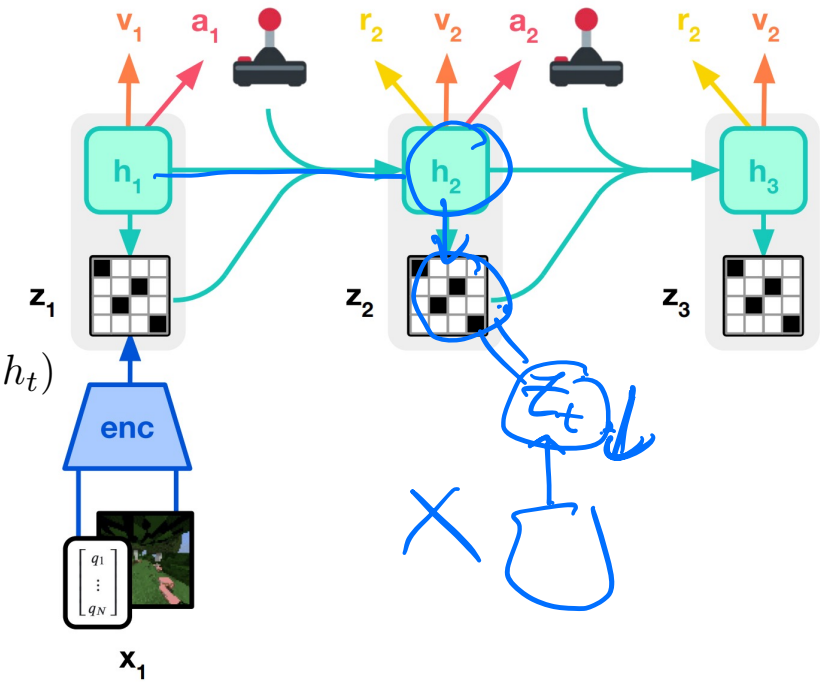
- Reconstruction, reward, continue

$$\mathcal{L}_{\text{pred}}(\phi) = -\log p_\phi(x_t \mid z_t, h_t) - \log p_\phi(r_t \mid z_t, h_t) - \log p_\phi(c_t \mid z_t, h_t)$$

- Dynamics: Predicting future z

$$\mathcal{L}_{\text{dyn}}(\phi) = \max(1, \mathrm{KL}[\mathrm{sg}(q_\phi(z_t \mid h_t, x_t)) \parallel p_\phi(z_t \mid h_t)])$$

Stop-gradient



NYU

Incorporating Rewards

- Predicting reward

$$\hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t)$$

- Reconstruction, reward, continue

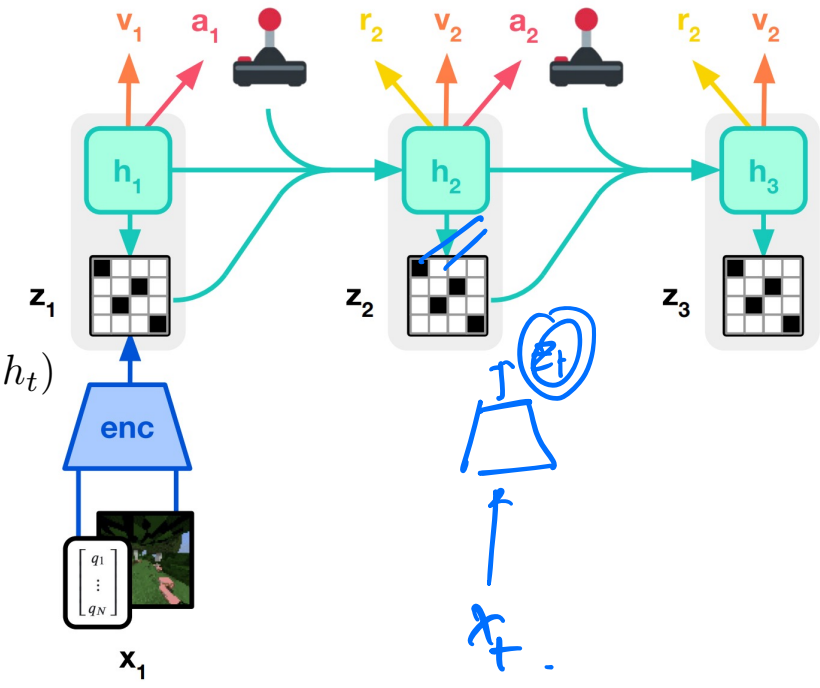
$$\mathcal{L}_{\text{pred}}(\phi) = -\log p_\phi(x_t \mid z_t, h_t) - \log p_\phi(r_t \mid z_t, h_t) - \log p_\phi(c_t \mid z_t, h_t)$$

- Dynamics: Predicting future z

$$\mathcal{L}_{\text{dyn}}(\phi) = \max(1, \mathrm{KL}[\mathrm{sg}(q_\phi(z_t \mid h_t, x_t) \parallel p_\phi(z_t|h_t))]$$

- Align representation

$$\mathcal{L}_{\text{rep}} = \max(1, \mathrm{KL}[q_\phi(z_t|h_t, x_t) \parallel \mathrm{sg}(p_\phi(z_t|h_t))])$$

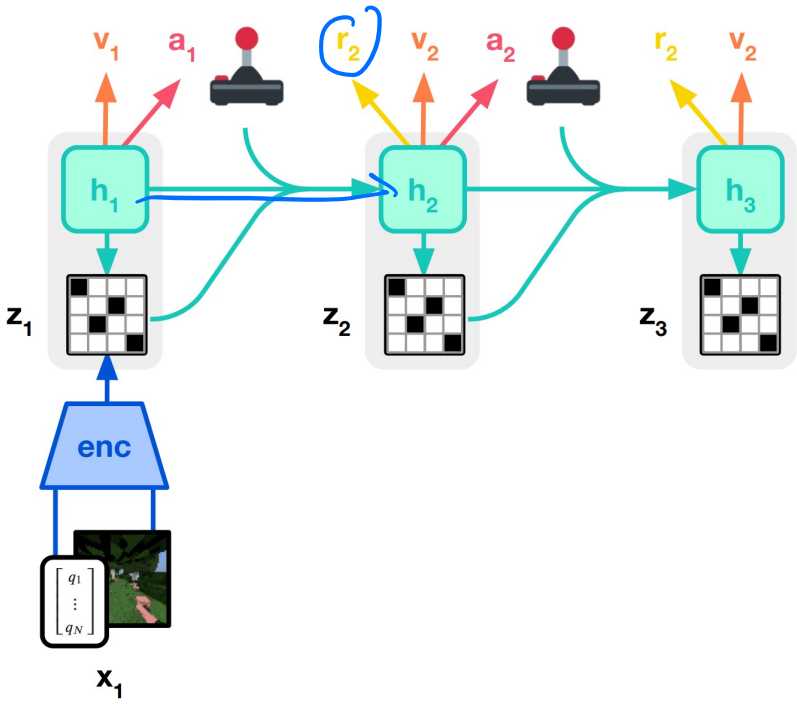


Hafner et al. Mastering Diverse Domains through World Models. 2023.

NYU

Incorporating Rewards

- How to use WM in planning? Predicting value by simulate a batch of trajectories.



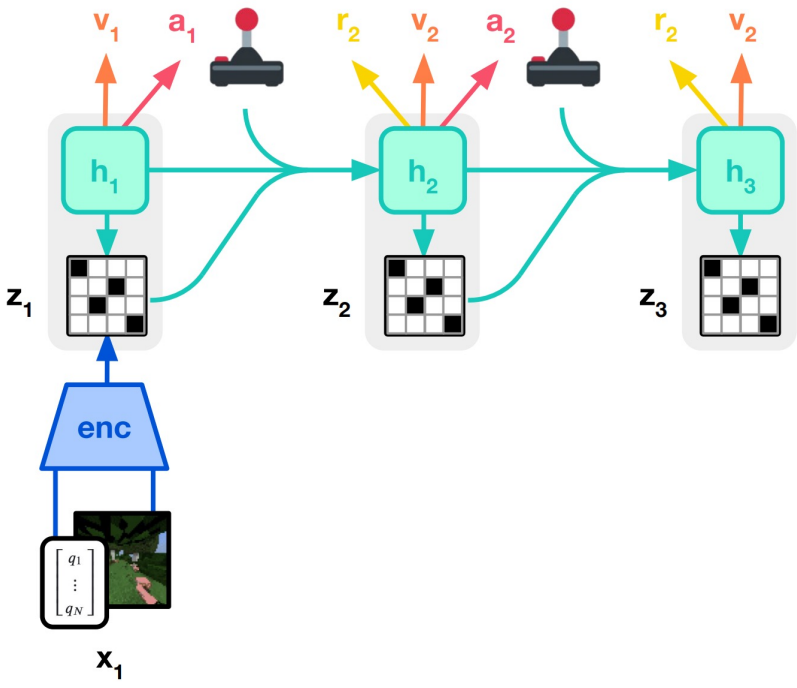
Hafner et al. Mastering Diverse Domains through World Models. 2023.

NYU

Incorporating Rewards

- How to use WM in planning? Predicting value by simulate a batch of trajectories.
- Actor-Critic RL:

$$a_t \sim \pi_\theta(a_t \mid s_t) \qquad v_\psi(R_t \mid s_t)$$



Hafner et al. Mastering Diverse Domains through World Models. 2023.

NYU

Incorporating Rewards

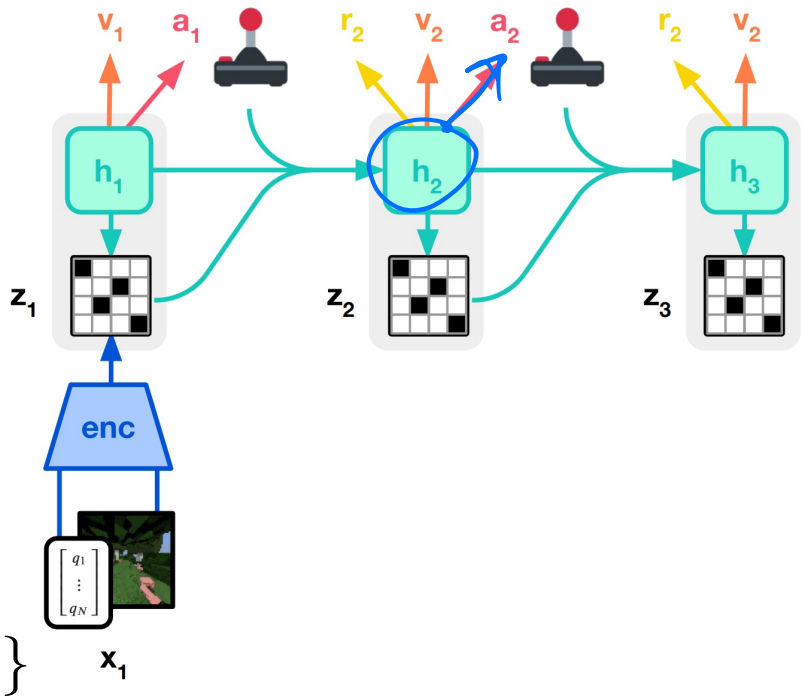
- How to use WM in planning?
  Predicting value by simulate a batch of trajectories.
- Actor-Critic RL:
  $a_t \sim \pi_\theta(a_t \mid s_t)$ $\quad$ $v_\psi(R_t \mid s_t)$
- Learning a critic:

Categorical distribution

$$\mathcal{L}_\phi = -\sum_{t=1}^{T} \log p_\phi(R_t^\lambda \mid s_t) \quad s_t = \{h_t, z_t\}$$

Sum of discounted future rewards

$$R_t^\lambda = r_t + \gamma c_t[(1-\lambda)v_t + \lambda R_{t+1}^\lambda] \quad R_T^\lambda = v_T$$



Hafner et al. Mastering Diverse Domains through World Models. 2023.

Actor Learning

- Learn a policy network $a_t \sim \pi_\theta(a_t \mid s_t)$

Levine. Policy Gradients. Deep RL course. 2017.

NYU

Actor Learning

- Learn a policy network $a_t \sim \pi_\theta(a_t \mid s_t)$
- REINFORCE algorithm [Williams 1992]

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \mathrm{sg}((R_t^\lambda - v_\phi(s_t))/\underbrace{\max(1, S)}_{\text{Normalization}})\log \pi_\theta(a_t \mid s_t) + \underbrace{\eta H[\pi_\theta(a_t \mid s_t)]}_{\text{Entropy Regularizer}}$$

Critic.

Levine. Policy Gradients. Deep RL course. 2017.

NYU

Actor Learning

- Learn a policy network $a_t \sim \pi_\theta(a_t \mid s_t)$
- REINFORCE algorithm [Williams 1992]

Normalization Entropy Regularizer

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \mathrm{sg}((R_t^\lambda - v_\phi(s_t))/\max(1, S)) \log \pi_\theta(a_t \mid s_t) + \eta H[\pi_\theta(a_t \mid s_t)]$$

- Notes on policy gradient:

Levine. Policy Gradients. Deep RL course. 2017.

NYU

Actor Learning

- Learn a policy network $a_t \sim \pi_\theta(a_t \mid s_t)$
- REINFORCE algorithm [Williams 1992]

Normalization Entropy Regularizer

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \mathrm{sg}((R_t^\lambda - v_\phi(s_t))/\max(1, S)) \log \pi_\theta(a_t \mid s_t) + \eta H[\pi_\theta(a_t \mid s_t)]$$

- Notes on policy gradient:

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau).$$

Levine. Policy Gradients. Deep RL course. 2017.

NYU

Actor Learning

- Learn a policy network $a_t \sim \pi_\theta(a_t \mid s_t)$
- REINFORCE algorithm [Williams 1992]

Normalization Entropy Regularizer

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \mathrm{sg}((R_t^\lambda - v_\phi(s_t))/\max(1, S)) \log \pi_\theta(a_t \mid s_t) + \eta H[\pi_\theta(a_t \mid s_t)]$$

- Notes on policy gradient:

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau).$$

$$\pi_\theta(\tau) = \pi_\theta(s_1, a_1, \ldots, s_T, a_T) = p(s_1)\prod_{t=1}^{T} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t).$$

Levine. Policy Gradients. Deep RL course. 2017.

NYU

Actor Learning

- Learn a policy network $a_t \sim \pi_\theta(a_t \mid s_t)$
- REINFORCE algorithm [Williams 1992]

Normalization

Entropy Regularizer

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \mathrm{sg}((R_t^\lambda - v_\phi(s_t))/\max(1, S)) \log \pi_\theta(a_t \mid s_t) + \eta H[\pi_\theta(a_t \mid s_t)]$$

- Notes on policy gradient:

$$\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau).$$

$$\pi_\theta(\tau) = \pi_\theta(s_1, a_1, \ldots, s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t).$$

$$\log \pi_\theta(\tau) = \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t).$$

Levine. Policy Gradients. Deep RL course. 2017.

NYU

Actor Learning

- Learn a policy network $a_t \sim \pi_\theta(a_t \mid s_t)$
- REINFORCE algorithm [Williams 1992]

Normalization Entropy Regularizer

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \mathrm{sg}((R_t^\lambda - v_\phi(s_t))/\max(1, S)) \log \pi_\theta(a_t \mid s_t) + \eta H[\pi_\theta(a_t \mid s_t)]$$

- Notes on policy gradient:

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau).$$

$$\pi_\theta(\tau) = \pi_\theta(s_1, a_1, \ldots, s_T, a_T) = p(s_1)\prod_{t=1}^{T} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t).$$

$$\log \pi_\theta(\tau) = \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t).$$

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[r(\tau)] = \int \pi_\theta r(\tau) d\tau.$$

Levine. Policy Gradients. Deep RL course. 2017.

NYU

Actor Learning

- Learn a policy network $a_t \sim \pi_\theta(a_t \mid s_t)$
- REINFORCE algorithm [Williams 1992]

Normalization — Entropy Regularizer

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \mathrm{sg}((R_t^\lambda - v_\phi(s_t))/\max(1, S)) \log \pi_\theta(a_t \mid s_t) + \eta H[\pi_\theta(a_t \mid s_t)]$$

log. softmax over action

- Notes on policy gradient:

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau).$$

$$\pi_\theta(\tau) = \pi_\theta(s_1, a_1, \ldots, s_T, a_T) = p(s_1)\prod_{t=1}^{T} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t).$$

$$\log \pi_\theta(\tau) = \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t).$$

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[r(\tau)] = \int \pi_\theta r(\tau) d\tau.$$

$$\nabla\mathcal{L}(\theta) = \int \nabla\pi_\theta r(\tau) d\tau = \int \pi_\theta(\tau)\nabla \log \pi_\theta r(\tau) d\tau = \mathbb{E}_{\tau\sim\pi} \log \pi_\theta r(\tau).$$

Levine. Policy Gradients. Deep RL course. 2017.

NYU

When Do We Need A Learned Actor?

- For low dimensional or discrete problems, we can directly take the argmax of the value function.

NYU

When Do We Need A Learned Actor?

- For low dimensional or discrete problems, we can directly take the argmax of the value function.
- For problems with a good model, we can roll out and sample many future trajectories. Evaluation can be done in real time with GPU.

model-based

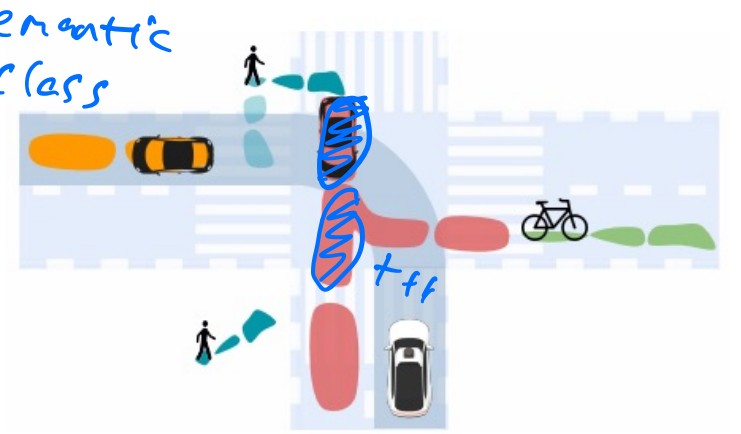
When Do We Need A Learned Actor?

- For low dimensional or discrete problems, we can directly take the argmax of the value function.
- For problems with a good model, we can roll out and sample many future trajectories. Evaluation can be done in real time with GPU.
- For general control problems, learning a separate actor can be a general solution without invoking domain knowledge.

NYU

Semantic Occupancy Volume Prediction

- 4-D volume: H, W, T, C (class) $o_{i,j}^{t,c}$



Perceive, Predict, and Plan: Safe Motion Planning Through Interpretable Semantic Representations. ECCV 2020.

NYU

Semantic Occupancy Volume Prediction

- 4-D volume: H, W, T, C (class) $o_{i,j}^{t,c}$
- Doesn't grow with the increasing number of actors.

fixed dimensional.

Perceive, Predict, and Plan: Safe Motion Planning Through Interpretable Semantic Representations. ECCV 2020.

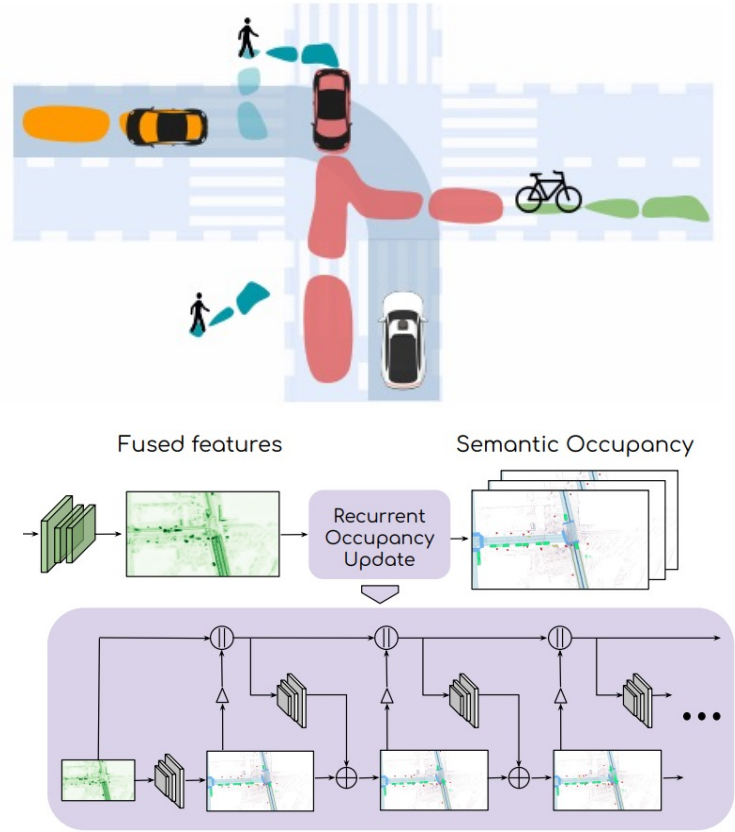
NYU

Semantic Occupancy Volume Prediction

- 4-D volume: H, W, T, C (class) $o_{i,j}^{t,c}$
- Doesn't grow with the increasing number of actors.
- Recurrent occupancy updates for further into the future.

$$l^{t,c} = l^{t-1,c} + \mathcal{U}_\theta^t(f_{\text{occ}}, l^{0:t-1,c})$$

occ next. prev occ. network.

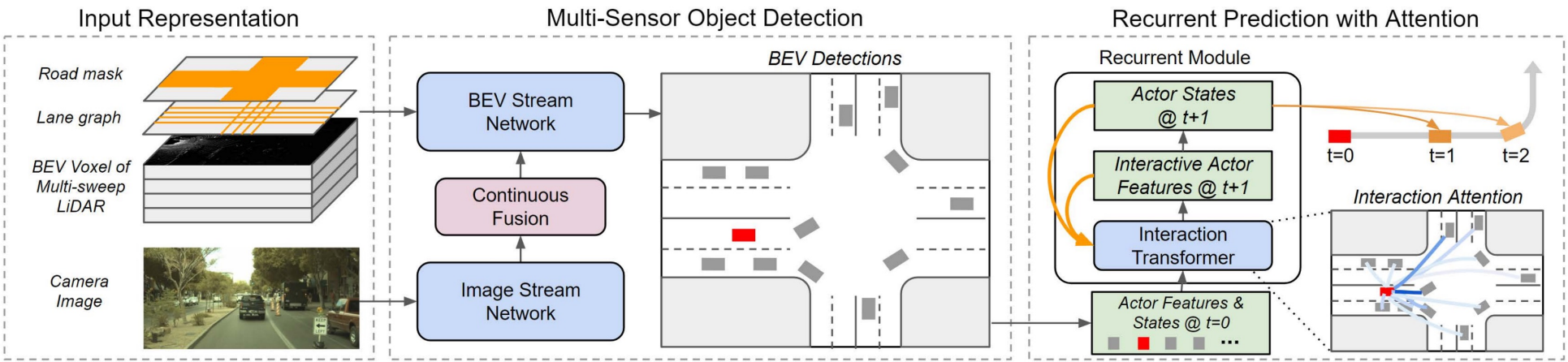


Perceive, Predict, and Plan: Safe Motion Planning Through Interpretable Semantic Representations. ECCV 2020.

NYU

Multi-Agents Joint Prediction

- Joint predict future trajectories by attending to other actors.

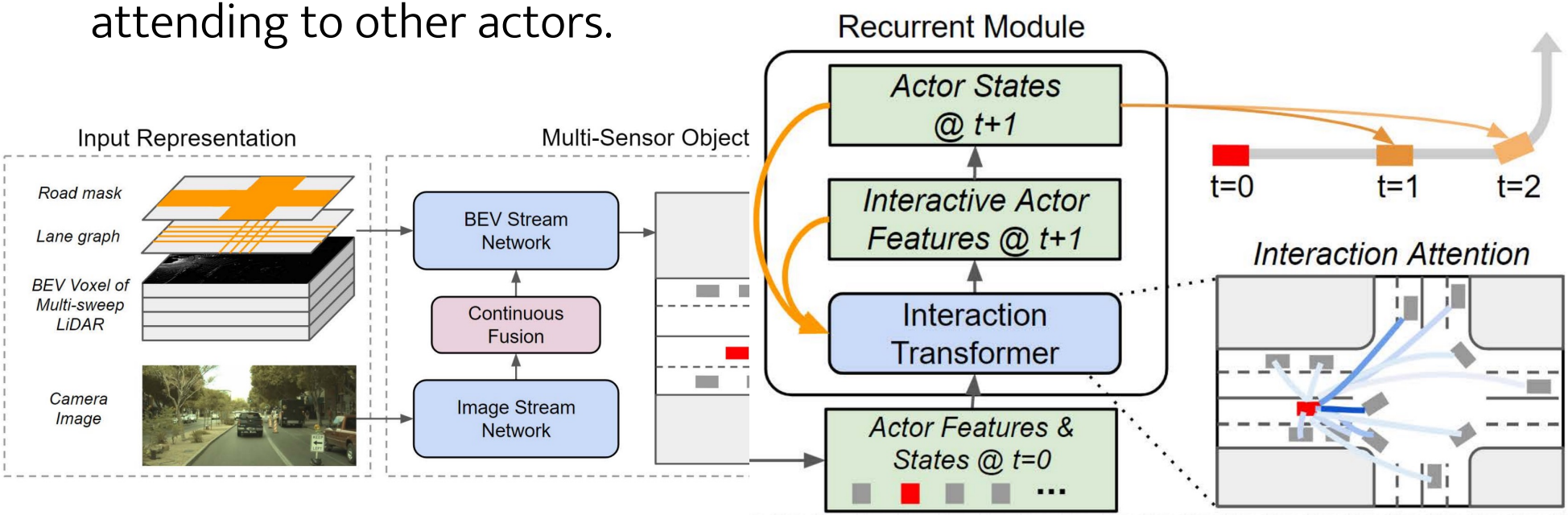


Li et al. End-to-end Contextual Perception and Prediction with Interaction Transformer. IROS 2020.

NYU

Multi-Agents Joint Prediction

- Joint predict future trajectories by attending to other actors.

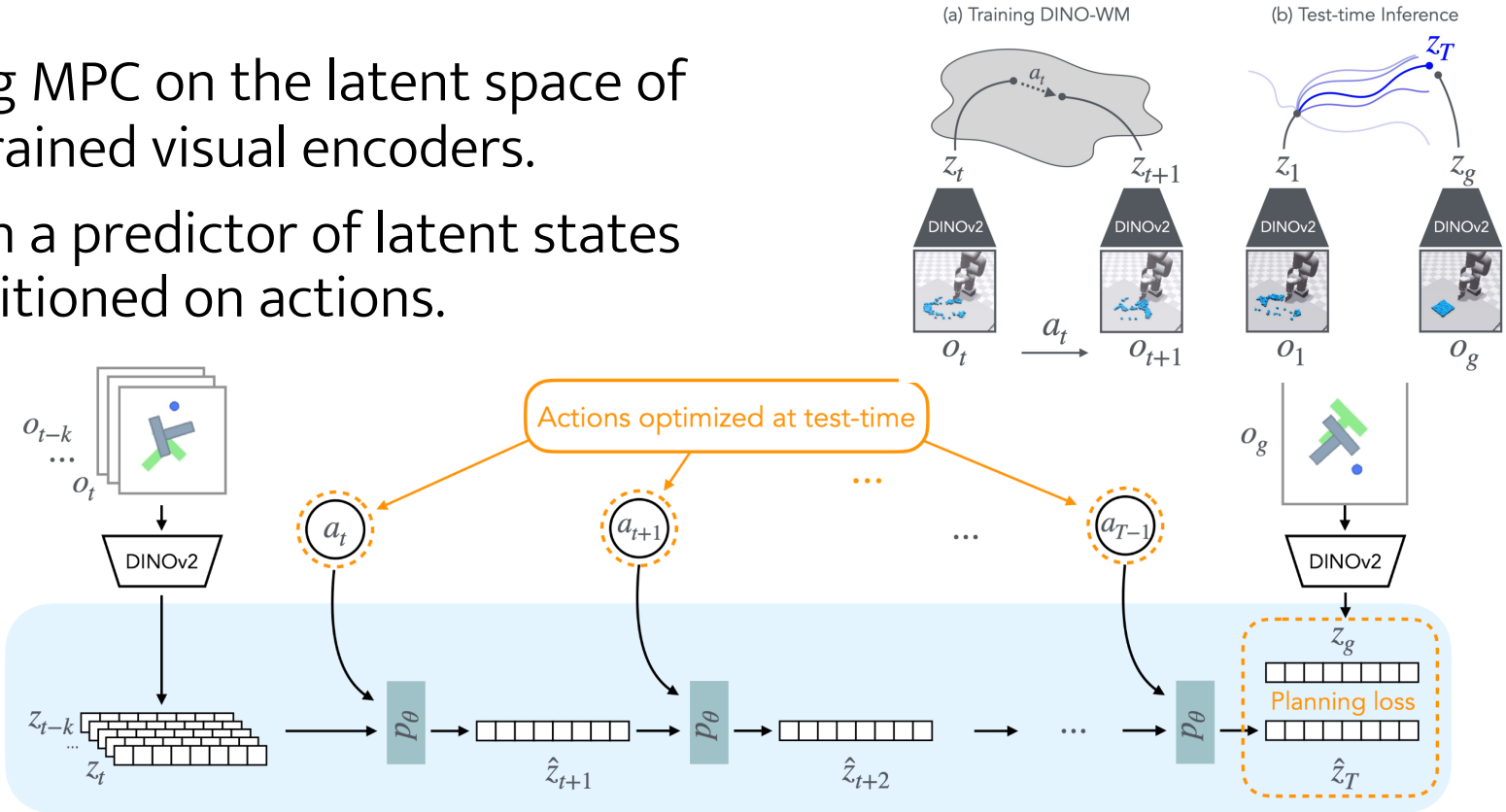


Li et al. End-to-end Contextual Perception and Prediction with Interaction Transformer. IROS 2020.

NYU

Latent Prediction + MPC

- Using MPC on the latent space of pretrained visual encoders.
- Learn a predictor of latent states conditioned on actions.

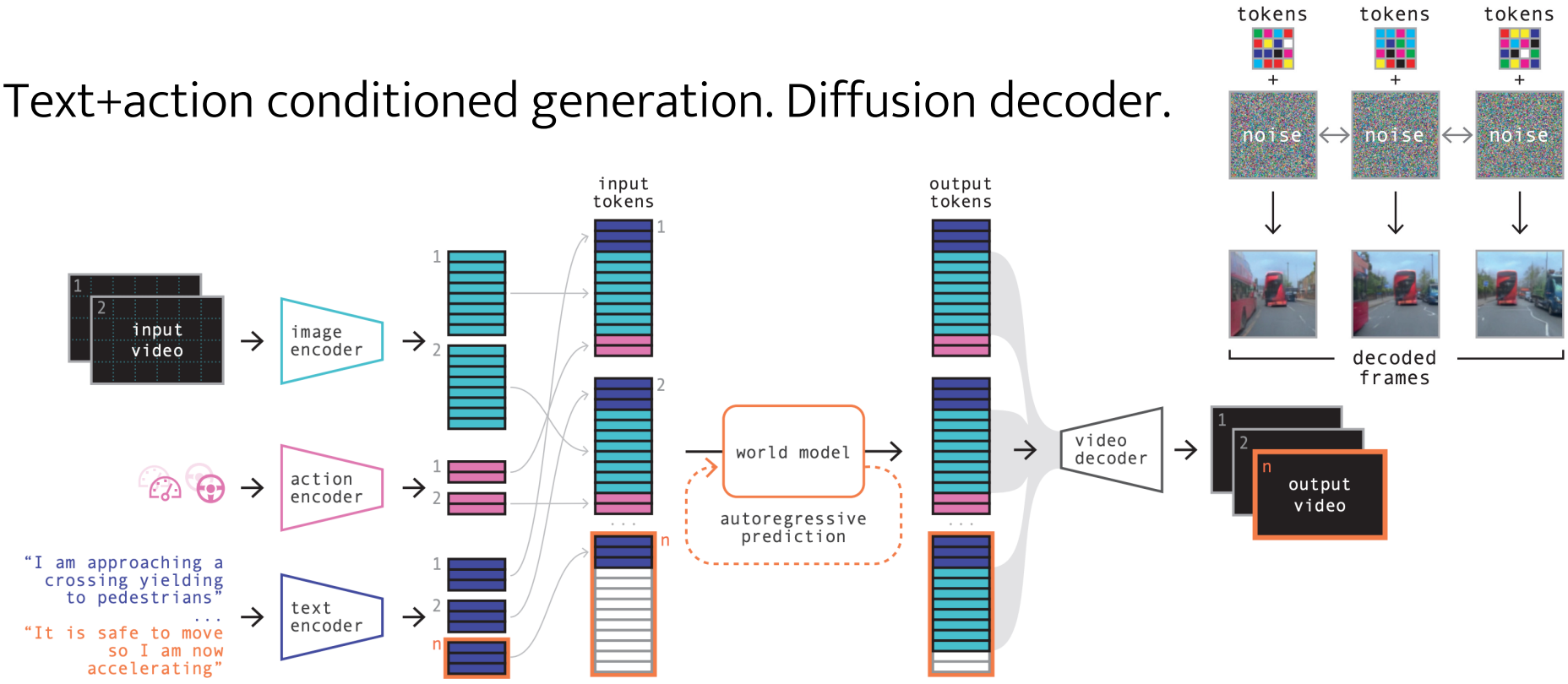


Zhou et al. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning. 2024.

NYU

World Model in Video Prediction

- Text+action conditioned generation. Diffusion decoder.

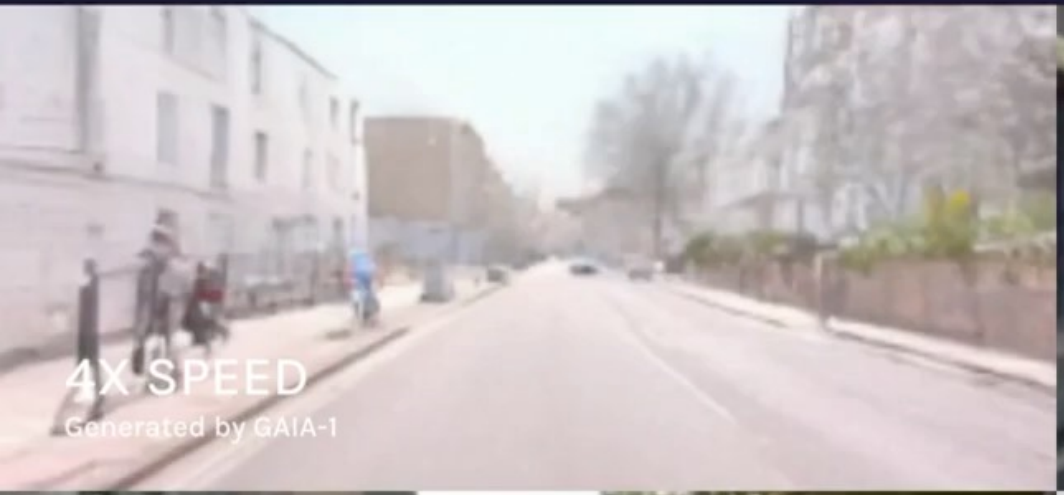


Hu et al. GAIA-1: A Generative World Model for Autonomous Driving. arXiv 2023.

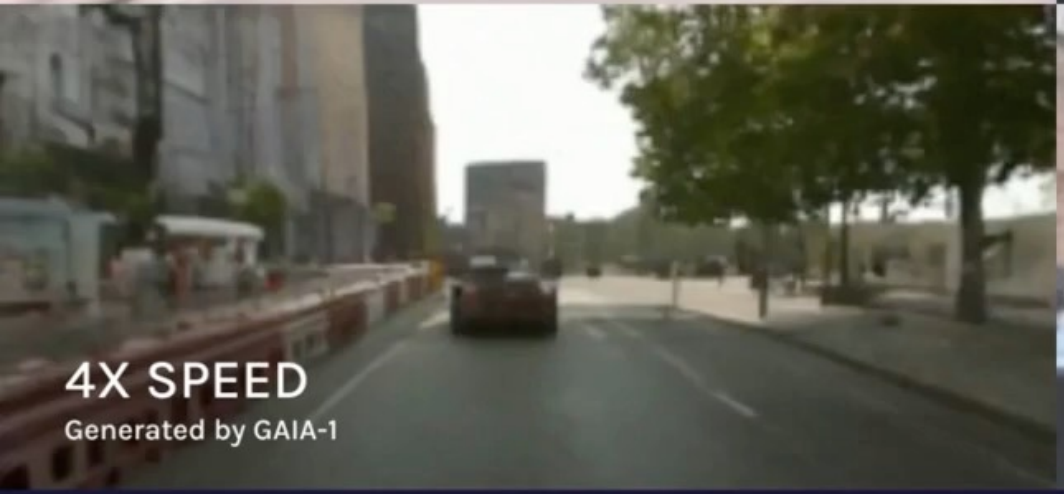
NYU

- X

WAYVE


4X SPEED
Generated by GAIA-1

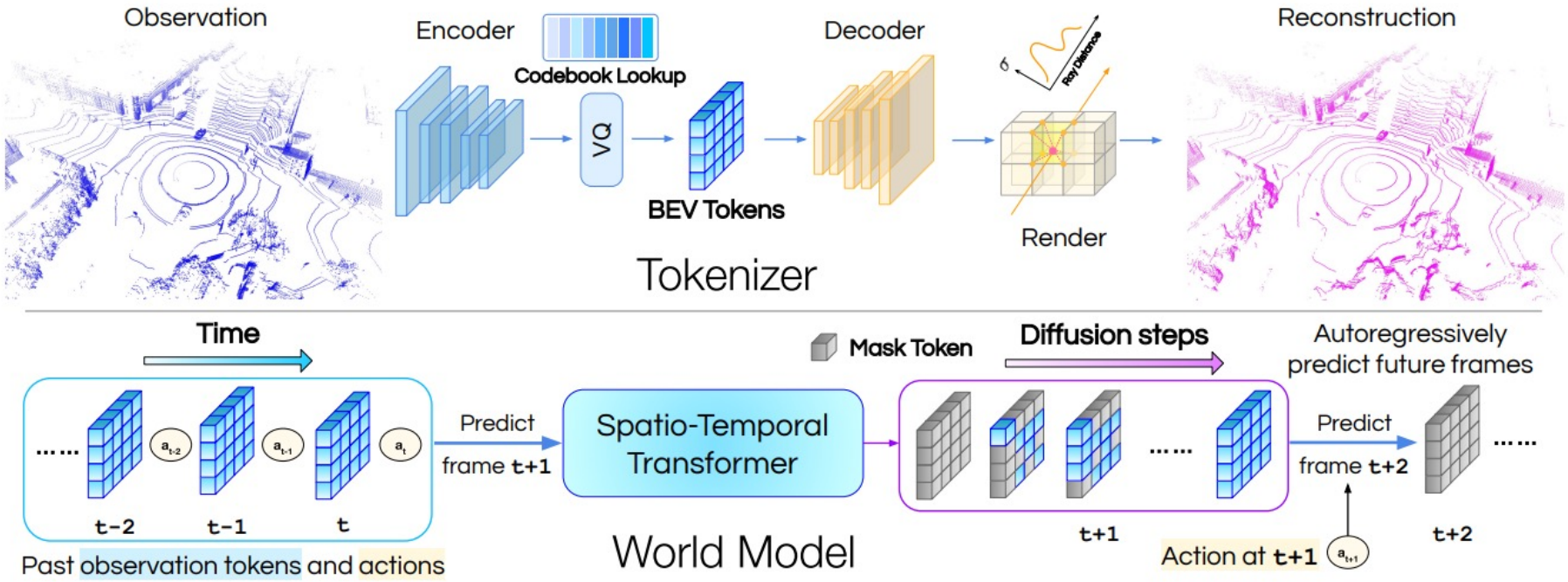

4X SPEED
Generated by GAIA-1


4X SPEED
Generated by GAIA-1


4X SPEED
Generated by GAIA-1

World Model in 3D volume prediction

- Autoregressively predict future 3D point clouds.



Zhang et al. Copilot4D: Learning Unsupervised World Models for Autonomous Driving via Discrete Diffusion. ICLR 2024.

NYU

Summary: World Models

NYU

Summary: World Models

- Explicit object representation
  - Traditional, light weight, instance-specific, hard to learn jointly

NYU

Summary: World Models

- Explicit object representation
  - Traditional, light weight, instance-specific, hard to learn jointly
- Differentiable occupancy, motion field
  - Relatively heavy, spatially grounded, end-to-end learnable

NYU

Summary: World Models

- Explicit object representation
  - Traditional, light weight, instance-specific, hard to learn jointly
- Differentiable occupancy, motion field
  - Relatively heavy, spatially grounded, end-to-end learnable
- Global latent, RNNs, graph landmarks
  - General-purpose, unstructured

NYU

Summary: World Models

- Explicit object representation
  - Traditional, light weight, instance-specific, hard to learn jointly
- Differentiable occupancy, motion field
  - Relatively heavy, spatially grounded, end-to-end learnable
- Global latent, RNNs, graph landmarks
  - General-purpose, unstructured
- Raw video/3D prediction
  - Expensive, good for simulation

NYU

Imitation Learning

- The explicit policy model, supervised learning (behavior cloning)

$$\hat{a} = f_\theta(x) \qquad \mathcal{L} = \min_i \|a_i - \hat{a}\|_2^2 \qquad \mathcal{L} = -\log \hat{a}_j$$

NYU

Imitation Learning

- The explicit policy model, supervised learning (behavior cloning)

$$\hat{a} = f_\theta(x) \qquad \mathcal{L} = \min_i \|a_i - \hat{a}\|_2^2 \qquad \mathcal{L} = -\log \hat{a}_j$$

- Energy-based (cost-based) approach

$$\tau^\star = \operatorname{argmin}_\tau E(x, \tau)$$

$$p(\tau \mid x) = \frac{\exp(E(x,\tau))}{\int_\tau \exp(E(x,\tau))}$$

NYU

Imitation Learning

- The explicit policy model, supervised learning (behavior cloning)

$$\hat{a} = f_\theta(x) \qquad \mathcal{L} = \min_i \|a_i - \hat{a}\|_2^2 \qquad \mathcal{L} = -\log \hat{a}_j$$

- Energy-based (cost-based) approach

$$\tau^\star = \operatorname{argmin}_\tau E(x, \tau)$$

$$p(\tau \mid x) = \frac{\exp(E(x,\tau))}{\int_\tau \exp(E(x,\tau))}$$

- Dataset Aggregation (DAgger)
  - Learned policy may deviate from experts
  - Need to collect more groundtruths

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

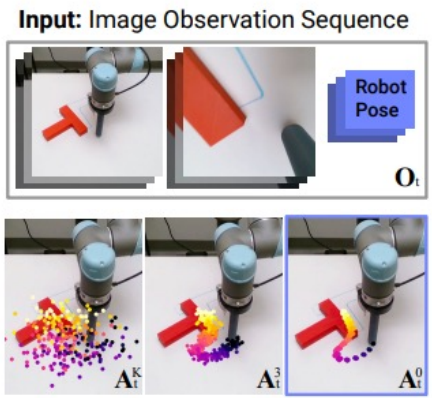
NYU

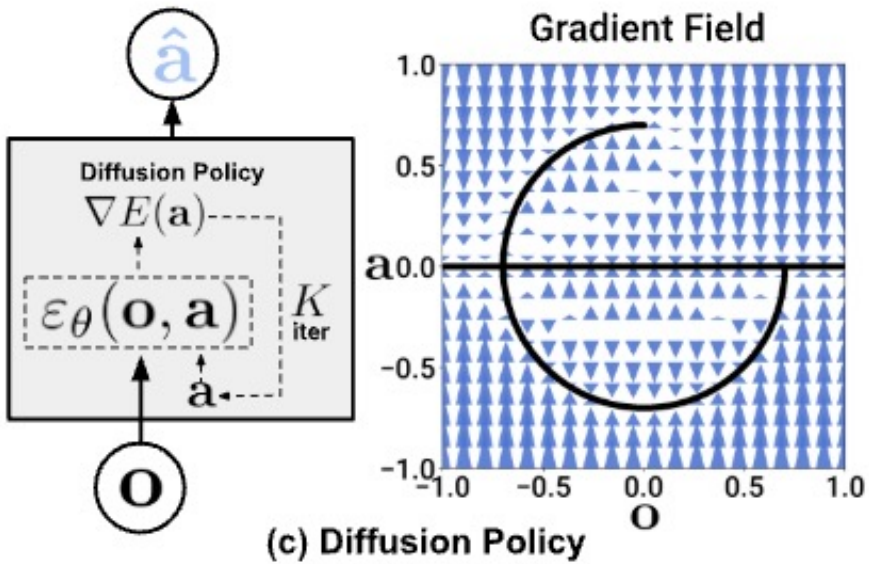
Direct Policy Learning from Diffusion

- Error prediction network is conditioned on observation features.

$$A_t^{k-1} = \alpha(A_t^k - \gamma\epsilon_\theta(O_t, A_t^k, k) + \mathcal{N}(0, \sigma^2 I)).$$

$$\mathcal{L} = MSE(\epsilon^k, \epsilon_\theta(O_t, A_t + \epsilon^k, k)).$$





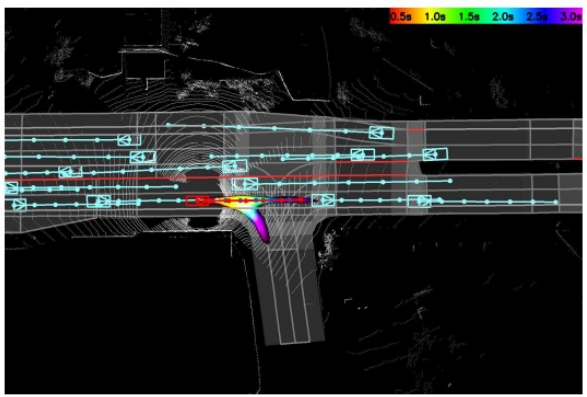
Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. RSS 2023

NYU

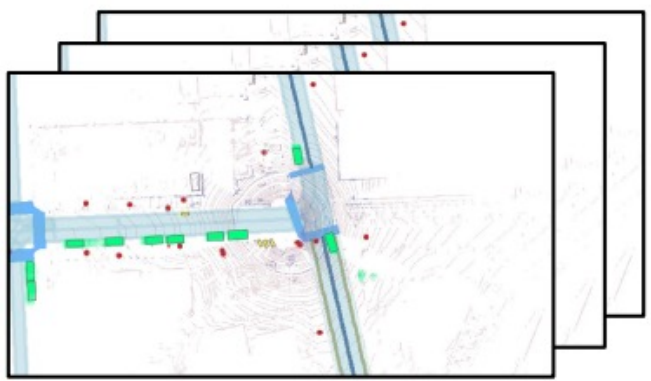
Cost/Value Volume Reasoning

- Interpretability (both costs and planner inputs)

Rasterization



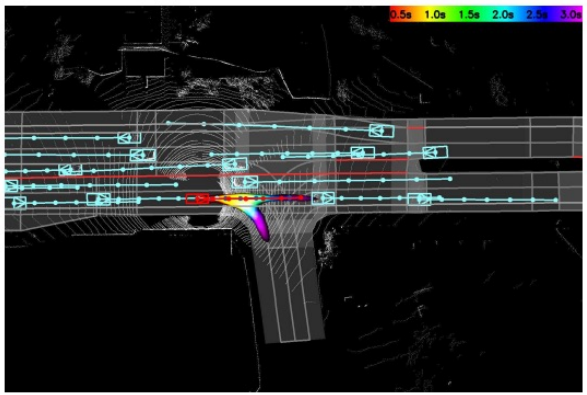
Semantic Occupancy



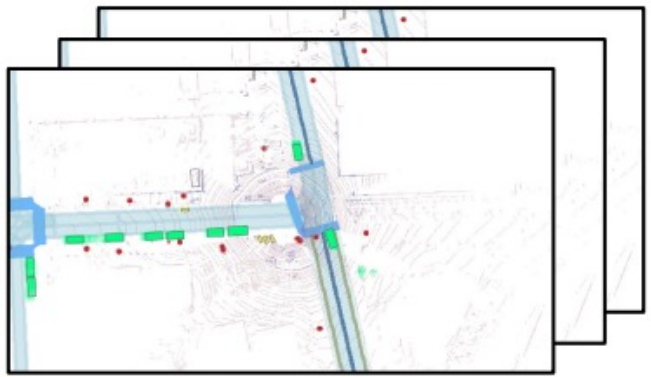
Cost/Value Volume Reasoning

- Interpretability (both costs and planner inputs)
- Use spatial geometry to form cost from explicit objects

Rasterization



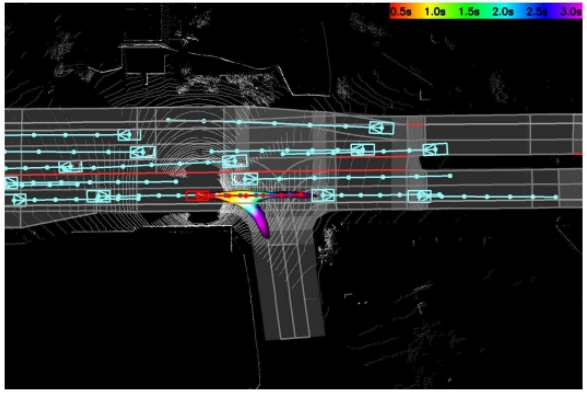
Semantic Occupancy



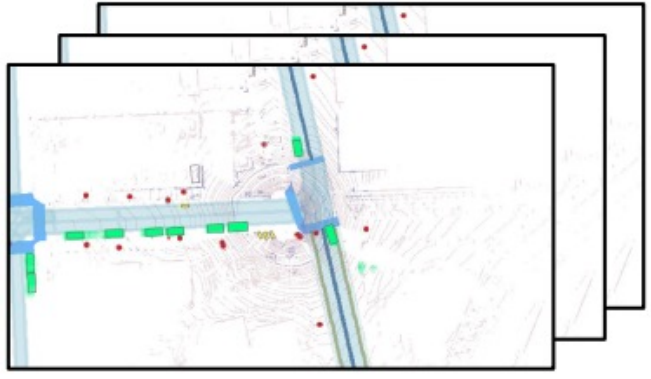
Cost/Value Volume Reasoning

- Interpretability (both costs and planner inputs)
- Use spatial geometry to form cost from explicit objects
- Predict spatial cost volume
  - Rasterize the scene for spatial inputs
  - Predict soft occupancy volumes (present and future)

Rasterization



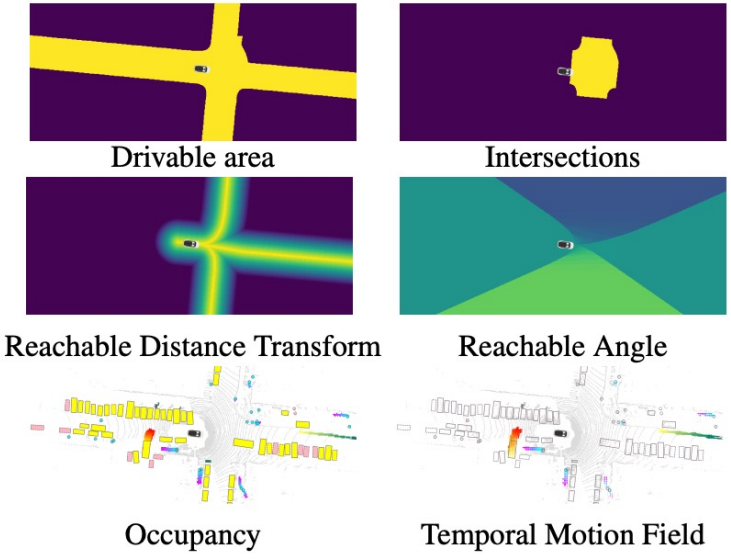
Semantic Occupancy

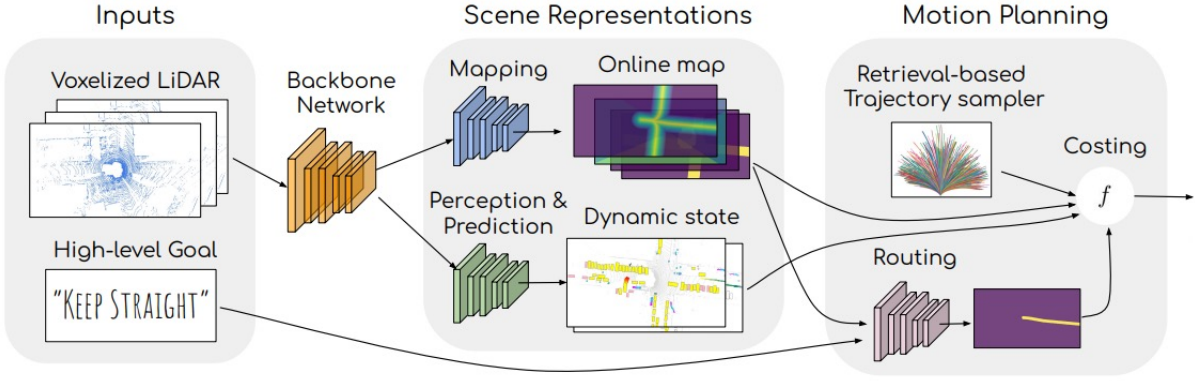


NYU

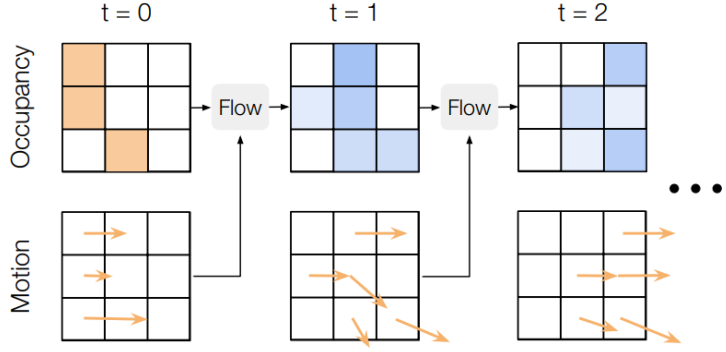
Learning Through Interpretable Predictions

- Semantic occupancy, motion field, mapping, etc. as intermediate predictions.



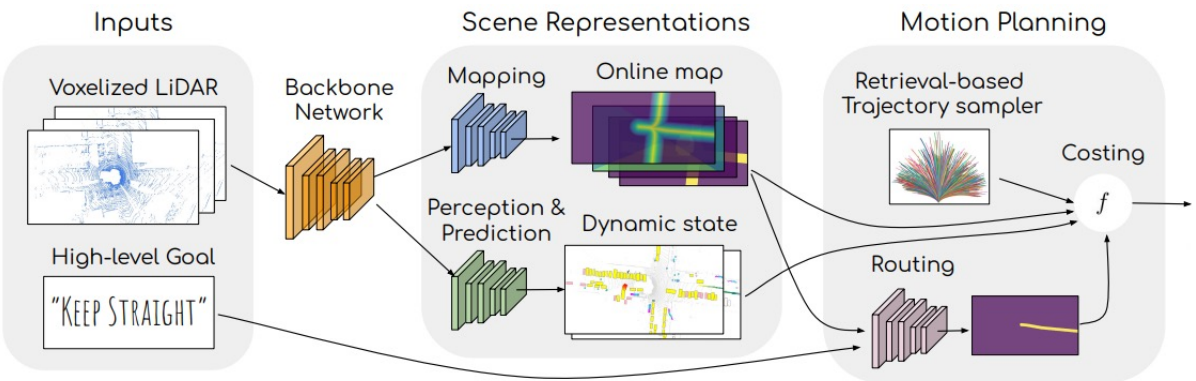


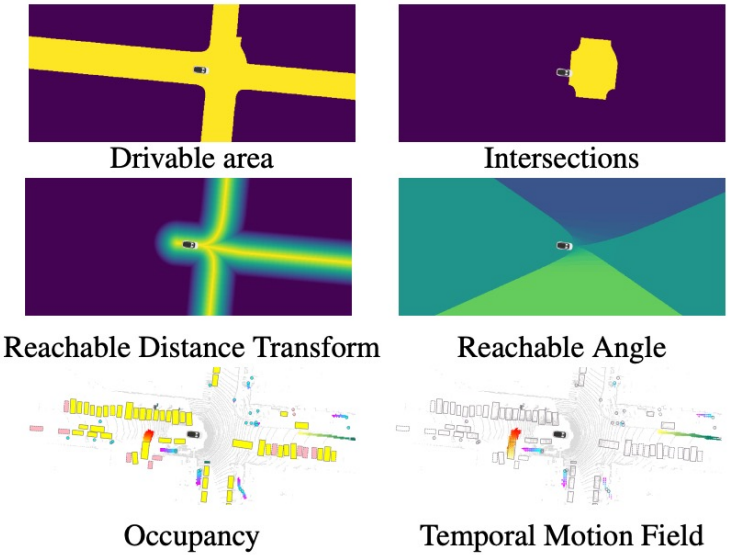
Casas et al. MP3: A Unified Model to Map, Perceive, Predict and Plan. CVPR 2021.

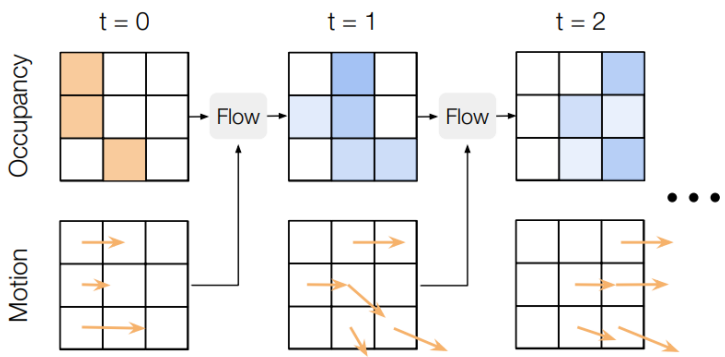


Learning Through Interpretable Predictions

- Semantic occupancy, motion field, mapping, etc. as intermediate predictions.
- Differentiable, supports end-to-end interpretable learning from perception to planning.







Casas et al. MP3: A Unified Model to Map, Perceive, Predict and Plan. CVPR 2021.

Max-Margin Planning with Explicit Cost Volume

- If we have an explicit cost volume, the cost of a trajectory can be directly queried.

Zeng et al. End-to-end Interpretable Neural Motion Planner. CVPR 2019.

NYU

Max-Margin Planning with Explicit Cost Volume

- If we have an explicit cost volume, the cost of a trajectory can be directly queried.
- We can use the max-margin objective to make the groundtruth trajectory have lower costs.

$$\underset{\theta}{\operatorname{argmin}} \sum_{\{(\hat{x}_i^t, \hat{y}_i^t)\}_{i=1\ldots N}} \max_i \sum_{t=1}^{T} C_\theta^t[x_t, y_t] - C_\theta^t[\hat{x}_i^t, \hat{y}_i^t] + d_i^t$$

Zeng et al. End-to-end Interpretable Neural Motion Planner. CVPR 2019.

NYU

Max-Margin Planning with Explicit Cost Volume

- If we have an explicit cost volume, the cost of a trajectory can be directly queried.
- We can use the max-margin objective to make the groundtruth trajectory have lower costs.
- Find the lowest cost trajectory among a batch of samples.

$$\underset{\theta}{\operatorname{argmin}} \sum_{\{(\hat{x}_i^t, \hat{y}_i^t)\}_{i=1\ldots N}} \max_i \sum_{t=1}^{T} C_\theta^t[x_t, y_t] - C_\theta^t[\hat{x}_i^t, \hat{y}_i^t] + d_i^t$$

Zeng et al. End-to-end Interpretable Neural Motion Planner. CVPR 2019.

NYU

Max-Margin Planning with Explicit Cost Volume

- If we have an explicit cost volume, the cost of a trajectory can be directly queried.
- We can use the max-margin objective to make the groundtruth trajectory have lower costs.
- Find the lowest cost trajectory among a batch of samples.
- Low-dimensional/known dynamics problems: External samplers

$$\underset{\theta}{\operatorname{argmin}} \sum_{\{(\hat{x}_i^t, \hat{y}_i^t)\}_{i=1\ldots N}} \max_i \sum_{t=1}^{T} C_\theta^t[x_t, y_t] - C_\theta^t[\hat{x}_i^t, \hat{y}_i^t] + d_i^t$$

Zeng et al. End-to-end Interpretable Neural Motion Planner. CVPR 2019.

NYU

Max-Margin Planning with Explicit Cost Volume

- If we have an explicit cost volume, the cost of a trajectory can be directly queried.
- We can use the max-margin objective to make the groundtruth trajectory have lower costs.
- Find the lowest cost trajectory among a batch of samples.
- Low-dimensional/known dynamics problems: External samplers
- In general, needs to perform optimization (e.g. DP)

$$\underset{\theta}{\operatorname{argmin}} \sum_{\{(\hat{x}_i^t, \hat{y}_i^t)\}_{i=1\dots N}} \max_i \sum_{t=1}^{T} C_\theta^t[x_t, y_t] - C_\theta^t[\hat{x}_i^t, \hat{y}_i^t] + d_i^t$$

Zeng et al. End-to-end Interpretable Neural Motion Planner. CVPR 2019.

NYU

EBM Planning

- Energy-based framework also needs negative samples.

Florence et al. Implicit Behavioral Cloning. CoRL 2021.

NYU

EBM Planning

- Energy-based framework also needs negative samples.
- “Pick” the groundtruth sample among others.

Florence et al. Implicit Behavioral Cloning. CoRL 2021.

NYU

EBM Planning

- Energy-based framework also needs negative samples.
- “Pick” the groundtruth sample among others.
- If there isn’t an external sampler, we can either use autoregressive energy of sampling one dimension at a time, or gradient-based Langevin MCMC.

Florence et al. Implicit Behavioral Cloning. CoRL 2021.

NYU

EBM Planning

- Energy-based framework also needs negative samples.
- “Pick” the groundtruth sample among others.
- If there isn’t an external sampler, we can either use autoregressive energy of sampling one dimension at a time, or gradient-based Langevin MCMC.

Langevin MCMC: $\tilde{\mathbf{y}}_i^k = \tilde{\mathbf{y}}_i^{k-1} - \lambda \left(\frac{1}{2} \nabla_{\mathbf{y}} E_\theta(\mathbf{x}_i, \mathbf{y}_i^{k-1}) + \omega^k \right), \omega^k \sim \mathcal{N}(0, \sigma).$

Florence et al. Implicit Behavioral Cloning. CoRL 2021.

NYU

EBM Planning

- Energy-based framework also needs negative samples.
- “Pick” the groundtruth sample among others.
- If there isn’t an external sampler, we can either use autoregressive energy of sampling one dimension at a time, or gradient-based Langevin MCMC.

Langevin MCMC: $\tilde{\mathbf{y}}_i^k = \tilde{\mathbf{y}}_i^{k-1} - \lambda\left(\frac{1}{2}\nabla_{\mathbf{y}} E_\theta(\mathbf{x}_i, \mathbf{y}_i^{k-1}) + \omega^k\right), \omega^k \sim \mathcal{N}(0, \sigma).$

Loss: $\mathcal{L} = \sum_i -\log(p_\theta(\mathbf{y}_i \mid \mathbf{x}, \{\tilde{\mathbf{y}}_i\}_j) \qquad p_\theta(\mathbf{y}_i \mid \mathbf{x}, \{\tilde{\mathbf{y}}_i\}_j = \frac{e^{-E_\theta(\mathbf{x}_i, \mathbf{y}_i)}}{e^{-E_\theta(\mathbf{x}_i, \mathbf{y}_i)} + \sum_j e^{-E_\theta(\mathbf{x}_i + \tilde{\mathbf{y}}_{i,j})}}$

Florence et al. Implicit Behavioral Cloning. CoRL 2021.

NYU

Value Iteration Networks

- A network design for predicting cost volumes that are grounded from the classic value iteration algorithm.

Tamar et al. Value Iteration Networks. NeurIPS 2016.

NYU

Value Iteration Networks

- A network design for predicting cost volumes that are grounded from the classic value iteration algorithm.
- Classic VI altorithm:

Tamar et al. Value Iteration Networks. NeurIPS 2016.

NYU

Value Iteration Networks

- A network design for predicting cost volumes that are grounded from the classic value iteration algorithm.
- Classic VI altorithm:

$$V^*(s) = \max_\pi V^\pi(s)$$

Tamar et al. Value Iteration Networks. NeurIPS 2016.

NYU

Value Iteration Networks

- A network design for predicting cost volumes that are grounded from the classic value iteration algorithm.
- Classic VI altorithm:

$$V^*(s) = \max_\pi V^\pi(s)$$

$$V^\pi(s) = \mathbb{E}^\pi \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

Tamar et al. Value Iteration Networks. NeurIPS 2016.

NYU

Value Iteration Networks

- A network design for predicting cost volumes that are grounded from the classic value iteration algorithm.
- Classic VI altorithm:

$$V^*(s) = \max_\pi V^\pi(s)$$

$$V^\pi(s) = \mathbb{E}^\pi \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

$$Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s')$$

Tamar et al. Value Iteration Networks. NeurIPS 2016.

NYU

Value Iteration Networks

- A network design for predicting cost volumes that are grounded from the classic value iteration algorithm.
- Classic VI altorithm:

$$V^*(s) = \max_\pi V^\pi(s)$$

$$V^\pi(s) = \mathbb{E}^\pi \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

$$Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s')$$

$$V_{n+1}(s) = \max_a Q_n(s,a)$$

Tamar et al. Value Iteration Networks. NeurIPS 2016.

NYU

Value Iteration Networks

- A network design for predicting cost volumes that are grounded from the classic value iteration algorithm.
- Classic VI altorithm:

$$V^*(s) = \max_\pi V^\pi(s)$$

$$V^\pi(s) = \mathbb{E}^\pi \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

$$Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s')$$

$$V_{n+1}(s) = \max_a Q_n(s,a)$$

$$\pi^*(s) = \operatorname{argmax}_a Q_\infty(s,a)$$

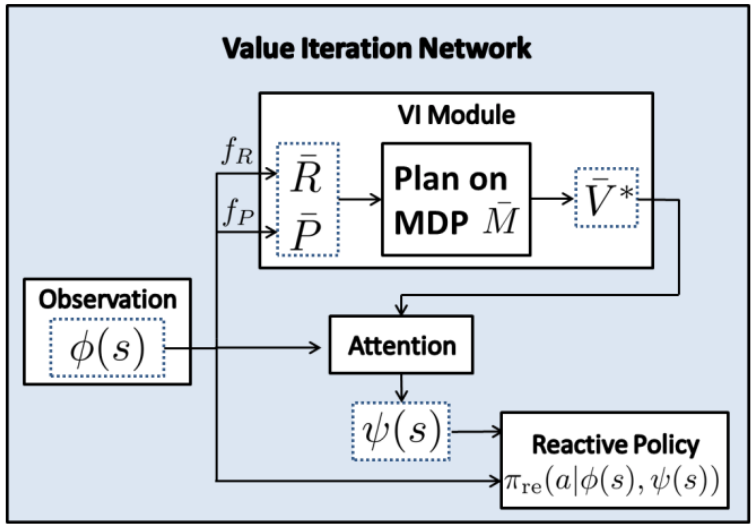
Tamar et al. Value Iteration Networks. NeurIPS 2016.

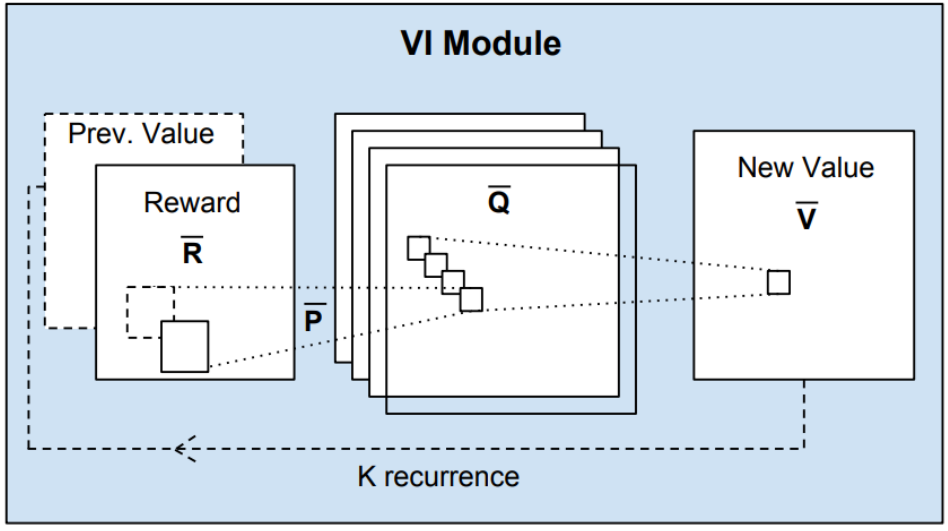
NYU

Value Iteration Networks

- Reward and previous value are fed into a CNN to generate Q of A channels. Transition matrix is convolutional kernel. Then Max-Pooling.

$$Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s') \quad V_{n+1}(s) = \max_a Q_n(s,a)$$





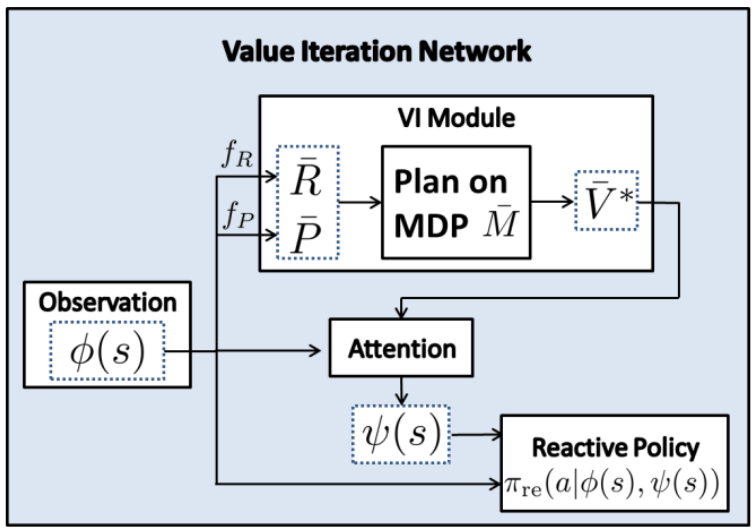
Tamar et al. Value Iteration Networks. NeurIPS 2016.

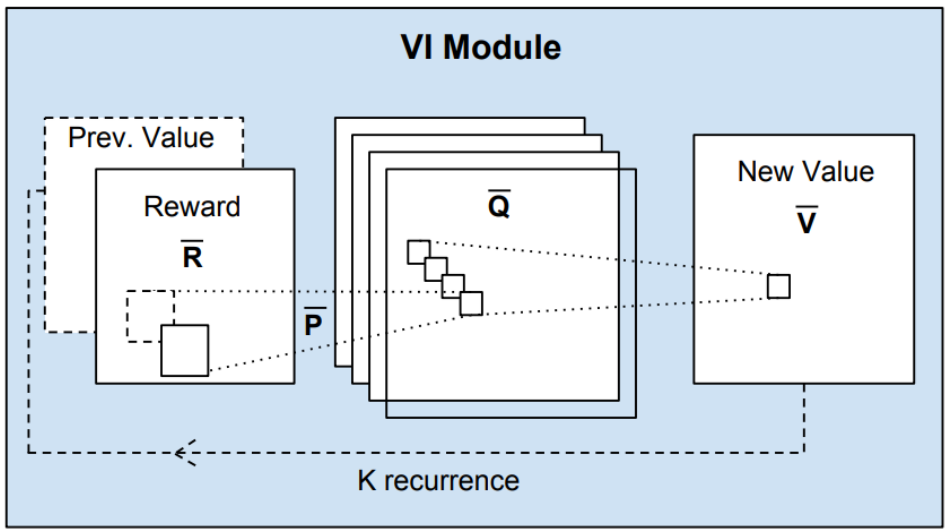
NYU

Value Iteration Networks

- Select the current state and choose an action from softmax.

$$\hat{a} \sim \mathrm{softmax}_a(Q(s,a))$$





Tamar et al. Value Iteration Networks. NeurIPS 2016.

NYU

Value Iteration Networks

- A baseline would be to untie the weights through iterations, more like a feedforward CNN.
- Achieve more training data efficiency by imposing the structure.

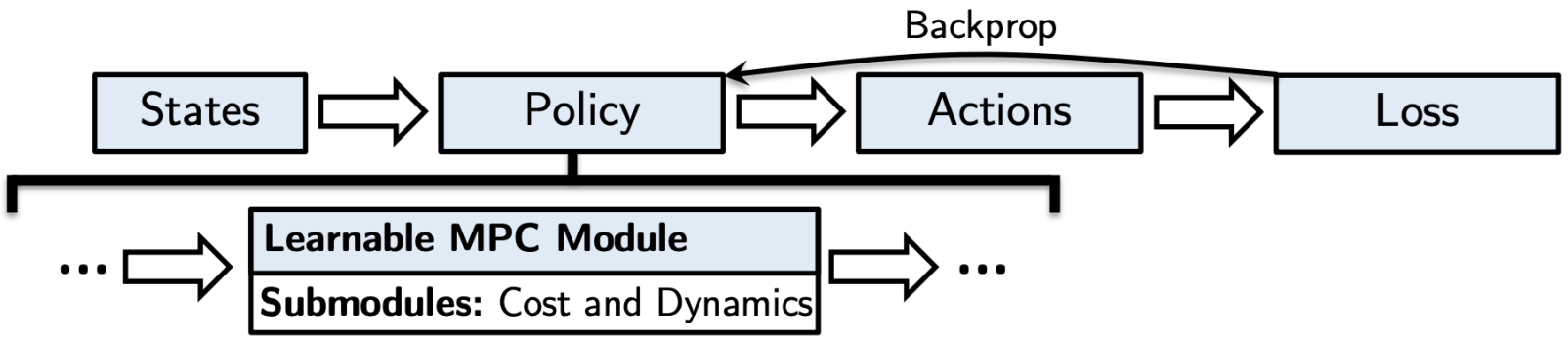
Training data	VIN			VIN Untied Weights		
	Pred. loss	Succ. rate	Traj. diff.	Pred. loss	Succ. rate	Traj. diff.
20%	0.06	98.2%	0.106	0.09	91.9%	0.094
50%	0.05	99.4%	0.018	0.07	95.2%	0.078
100%	0.05	99.3%	0.089	0.05	95.6%	0.068

Tamar et al. Value Iteration Networks. NeurIPS 2016.

NYU

Backprop through Planning

- Treat planning as an end-to-end layer. Can be used for RL/Imitation.

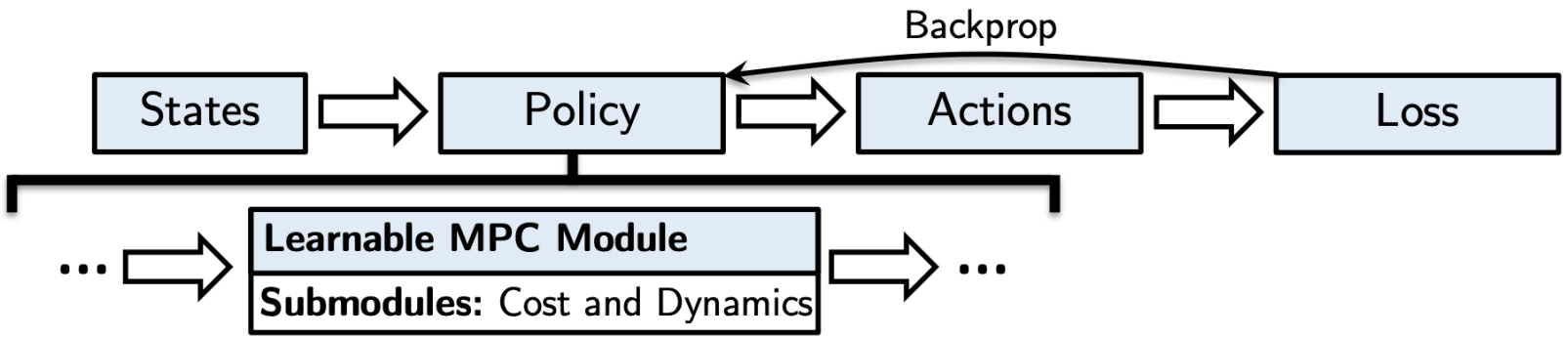


Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Backprop through Planning

- Treat planning as an end-to-end layer. Can be used for RL/Imitation.
- Option 1: Unrolling a finite number of steps

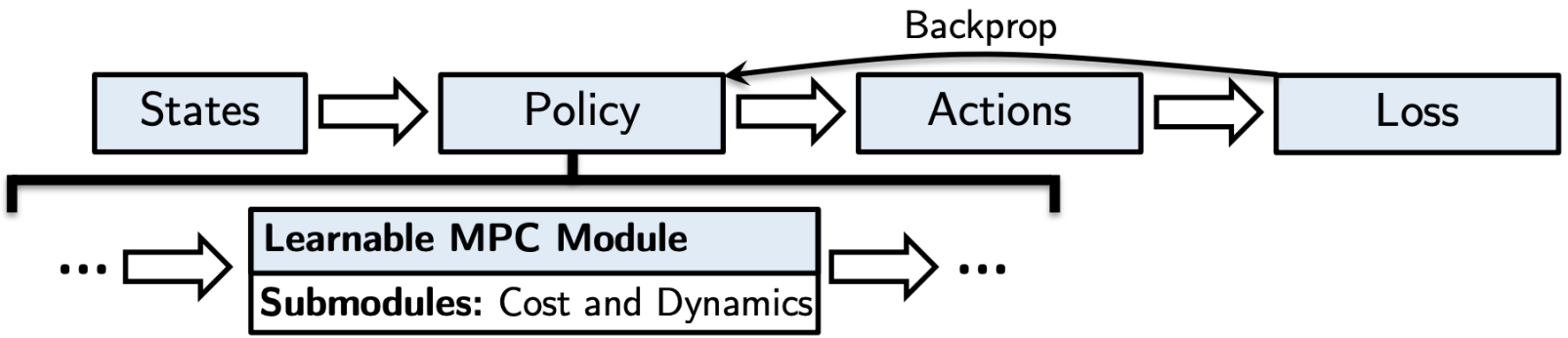


Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Backprop through Planning

- Treat planning as an end-to-end layer. Can be used for RL/Imitation.
- Option 1: Unrolling a finite number of steps
- Option 2: Solve till convergence, backprop for a finite step

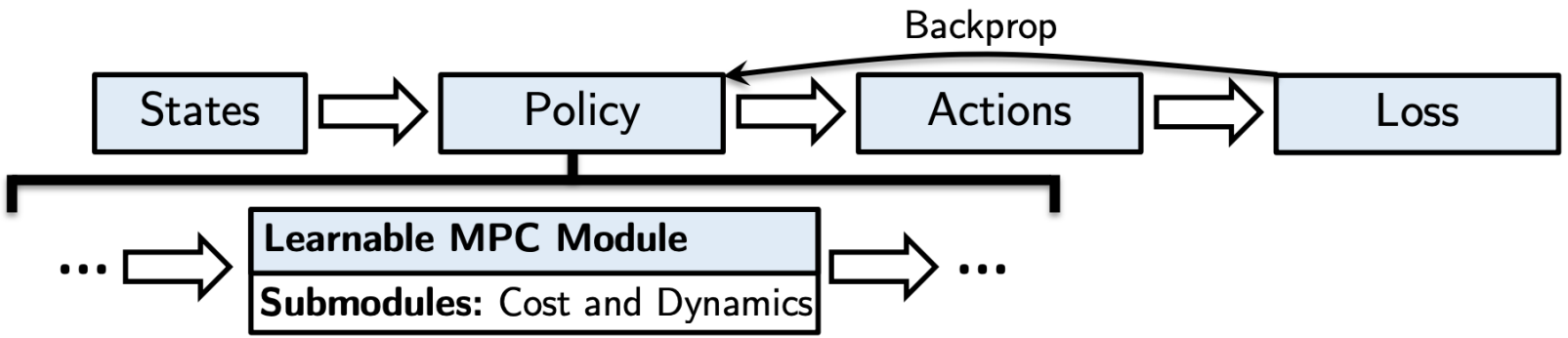


Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Backprop through Planning

- Treat planning as an end-to-end layer. Can be used for RL/Imitation.
- Option 1: Unrolling a finite number of steps
- Option 2: Solve till convergence, backprop for a finite step
- Option 3: Converged at fixed point: Implicit differentiation



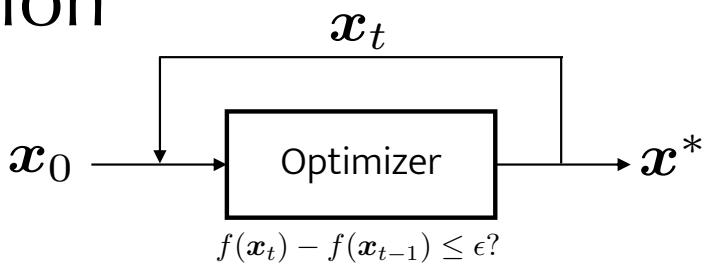
Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Implicit Differentiation

- Unconstrained case

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} f(\boldsymbol{x}; \boldsymbol{\theta}).$$



Liao et al. Reviving and Improving Recurrent Back-Propagation. ICML 2018.

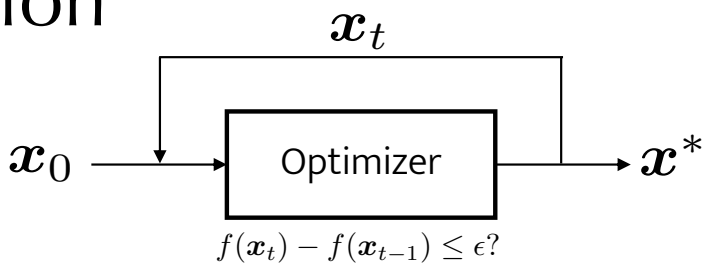
NYU

Implicit Differentiation

- Unconstrained case

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} f(\boldsymbol{x}; \boldsymbol{\theta}).$$

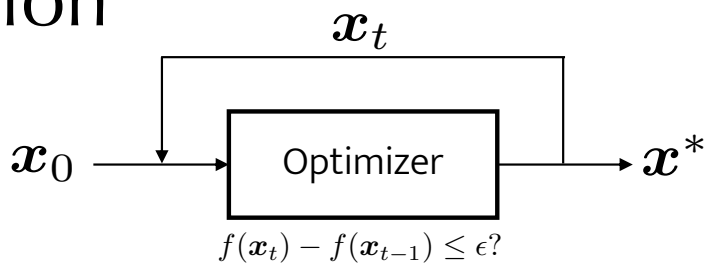
$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$



Liao et al. Reviving and Improving Recurrent Back-Propagation. ICML 2018.

NYU

Implicit Differentiation



- Unconstrained case

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} f(\boldsymbol{x}; \boldsymbol{\theta}).$$

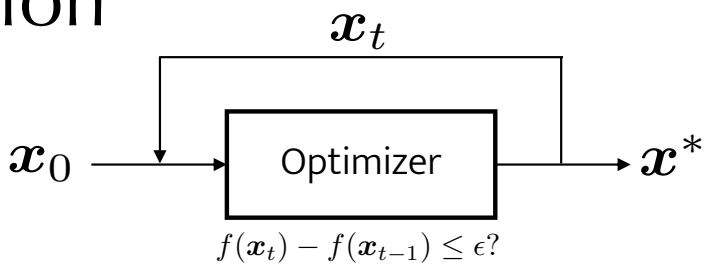
$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

$$\mathbf{0} = \frac{\partial}{\partial \boldsymbol{x}^*} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta}) \frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{\theta}} + \frac{\partial}{\partial \boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

Liao et al. Reviving and Improving Recurrent Back-Propagation. ICML 2018.

NYU

Implicit Differentiation



- Unconstrained case

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} f(\boldsymbol{x}; \boldsymbol{\theta}).$$

$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

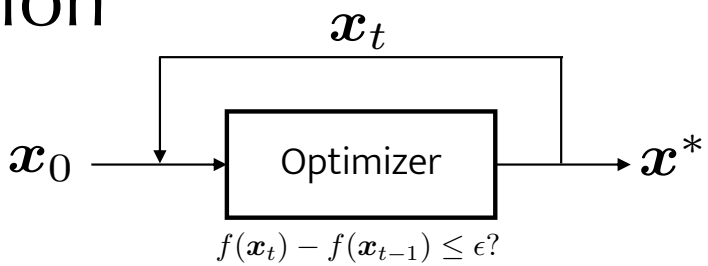
$$\mathbf{0} = \frac{\partial}{\partial \boldsymbol{x}^*} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta}) \frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{\theta}} + \frac{\partial}{\partial \boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

$$\mathbf{0} = H_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta}) \frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{\theta}} + \frac{\partial}{\partial \boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

Liao et al. Reviving and Improving Recurrent Back-Propagation. ICML 2018.

NYU

Implicit Differentiation



- Unconstrained case

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} f(\boldsymbol{x}; \boldsymbol{\theta}).$$

$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

$$\mathbf{0} = \frac{\partial}{\partial \boldsymbol{x}^*} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta}) \frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{\theta}} + \frac{\partial}{\partial \boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

$$\mathbf{0} = H_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta}) \frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{\theta}} + \frac{\partial}{\partial \boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

$$\frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{\theta}} = H_{f,x^*}(\boldsymbol{x}^*; \boldsymbol{\theta})^{-1} \frac{\partial}{\partial \boldsymbol{\theta}} J_{f,x^*}(\boldsymbol{x}^*; \boldsymbol{\theta}).$$

Liao et al. Reviving and Improving Recurrent Back-Propagation. ICML 2018.

NYU

Implicit Differentiation

- How to compute Hessian inverse vector product?

Liao et al. Reviving and Improving Recurrent Back-Propagation. ICML 2018.

NYU

Implicit Differentiation

- How to compute Hessian inverse vector product?
- Conjugate gradient, solve $A\boldsymbol{x} = \boldsymbol{b}$

Conjugate Gradient Method

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
$\alpha_k := \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
$\beta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result

Liao et al. Reviving and Improving Recurrent Back-Propagation. ICML 2018.

NYU

Implicit Differentiation

- How to compute Hessian inverse vector product?
- Conjugate gradient, solve $A\boldsymbol{x} = \boldsymbol{b}$
- Neumann series (finite truncation)

$$(I - A)^{-1} = \sum_{k=0}^{\infty} A^k.$$

  - Same as backprop the last K steps (Option 2).
  - Memory savings.

Conjugate Gradient Method

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
  $\alpha_k := \dfrac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
  $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
  $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
  if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
  $\beta_k := \dfrac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
  $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
  $k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result

Liao et al. Reviving and Improving Recurrent Back-Propagation. ICML 2018.

NYU

Differentiable LQR

- Now add linear equality constraints on the dynamics and initialization.

$$\tau_{1:T} = \{x_t, u_t\}_{1:T}$$

$$\underset{\tau_{1:T}}{\text{argmin}} \sum_{t=1}^{T} \frac{1}{2} \tau_t^\top C_t \tau_t$$

$$\text{subject to } x_{t+1} = F_t \tau_t + f_t, x_1 = x_{\text{init}}.$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- Now add linear equality constraints on the dynamics and initialization.

$$\tau_{1:T} = \{x_t, u_t\}_{1:T}$$

- Chain rule: $\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial \tau^\star_{1:T}} \frac{\partial \tau^\star_{1:T}}{\partial \theta}$.

$$\underset{\tau_{1:T}}{\text{argmin}} \sum_{t=1}^{T} \frac{1}{2} \tau_t^\top C_t \tau_t$$

$$\text{subject to } x_{t+1} = F_t \tau_t + f_t, x_1 = x_{\text{init}}.$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- Now add linear equality constraints on the dynamics and initialization.

$$\tau_{1:T} = \{x_t, u_t\}_{1:T}$$

- Chain rule: $\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial \tau^\star_{1:T}} \frac{\partial \tau^\star_{1:T}}{\partial \theta}$.

$$\underset{\tau_{1:T}}{\text{argmin}} \sum_{t=1}^{T} \frac{1}{2} \tau_t^\top C_t \tau_t$$

$$\text{subject to } x_{t+1} = F_t \tau_t + f_t, x_1 = x_{\text{init}}.$$

General QP:

$$\boldsymbol{x}^* = \text{argmin} \frac{1}{2} \boldsymbol{x}^\top Q \boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x}$$

$$\text{subject to } \; A\boldsymbol{x} = \boldsymbol{b}.$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- Now add linear equality constraints on the dynamics and initialization.

$$\tau_{1:T} = \{x_t, u_t\}_{1:T}$$

- Chain rule: $\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial \tau^\star_{1:T}} \frac{\partial \tau^\star_{1:T}}{\partial \theta}$.
- KKT:

$$\underset{\tau_{1:T}}{\text{argmin}} \sum_{t=1}^{T} \frac{1}{2} \tau_t^\top C_t \tau_t$$

$$\text{subject to } x_{t+1} = F_t \tau_t + f_t, x_1 = x_{\text{init}}.$$

General QP:

$$\boldsymbol{x}^* = \text{argmin} \frac{1}{2} \boldsymbol{x}^\top Q \boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x}$$

$$\text{subject to } \; A\boldsymbol{x} = \boldsymbol{b}.$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- Now add linear equality constraints on the dynamics and initialization.

$$\tau_{1:T} = \{x_t, u_t\}_{1:T}$$

- Chain rule: $\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial \tau^\star_{1:T}} \frac{\partial \tau^\star_{1:T}}{\partial \theta}$.
- KKT:

$$\begin{bmatrix} Q & A^\top \\ A & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} -\boldsymbol{c} \\ \boldsymbol{b} \end{bmatrix}$$

$$K \begin{bmatrix} \boldsymbol{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = v.$$

$$\underset{\tau_{1:T}}{\text{argmin}} \sum_{t=1}^{T} \frac{1}{2} \tau_t^\top C_t \tau_t$$

$$\text{subject to } x_{t+1} = F_t \tau_t + f_t, x_1 = x_{\text{init}}.$$

General QP:

$$\boldsymbol{x}^* = \text{argmin} \frac{1}{2} \boldsymbol{x}^\top Q \boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x}$$

$$\text{subject to } \; A\boldsymbol{x} = \boldsymbol{b}.$$

In classic LQR solver, the Riccati recursion solves this linear system.

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- Apply differentiation

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(K\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}\right)=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}.$$

$$\frac{\mathrm{d}K}{\mathrm{d}\boldsymbol{\theta}}\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}+K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\theta}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{\theta}}\end{bmatrix}=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- Apply differentiation

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(K\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}\right)=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}.$$

$$\frac{\mathrm{d}K}{\mathrm{d}\boldsymbol{\theta}}\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}+K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\theta}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{\theta}}\end{bmatrix}=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}$$

$$K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\theta}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{\theta}}\end{bmatrix}=K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{b}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}Q} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}A}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{b}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}Q} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}A}\end{bmatrix}=\begin{bmatrix}-I & 0 & -x^* & -\lambda^*\\ 0 & I & 0 & -x^*\end{bmatrix}$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- Apply differentiation

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(K\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}\right)=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}.$$

$$\frac{\mathrm{d}K}{\mathrm{d}\boldsymbol{\theta}}\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}+K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\theta}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{\theta}}\end{bmatrix}=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}$$

$$K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\theta}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{\theta}}\end{bmatrix}=K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{b}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}Q} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}A}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{b}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}Q} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}A}\end{bmatrix}=\begin{bmatrix}-I & 0 & -x^* & -\lambda^*\\ 0 & I & 0 & -x^*\end{bmatrix}$$

$$K\frac{\partial\ell}{\partial\boldsymbol{z}^*}\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{b}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{b}}\end{bmatrix}=\begin{bmatrix}-\frac{\partial\ell}{\partial\boldsymbol{x}^*}\\ \mathbf{0}\end{bmatrix}$$

$$K\boldsymbol{d}^*=\begin{bmatrix}-\frac{\partial\ell}{\partial\boldsymbol{x}^*}\\ \mathbf{0}\end{bmatrix}$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- Apply differentiation

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(K\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}\right)=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}.$$

$$\frac{\mathrm{d}K}{\mathrm{d}\boldsymbol{\theta}}\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}+K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\theta}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{\theta}}\end{bmatrix}=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}$$

$$K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\theta}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{\theta}}\end{bmatrix}=K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{b}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}Q} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}A}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{b}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}Q} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}A}\end{bmatrix}=\begin{bmatrix}-I & 0 & -x^* & -\lambda^*\\ 0 & I & 0 & -x^*\end{bmatrix}$$

$$K\frac{\partial\ell}{\partial\boldsymbol{z}^*}\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{b}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{b}}\end{bmatrix}=\begin{bmatrix}-\frac{\partial\ell}{\partial\boldsymbol{x}^*}\\ \mathbf{0}\end{bmatrix}$$

$$K\boldsymbol{d}^*=\begin{bmatrix}-\frac{\partial\ell}{\partial\boldsymbol{x}^*}\\ \mathbf{0}\end{bmatrix}$$

- Equivalent QP:

$$\boldsymbol{d}^*=\underset{\boldsymbol{d}}{\operatorname{argmin}}\ \frac{1}{2}\boldsymbol{d}^\top Q\boldsymbol{d}+\frac{\partial\ell}{\partial\boldsymbol{x}^*}^\top\boldsymbol{d},$$
$$\text{subject to}\quad A\boldsymbol{d}=\mathbf{0}.$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- Apply differentiation

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(K\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}\right)=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}.$$

$$\frac{\mathrm{d}K}{\mathrm{d}\boldsymbol{\theta}}\begin{bmatrix}\boldsymbol{x}^*\\ \boldsymbol{\lambda}^*\end{bmatrix}+K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\theta}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{\theta}}\end{bmatrix}=\frac{\mathrm{d}v}{\mathrm{d}\boldsymbol{\theta}}$$

$$K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\theta}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{\theta}}\end{bmatrix}=K\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{b}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}Q} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}A}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{b}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}Q} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}A}\end{bmatrix}=\begin{bmatrix}-I & 0 & -x^* & -\lambda^*\\ 0 & I & 0 & -x^*\end{bmatrix}$$

$$K\frac{\partial\ell}{\partial\boldsymbol{z}^*}\begin{bmatrix}\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{b}}\\ \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{c}} & \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\boldsymbol{b}}\end{bmatrix}=\begin{bmatrix}-\frac{\partial\ell}{\partial\boldsymbol{x}^*}\\ \mathbf{0}\end{bmatrix}$$

$$K\boldsymbol{d}^*=\begin{bmatrix}-\frac{\partial\ell}{\partial\boldsymbol{x}^*}\\ \mathbf{0}\end{bmatrix}$$

- Equivalent QP:

$$\boldsymbol{d}^*=\underset{\boldsymbol{d}}{\operatorname{argmin}}\ \frac{1}{2}\boldsymbol{d}^\top Q\boldsymbol{d}+\frac{\partial\ell}{\partial\boldsymbol{x}^*}^\top\boldsymbol{d},\qquad \text{subject to}\quad A\boldsymbol{d}=\mathbf{0}.$$

$$\frac{\partial\ell}{\partial Q}=\frac{1}{2}(\boldsymbol{d}_{\boldsymbol{x}}^*\otimes\boldsymbol{x}^*+\boldsymbol{x}^*\otimes\boldsymbol{d}_{\boldsymbol{x}}^*)$$

$$\frac{\partial\ell}{\partial A}=\boldsymbol{d}_{\boldsymbol{\lambda}}^*\otimes\boldsymbol{x}^*+\boldsymbol{\lambda}^*\otimes\boldsymbol{d}_{\boldsymbol{x}}^*.$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable LQR

- The backward pass can also be formulated as a LQR problem.
- Swap c to $\nabla_{\tau^\star}\ell$ and f to 0.

Module 1 Differentiable LQR *(The LQR algorithm is defined in appendix A)*

Input: Initial state x_{init}
Parameters: $\theta = \{C, c, F, f\}$

Forward Pass:

1: $\tau^\star_{1:T} = \text{LQR}_T(x_{\text{init}}; C, c, F, f)$ ▷ Solve (2)
2: Compute $\lambda^\star_{1:T}$ with (7)

Backward Pass:

1: $d^\star_{\tau_{1:T}} = \text{LQR}_T(0; C, \nabla_{\tau^\star}\ell, F, 0)$ ▷ Solve (9), ideally reusing the factorizations from the forward pass
2: Compute $d^\star_{\lambda_{1:T}}$ with (7)
3: Compute the derivatives of ℓ with respect to C, c, F, f, and x_{init} with (8)

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable MPC

- What about general MPC?

$$\underset{x_{1:T} \in \mathcal{X}, u_{1:T} \in \mathcal{U}}{\operatorname{argmin}} \sum_{t=1}^{T} C_t(x_t, u_t)$$

$$\text{subject to } x_{t+1} = f(x_t, u_t), x_1 = x_{\text{init}}.$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable MPC

- What about general MPC?

$$\underset{x_{1:T} \in \mathcal{X}, u_{1:T} \in \mathcal{U}}{\operatorname{argmin}} \sum_{t=1}^{T} C_t(x_t, u_t)$$

$$\text{subject to } x_{t+1} = f(x_t, u_t), x_1 = x_{\text{init}}.$$

- Use Taylor expansion to approximate.

$$\tilde{C}^i_{\theta,t} = C_{\theta,t}(\tau^i_t) + {p^i_t}^\top (\tau_t - \tau^i_t) + \frac{1}{2}(\tau_t - \tau^i_t)^\top H^i_t (\tau_t - \tau^i_t).$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable MPC

- What about general MPC?

$$\underset{x_{1:T} \in \mathcal{X}, u_{1:T} \in \mathcal{U}}{\operatorname{argmin}} \sum_{t=1}^{T} C_t(x_t, u_t)$$

$$\text{subject to } x_{t+1} = f(x_t, u_t), x_1 = x_{\text{init}}.$$

- Use Taylor expansion to approximate.

$$\tilde{C}^i_{\theta,t} = C_{\theta,t}(\tau^i_t) + {p^i_t}^\top (\tau_t - \tau^i_t) + \tfrac{1}{2}(\tau_t - \tau^i_t)^\top H^i_t (\tau_t - \tau^i_t).$$

- Fixed point iteration.

$$\tau^{i+1} = \operatorname{argmin}_\tau \textstyle\sum_t^T \tilde{C}^i_t(\tau^i_t).$$

Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Differentiable MPC

- What about general MPC?

$$\underset{x_{1:T} \in \mathcal{X}, u_{1:T} \in \mathcal{U}}{\operatorname{argmin}} \sum_{t=1}^{T} C_t(x_t, u_t)$$

$$\text{subject to } x_{t+1} = f(x_t, u_t), x_1 = x_{\text{init}}.$$

- Use Taylor expansion to approximate.

$$\tilde{C}^i_{\theta,t} = C_{\theta,t}(\tau^i_t) + {p^i_t}^\top (\tau_t - \tau^i_t) + \tfrac{1}{2}(\tau_t - \tau^i_t)^\top H^i_t (\tau_t - \tau^i_t).$$

- Fixed point iteration.

$$\tau^{i+1} = \operatorname{argmin}_\tau \textstyle\sum_t^T \tilde{C}^i_t(\tau^i_t).$$

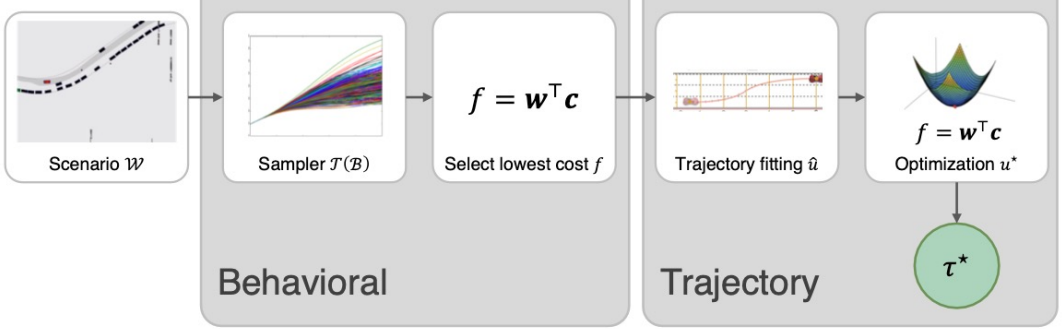
- Backward only depends on final quadratic approximation.

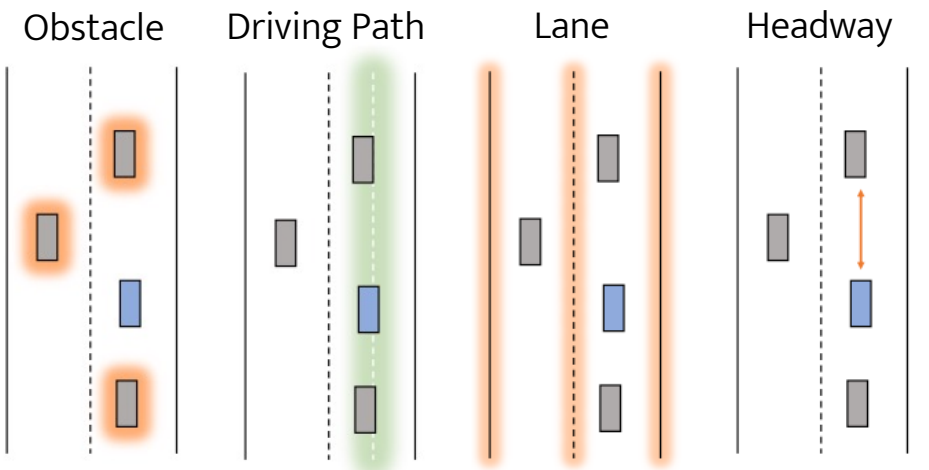
Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Behavioral vs. Trajectory Planning

- Gradient-based optimization provides a locally optimized trajectory.



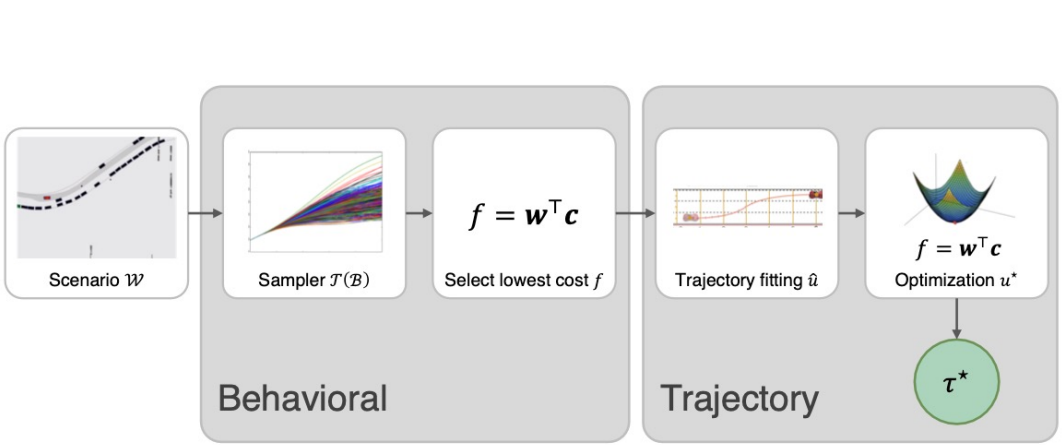


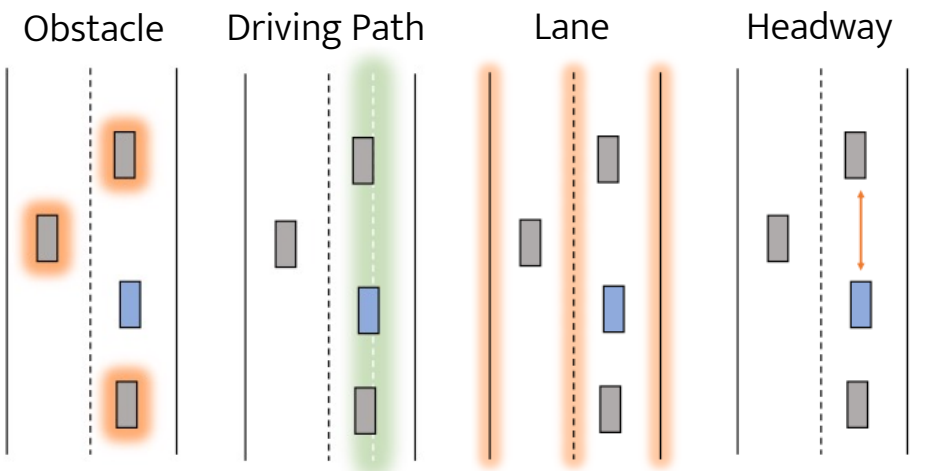
Sadat et al. Jointly Learnable Behavior and Trajectory Planning for Self-Driving Vehicles. IROS 2019.

NYU

Behavioral vs. Trajectory Planning

- Gradient-based optimization provides a locally optimized trajectory.
- Samples may be needed for reasoning global structure.



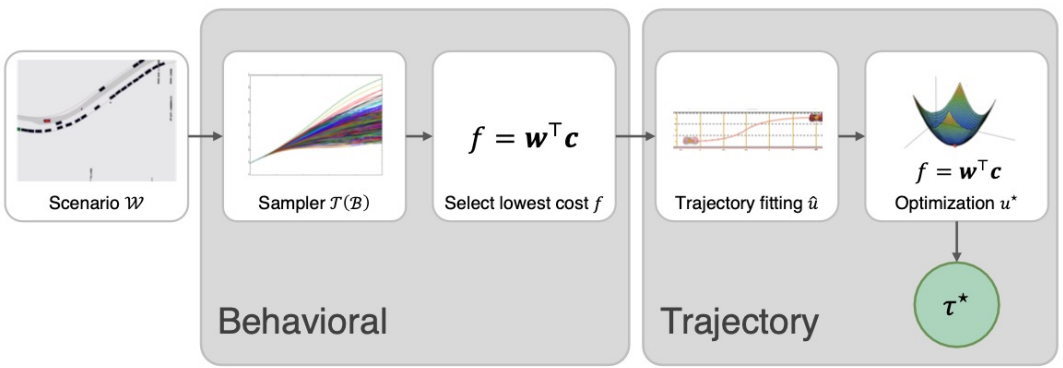


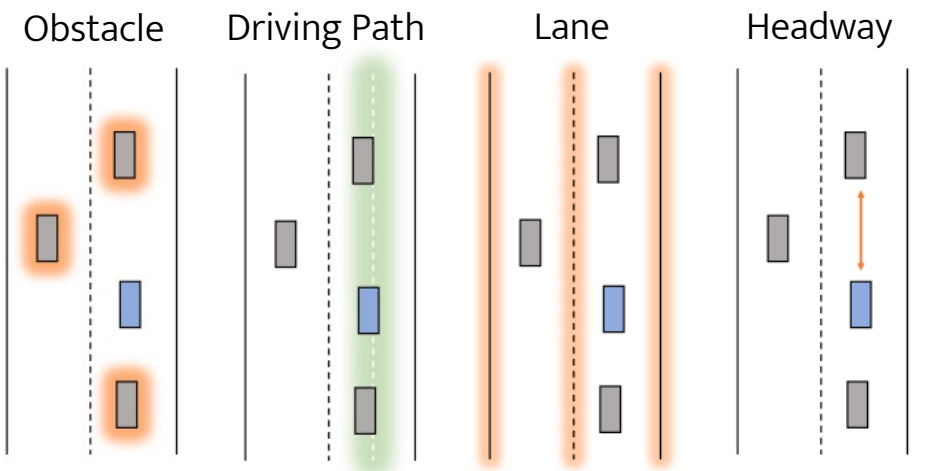
Sadat et al. Jointly Learnable Behavior and Trajectory Planning for Self-Driving Vehicles. IROS 2019.

NYU

Behavioral vs. Trajectory Planning

- Gradient-based optimization provides a locally optimized trajectory.
- Samples may be needed for reasoning global structure.
- Can learn together using the same learned costs.





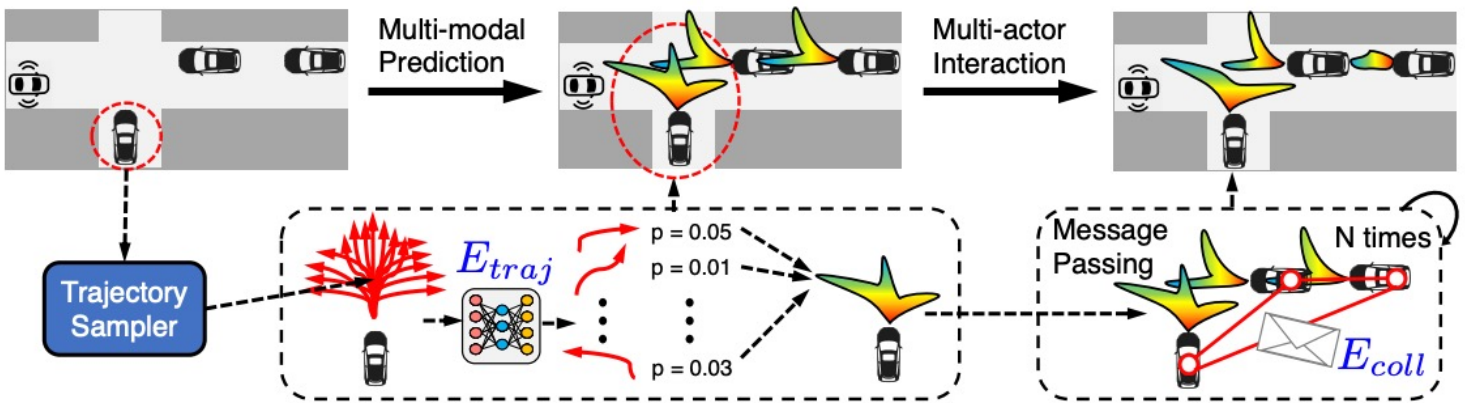
Sadat et al. Jointly Learnable Behavior and Trajectory Planning for Self-Driving Vehicles. IROS 2019.

NYU

Planning with Social Reasoning

- Jointly reason the future trajectories of multiple agents as an energy-based graphical model.

$$p(\mathbf{s}_1, \ldots, \mathbf{s}_N \mid \mathbf{X}) = \frac{1}{Z} \exp(-E_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_N) \mid \mathbf{X})$$



Zeng et al. DSDNet: Deep Structured Self-Driving Network. ECCV 2020.

NYU

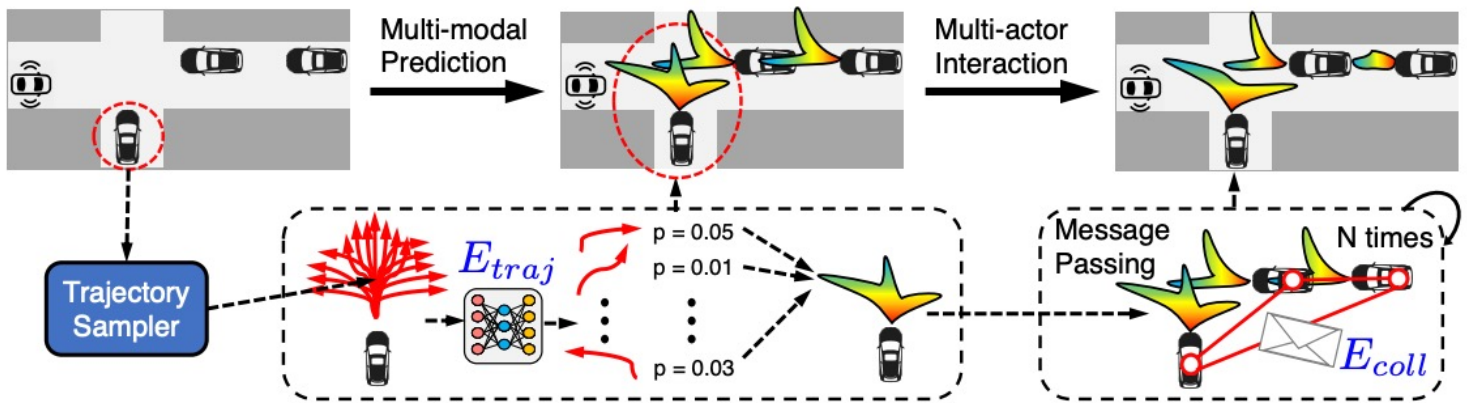
Planning with Social Reasoning

- Jointly reason the future trajectories of multiple agents as an energy-based graphical model.
- Trajectory Goodness + Collision.

$$p(\mathbf{s}_1, \ldots, \mathbf{s}_N \mid \mathbf{X}) = \frac{1}{Z} \exp(-E_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_N) \mid \mathbf{X})$$

$$\sum_i E_\theta(s_i \mid X) + \sum_{i \neq j} E(\mathbf{s}_i, \mathbf{s}_j)$$

$$E(\mathbf{s}_i, \mathbf{s}_j) = \gamma \quad \text{if } \mathbf{s}_i \text{ collides } \mathbf{s}_j$$



Zeng et al. DSDNet: Deep Structured Self-Driving Network. ECCV 2020.

NYU

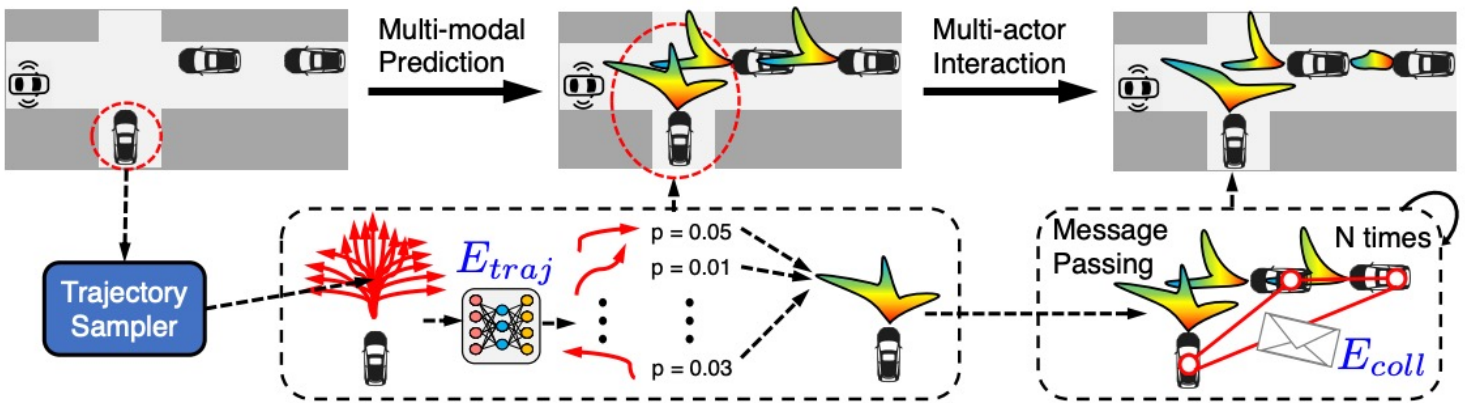
Planning with Social Reasoning

- Jointly reason the future trajectories of multiple agents as an energy-based graphical model.
- Trajectory Goodness + Collision.
- Batch of trajectory samples.

$$p(\mathbf{s}_1, \ldots, \mathbf{s}_N \mid \mathbf{X}) = \frac{1}{Z} \exp(-E_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_N) \mid \mathbf{X})$$

$$\sum_i E_\theta(s_i \mid X) + \sum_{i \neq j} E(\mathbf{s}_i, \mathbf{s}_j)$$

$$E(\mathbf{s}_i, \mathbf{s}_j) = \gamma \quad \text{if } \mathbf{s}_i \text{ collides } \mathbf{s}_j$$



Zeng et al. DSDNet: Deep Structured Self-Driving Network. ECCV 2020.

NYU

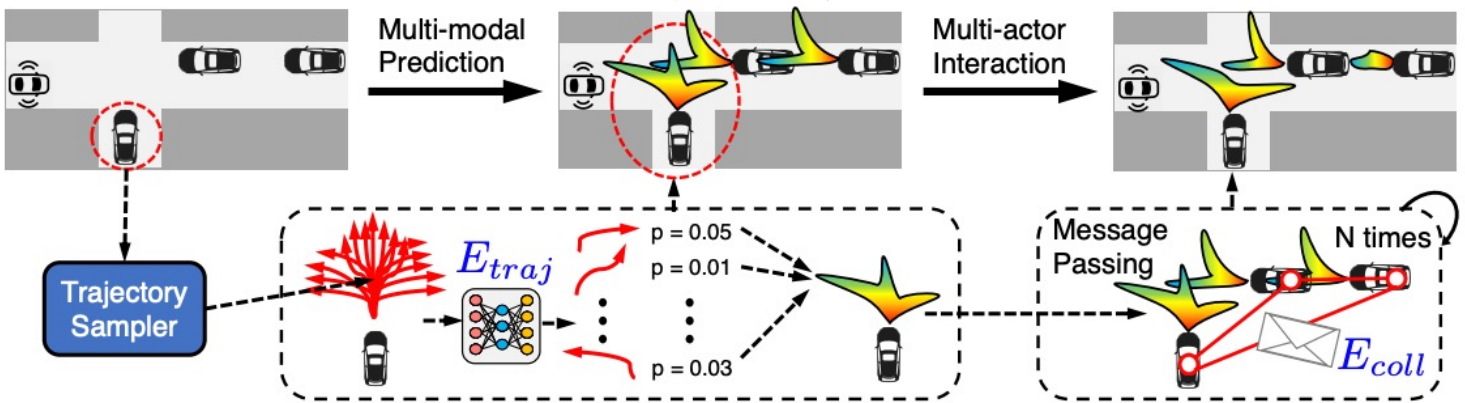
Planning with Social Reasoning

- Jointly reason the future trajectories of multiple agents as an energy-based graphical model.
- Trajectory Goodness + Collision.
- Batch of trajectory samples.
- Classification of groundtruth trajectory.

$$p(\mathbf{s}_1, \ldots, \mathbf{s}_N \mid \mathbf{X}) = \frac{1}{Z} \exp(-E_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_N) \mid \mathbf{X})$$

$$\sum_i E_\theta(s_i \mid X) + \sum_{i \neq j} E(\mathbf{s}_i, \mathbf{s}_j)$$

$$E(\mathbf{s}_i, \mathbf{s}_j) = \gamma \quad \text{if } \mathbf{s}_i \text{ collides } \mathbf{s}_j$$



Zeng et al. DSDNet: Deep Structured Self-Driving Network. ECCV 2020.

NYU

Summary: End-to-End Planning

- Direct Policy Prediction
  - Condition perception features into the model
  - Use of diffusion models

NYU

Summary: End-to-End Planning

- Direct Policy Prediction
  - Condition perception features into the model
  - Use of diffusion models
- Cost Learning (IRL) from Experts
  - Max-margin, max-entropy/EBM
  - Need negative samples
  - Can be combined with efficient external samplers
  - Cost volume prediction: parametric + non-parametric

NYU

Summary: End-to-End Planning

- Direct Policy Prediction
  - Condition perception features into the model
  - Use of diffusion models
- Cost Learning (IRL) from Experts
  - Max-margin, max-entropy/EBM
  - Need negative samples
  - Can be combined with efficient external samplers
  - Cost volume prediction: parametric + non-parametric
- Differentiable Planner
  - Backprop through local optimization
  - Can be memory efficient, implicit differentiation

NYU