DS-GA.3001 Embodied Learning and Vision

Mengye Ren

NYU

Spring 2025

embodied-learning-vision-course.github.io

Lecture Slides for Note Taking

Logistics

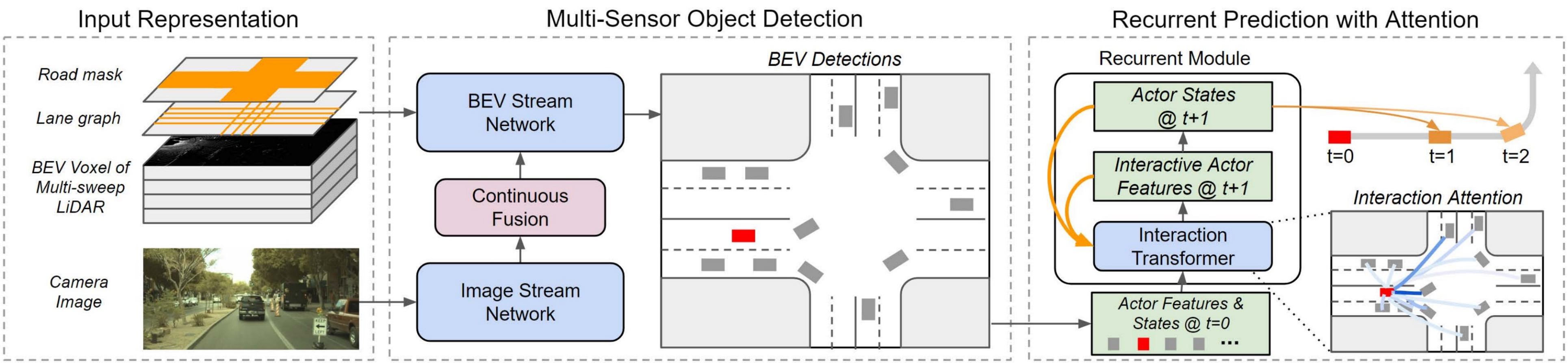
- Next week will be the last lecture.
- Student topic presentation begins.

	Component 1	Component 2
Week 6 (Feb 27)	End-to-End Planning, Continual Learning (1.5 hr)	Tutorial: LLM Agents
Week 7 (Mar 6)	Continual Learning, Few-shot Learning (1 hr)	**Deep Learning for Structured Prediction** Kangrui Yu, Tanishq Sardana, Qing Mu, Owais Shuja
Week 8 (Mar 13)	Guest Lecture – Prof. Wei-Chiu Ma (1 hr)	**3D Vision and Mapping** Sihang Li, Kanishkha Jaisankar, Denis Mbey Akola, Zijin Hu
Week 9 (Mar 20)	**SSL and Object Discovery** Anurup Naskar, Dahye Kim, Sal Yeung, Surbhi (1.5 hr)	**World Model 1** Sidhartha Reddy Potu, Andrew Deur
Week 10	Spring Break.	

NYU

Multi-Agents Joint Prediction

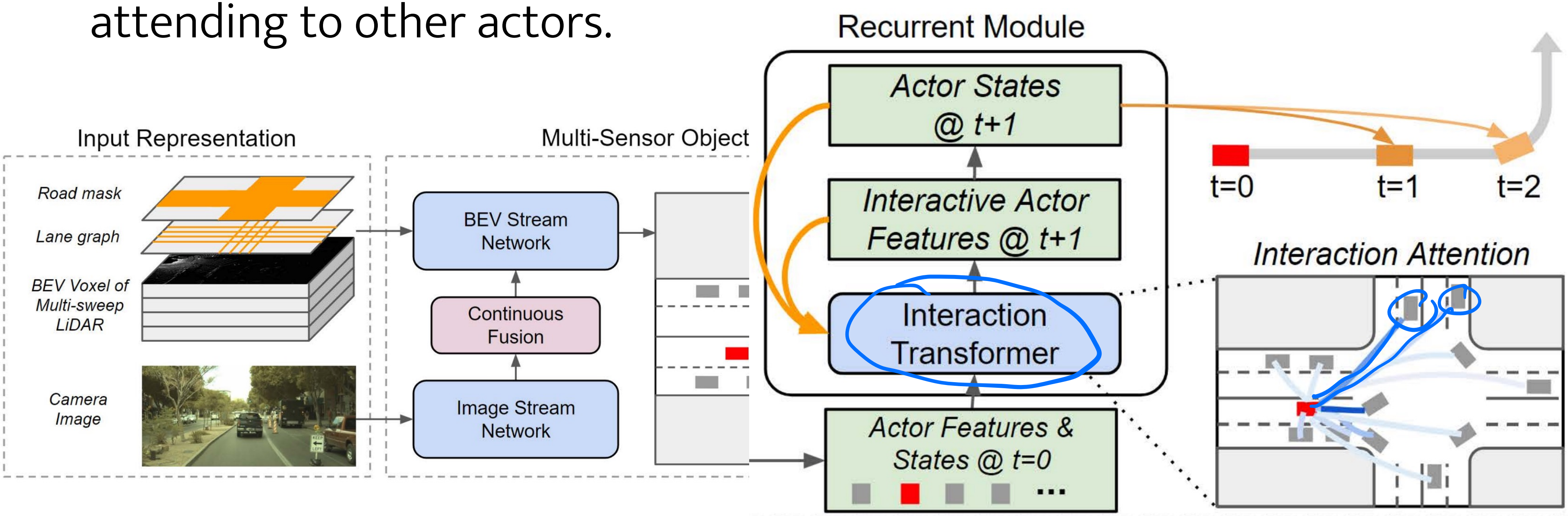
- Joint predict future trajectories by attending to other actors.



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

Multi-Agents Joint Prediction

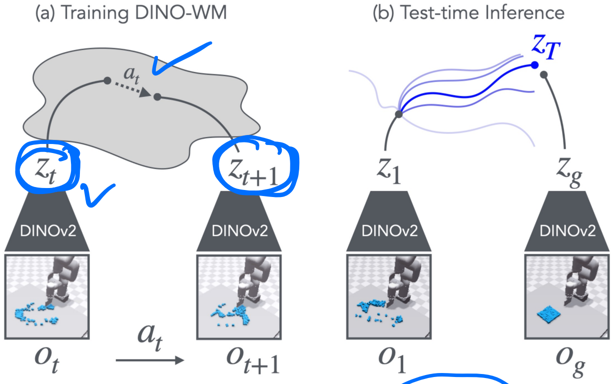
- Joint predict future trajectories by attending to other actors.

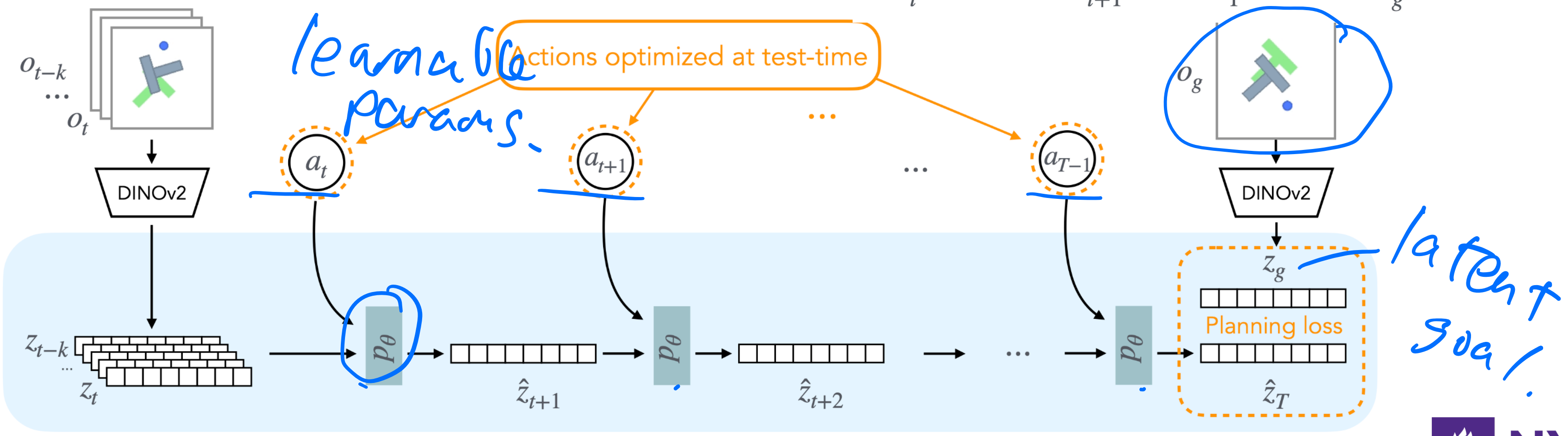


Li et al. End-to-end Contextual Perception and Prediction with Interaction Transformer. IROS 2020.

Latent Prediction + MPC

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





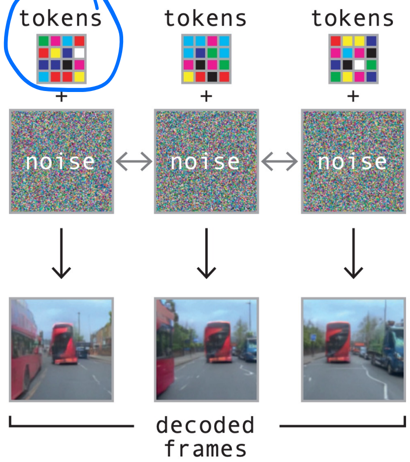
Zhou et al. *DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning.* 2024.

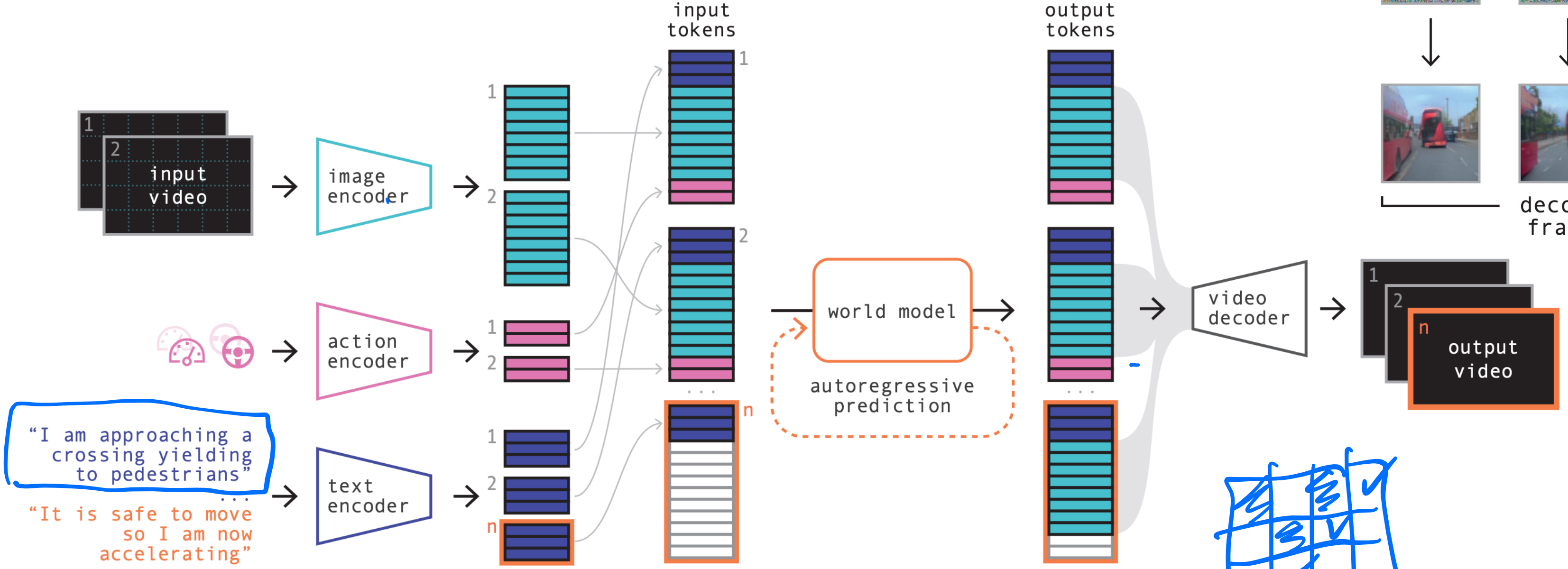
NYU

World Model in Video Prediction

Mask GIT.

- Text+action conditioned generation. Diffusion decoder.





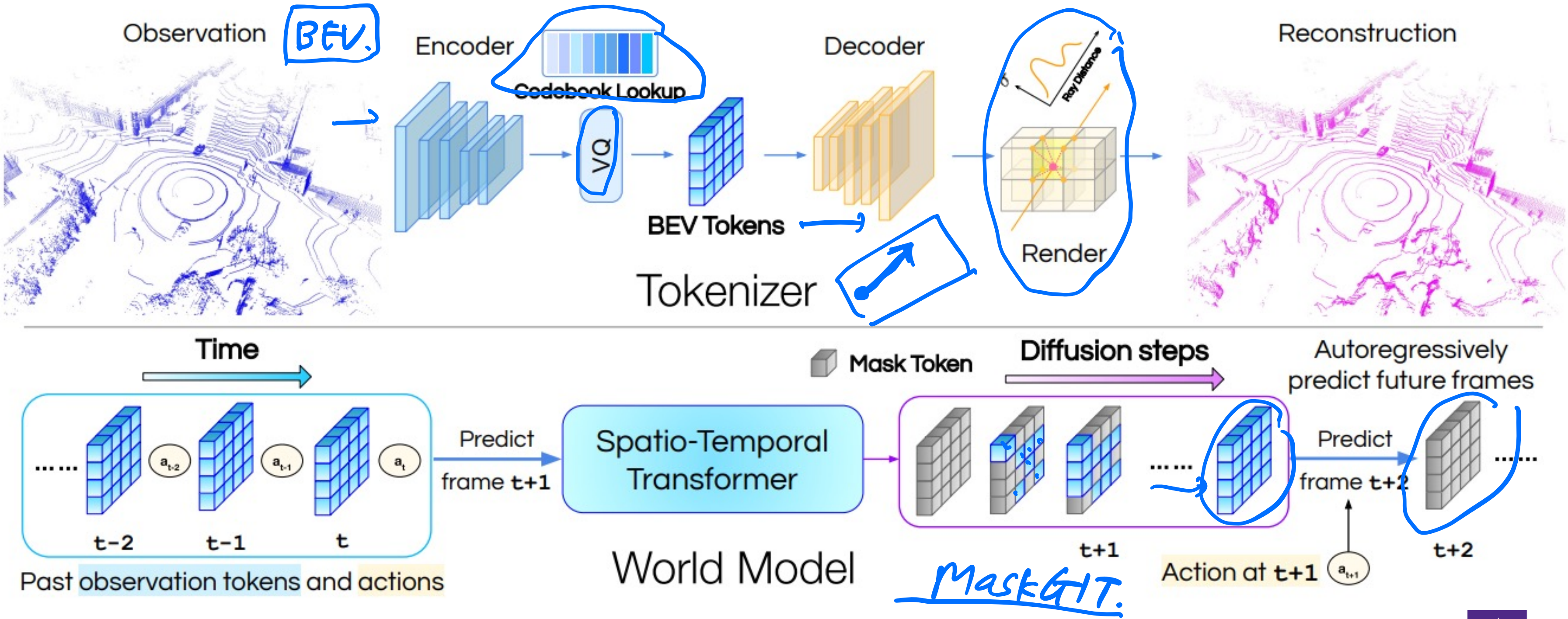
Hu et al. GAIA-1: A Generative World Model for Autonomous Driving. arXiv 2023.

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

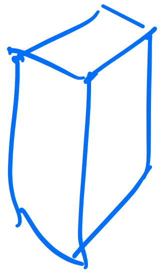
- Autoregressively predict future 3D point clouds.



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

Summary: World Models

Summary: World Models



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

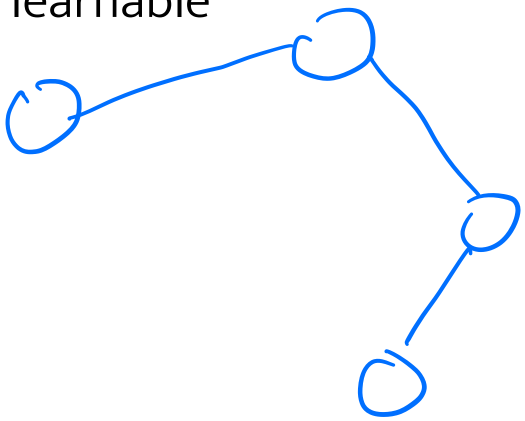
Summary: World Models

- 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



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

Imitation Learning

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

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

MSE

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

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

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

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

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

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

- 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

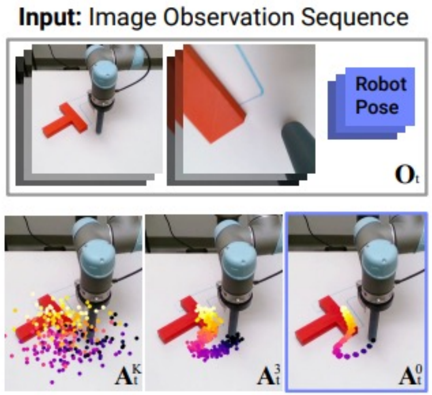
Direct Policy Learning from Diffusion

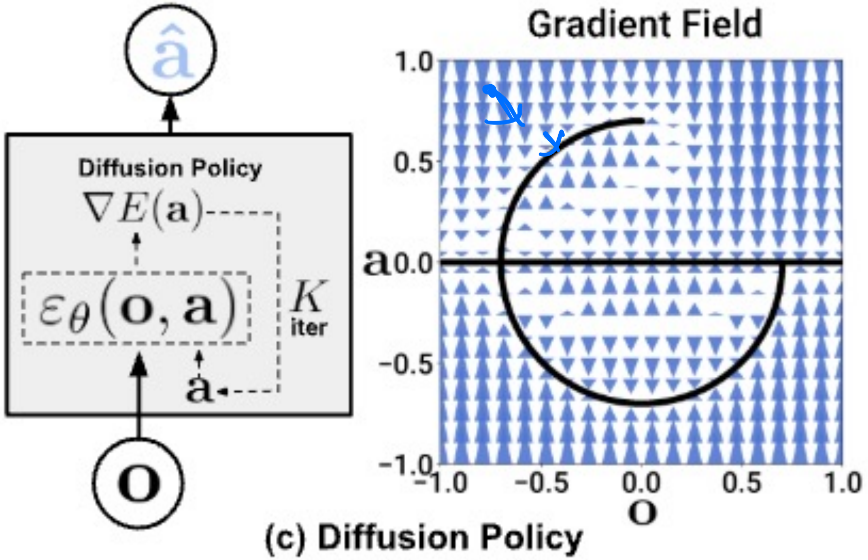
- Error prediction network is conditioned on observation features.

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

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

A_t^0



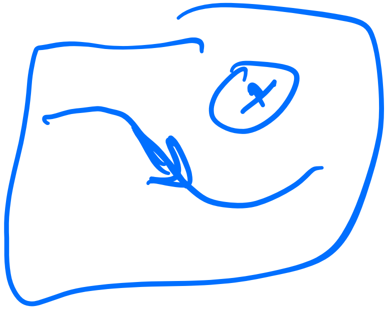


Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. RSS 2023

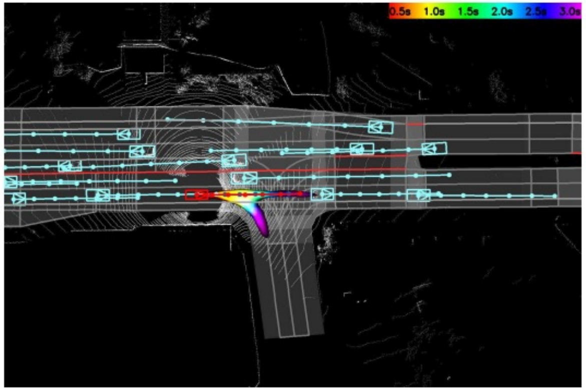
NYU

Cost/Value Volume Reasoning

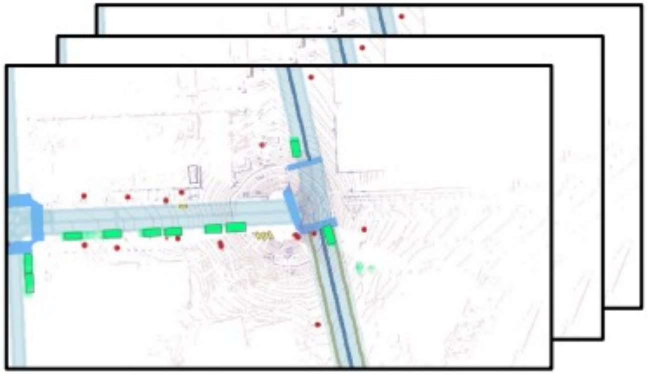
- Interpretability (both costs and planner inputs)



Rasterization



Semantic Occupancy



NYU

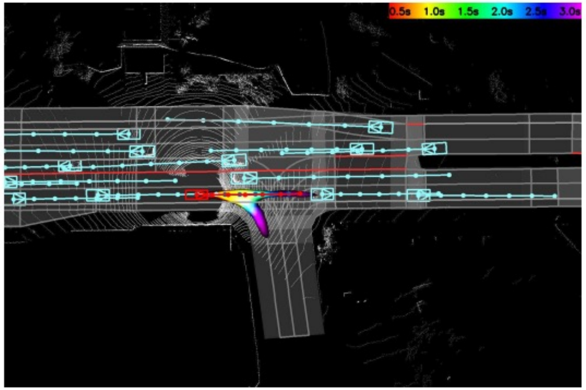
Cost/Value Volume Reasoning

$*\alpha$

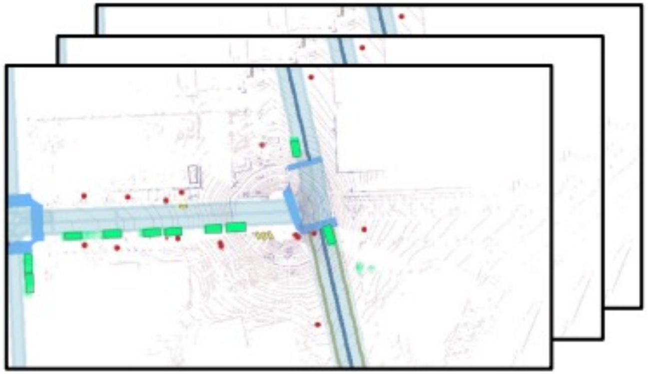
obstacles.

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

Rasterization

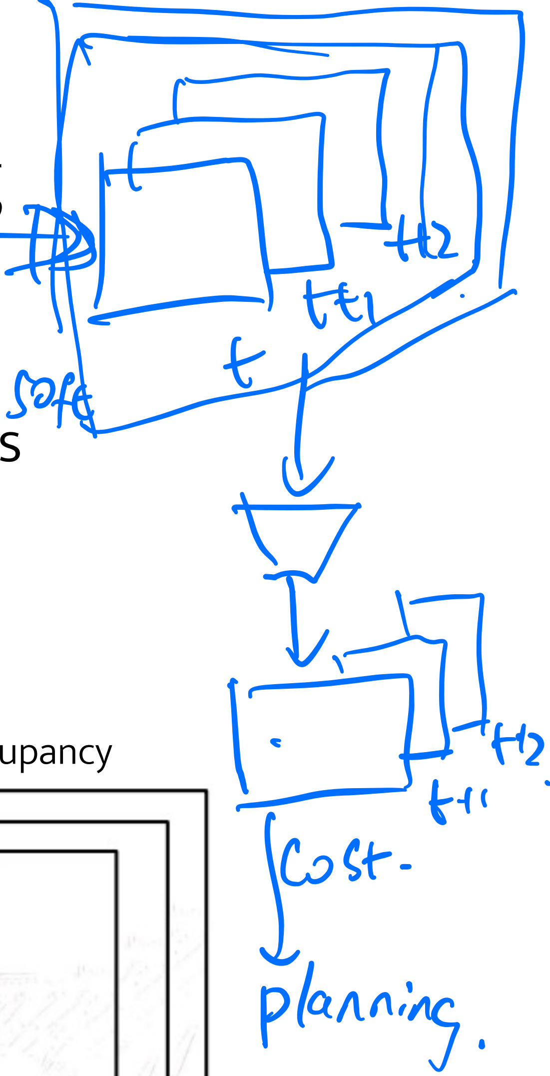


Semantic Occupancy

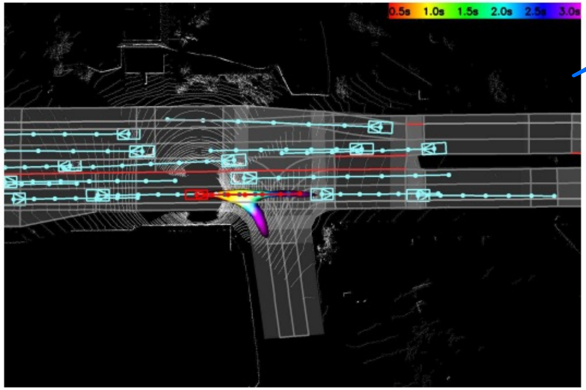


Cost/Value Volume Reasoning

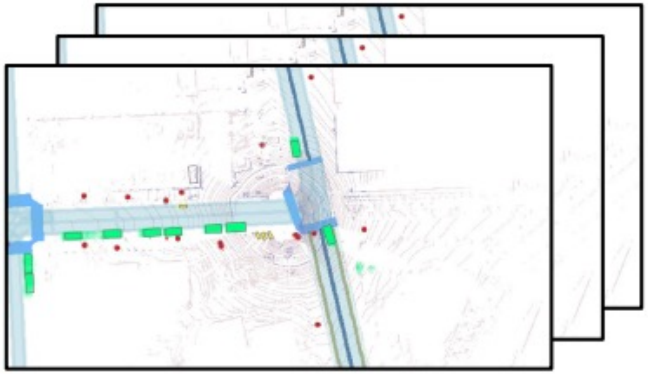
- Interpretability (both costs and planner inputs)
- Use spatial geometry to form cost from explicit objects
- Predict spatial cost volume
  - Rasterize the scene for spatial inputs
  - Predict soft occupancy volumes (present and future)



Rasterization



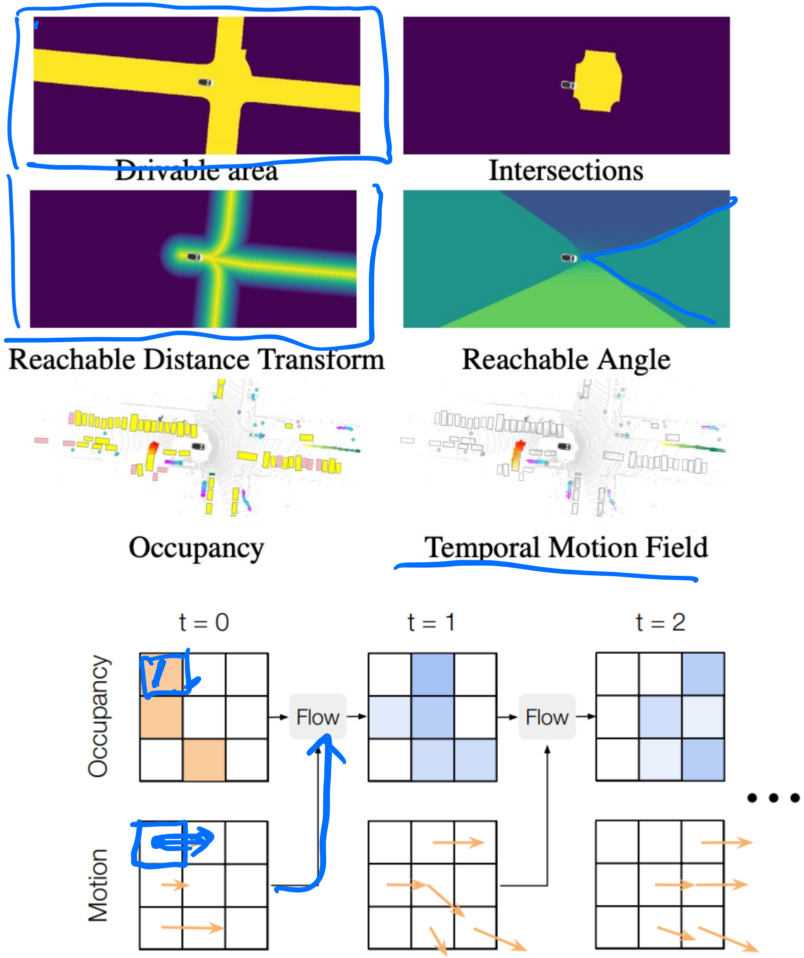
Semantic Occupancy

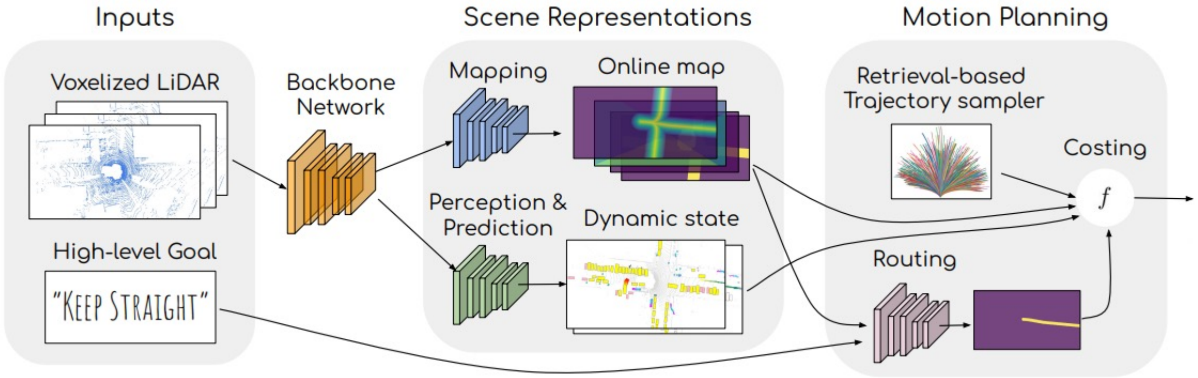


NYU

Learning Through Interpretable Predictions

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

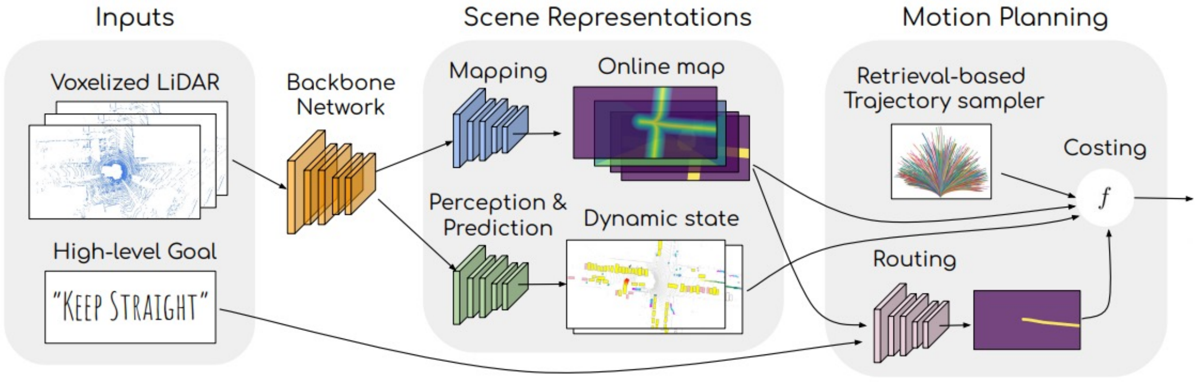


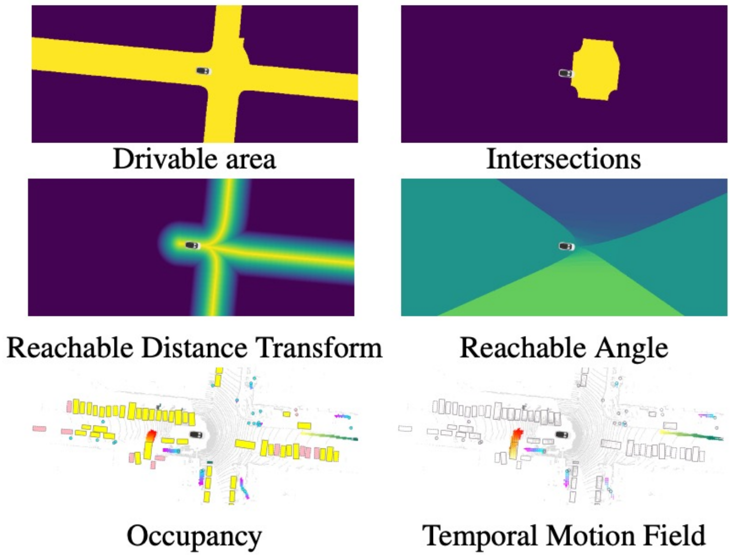


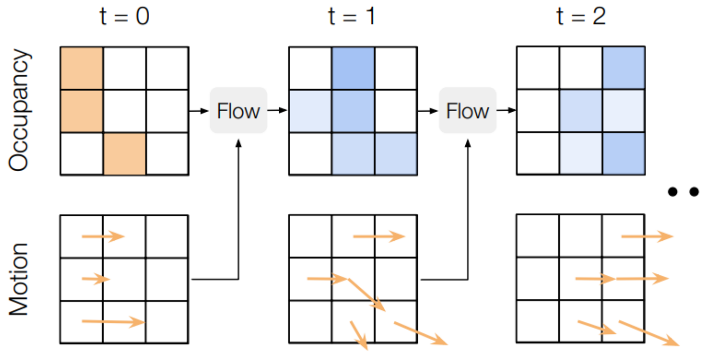
Casas et al. MP3: A Unified Model to Map, Perceive, Predict and Plan. CVPR 2021.

Learning Through Interpretable Predictions

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







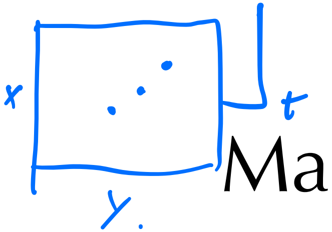
Casas et al. MP3: A Unified Model to Map, Perceive, Predict and Plan. CVPR 2021.

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.

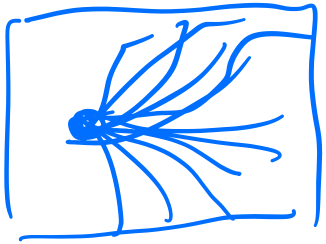
$$\operatorname*{argmin}_{\theta} \sum_{\{(\hat{x}_i^t, \hat{y}_i^t)\}_{i=1\ldots N}} \max_{i} \sum_{t=1}^{T} \underbrace{C_\theta^t[x_t, y_t]}_{\text{expert}} \underbrace{- C_\theta^t[\hat{x}_i^t, \hat{y}_i^t]}_{\text{model.}} + d_i^t$$

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.



✓ dynamics

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

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.

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)

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

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

NYU

EBM Planning

- Energy-based framework also needs negative samples.

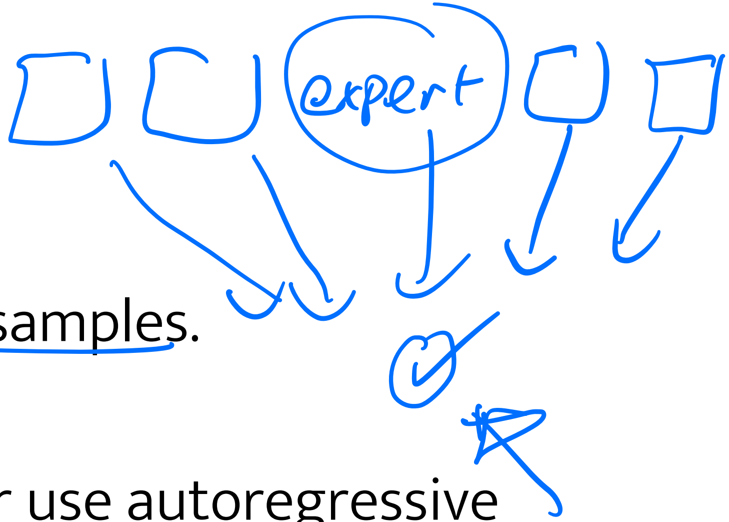
Florence et al. Implicit Behavioral Cloning. CoRL 2021.

EBM Planning

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

Florence et al. Implicit Behavioral Cloning. CoRL 2021.

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.

00100

Florence et al. Implicit Behavioral Cloning. CoRL 2021.

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.

EBM Planning

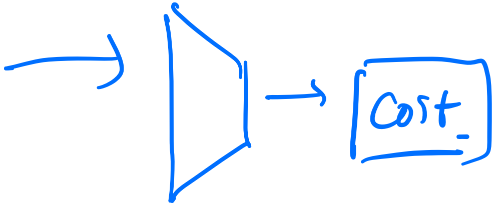
- Energy-based framework also needs negative samples.
- “Pick” the groundtruth sample among others.
- If there isn’t an external sampler, we can either use autoregressive energy of sampling one dimension at a time, or gradient-based Langevin MCMC.

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

Florence et al. Implicit Behavioral Cloning. CoRL 2021.

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.

Value Iteration Networks

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

NYU

Value Iteration Networks

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

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

bad policy.

NYU

Value Iteration Networks

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

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

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

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:

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

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

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

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

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.

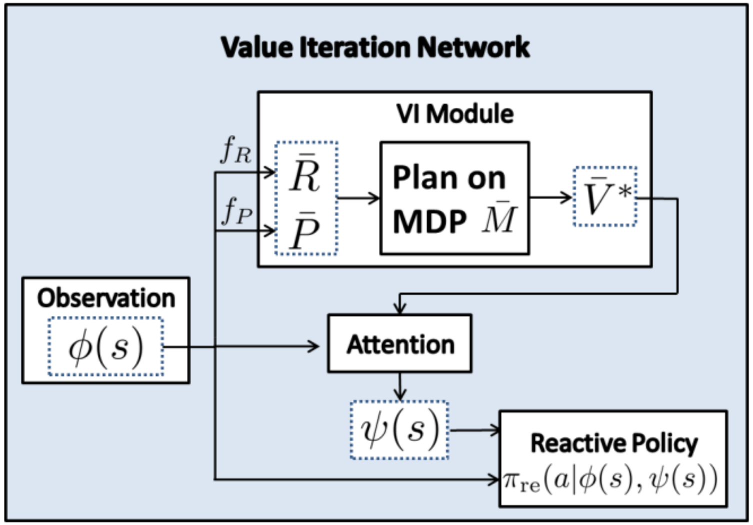
Value Iteration Networks

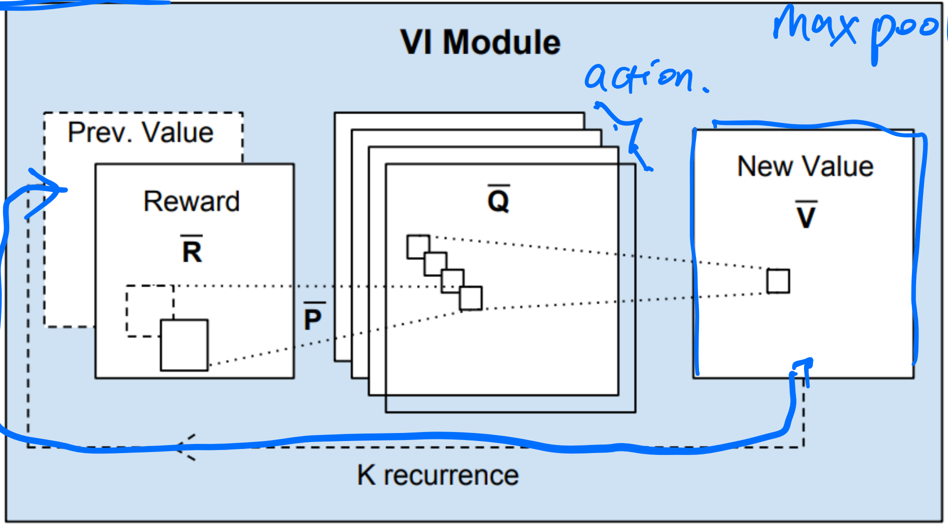
- Reward and previous value are fed into a CNN to generate Q of A channels. Transition matrix is convolutional kernel. Then Max-Pooling.

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

max pooling.

action.





Tamar et al. Value Iteration Networks. NeurIPS 2016.

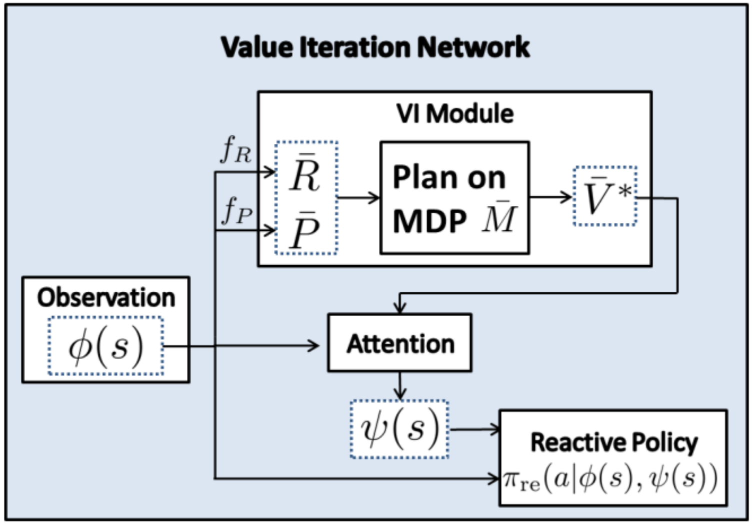
NYU

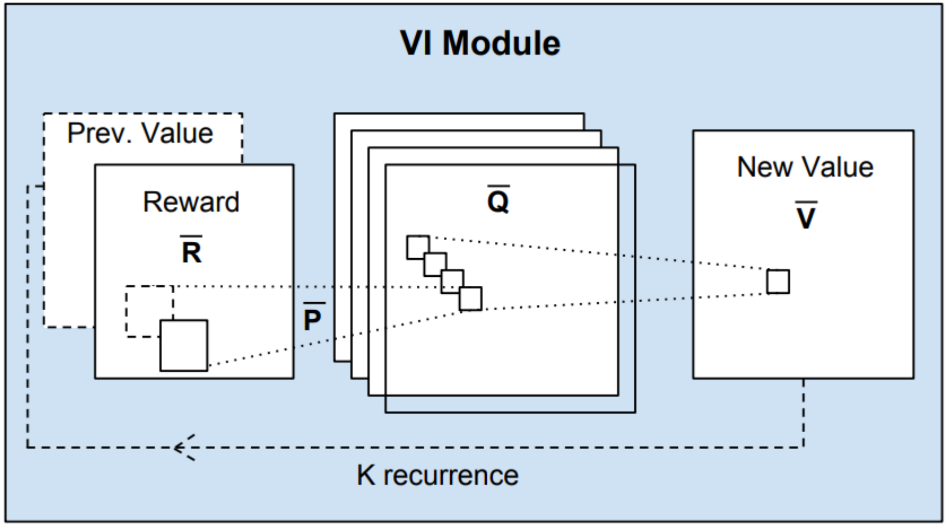
Value Iteration Networks

{ imitation
RL.

- 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.

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.

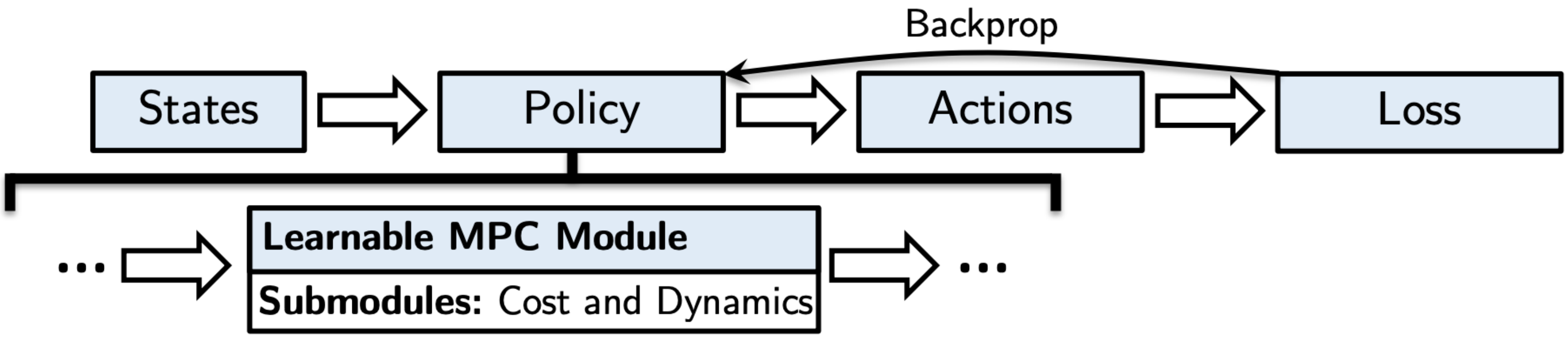
NYU

VIN → RNN.

$\underbrace{\text{k steps.}}$

Backprop through Planning

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

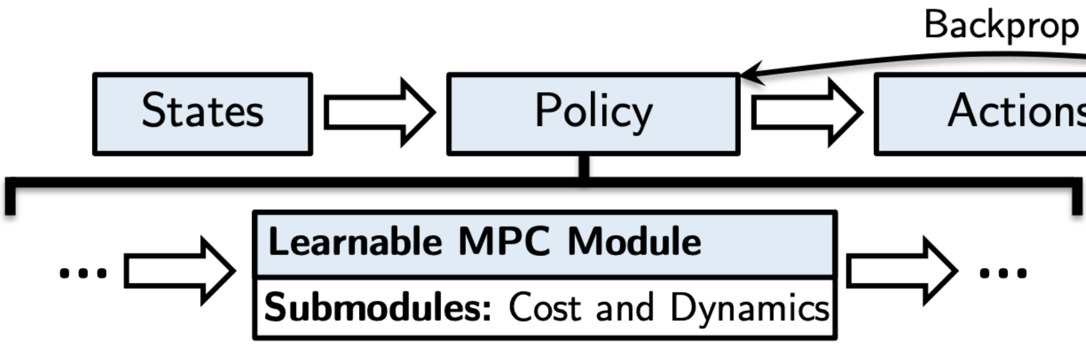


Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

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.

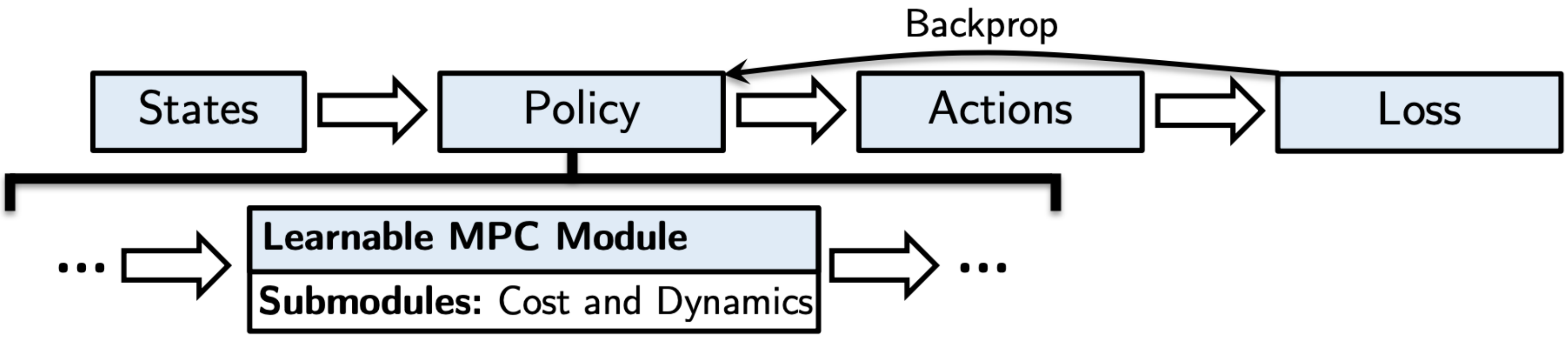
Backprop through Planning

- 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

K

a^*.

T-BPTT

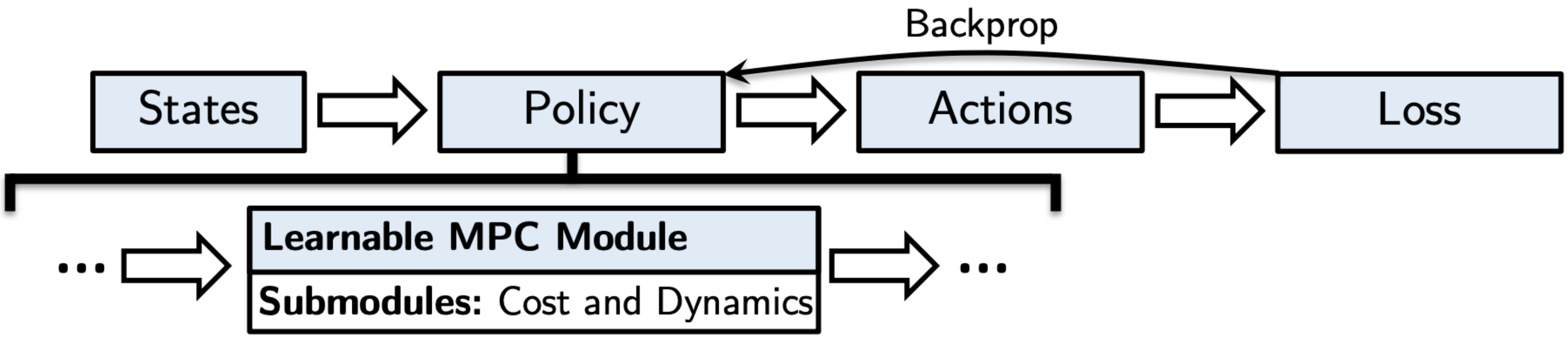


Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

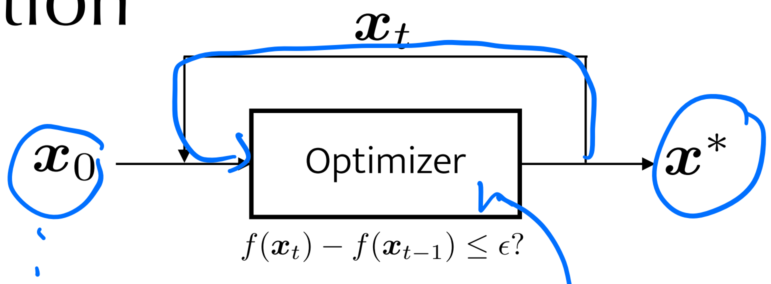
Backprop through Planning

- 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.

Implicit Differentiation



- Unconstrained case

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

Cost.

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

Implicit Differentiation

- Unconstrained case

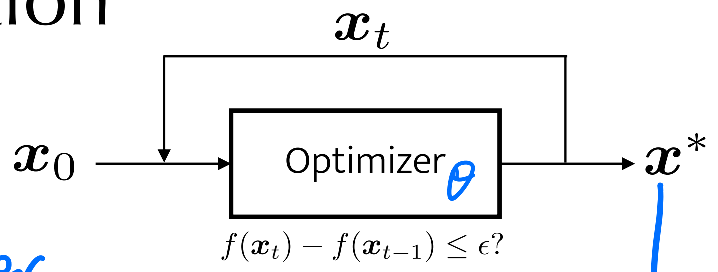
cost

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x};\boldsymbol{\theta}).$$

parameter.

$\frac{d}{d\theta}$

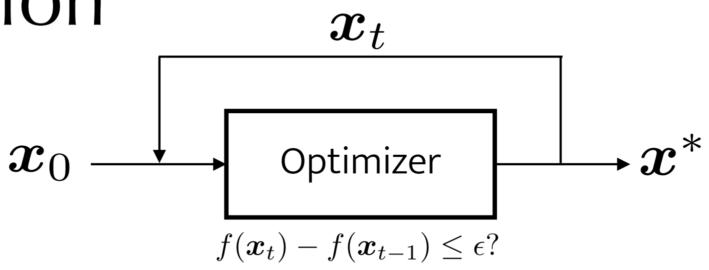
$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*;\boldsymbol{\theta})$$



Loss.

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

Implicit Differentiation



- Unconstrained case

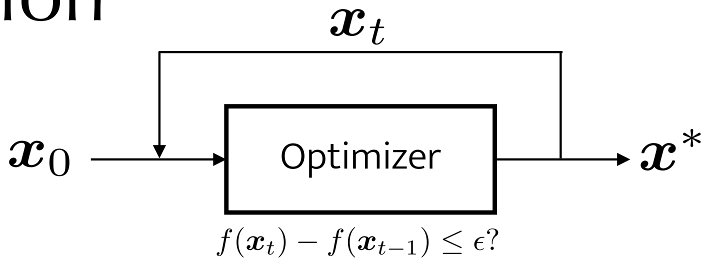
$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, f(\boldsymbol{x}; \boldsymbol{\theta}).$$

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

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

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

Implicit Differentiation



- Unconstrained case

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

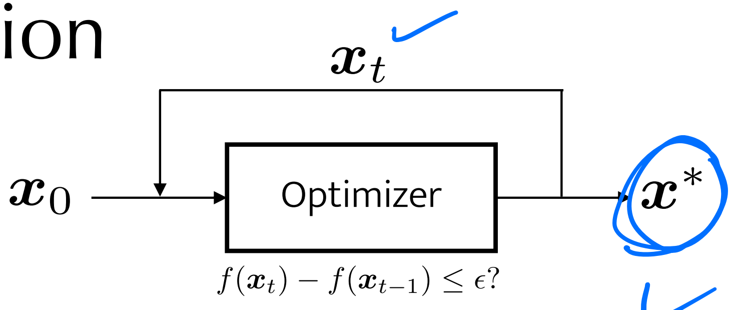
$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

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

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

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

Implicit Differentiation



- Unconstrained case

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

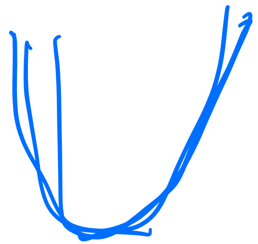
$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} J_{f,\boldsymbol{x}^*}(\boldsymbol{x}^*; \boldsymbol{\theta})$$

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

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

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

$\boxed{N \times N}$



↳ vector.

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.

Implicit Differentiation

- How to compute Hessian inverse vector product?
- Conjugate gradient, solve $\boxed{Ax = b}$

Solve x

$x = \boxed{A^{-1}b}$

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

$$\alpha_k := \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}$$

$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \boxed{\mathbf{A}\mathbf{p}_k}$
if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop

$$\beta_k := \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$$

$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
$k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result

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

NYU

Implicit Differentiation

Jv

- How to compute Hessian inverse vector product?
- Conjugate gradient, solve $Ax = b$
- Neumann series (finite truncation)

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

  - 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

$$\alpha_k := \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}$$

$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$

$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \boxed{\mathbf{A}\mathbf{p}_k}$ Hv

if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop

$$\beta_k := \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$$

$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$

$k := k + 1$

end repeat

return $\mathbf{x}_{k+1}$ as the result

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

Differentiable LQR

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

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

quadratic.

function ← θ.

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

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

linear

Differentiable LQR

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

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

$$\operatorname*{argmin}_{\tau_{1:T}} \sum_{t=1}^{T} \frac{1}{2} \tau_t^\top C_t \tau_t$$

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

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

Differentiable LQR

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

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

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

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

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

General QP:

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

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

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:

$$\operatorname*{argmin}_{\tau_{1:T}} \sum_{t=1}^{T} \frac{1}{2} \tau_t^\top C_t \tau_t$$

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

General QP:

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

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

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:

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

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

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

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

General QP:

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

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

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.

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.

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.

Differentiable LQR

- Apply differentiation

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

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

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

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

- Equivalent QP:

$$\left.\begin{aligned}\boldsymbol{d}^* = \underset{\boldsymbol{d}}{\operatorname{argmin}}\ & \frac{1}{2}\boldsymbol{d}^\top Q\boldsymbol{d} + \frac{\partial \ell}{\partial \boldsymbol{x}^*}^\top \boldsymbol{d},\\ \text{subject to}\ & A\boldsymbol{d} = \boldsymbol{0}.\end{aligned}\right\} \to \underline{LQR}$$

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

NYU

Differentiable LQR

- Apply differentiation

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

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

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

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

- Equivalent QP:

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

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

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

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.

Differentiable MPC

- What about general MPC?

$$\underset{x_{1:T}\in\mathcal{X},u_{1:T}\in\mathcal{U}}{\operatorname{argmin}} \sum_{t=1}^{T} \boxed{C_t(x_t, u_t)}$$
$$\text{subject to } x_{t+1} = f(x_t, u_t), x_1 = x_{\text{init}}.$$

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

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.

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.

$$\tilde{C}_{\theta,t}^{i} = C_{\theta,t}(\tau_t^i) + {p_t^i}^\top(\tau_t - \tau_t^i) + \frac{1}{2}(\tau_t - \tau_t^i)^\top H_t^i(\tau_t - \tau_t^i).$$

- Fixed point iteration.

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

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

NYU

Differentiable MPC

- What about general MPC?

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

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

- Use Taylor expansion to approximate.

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

- Fixed point iteration.

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

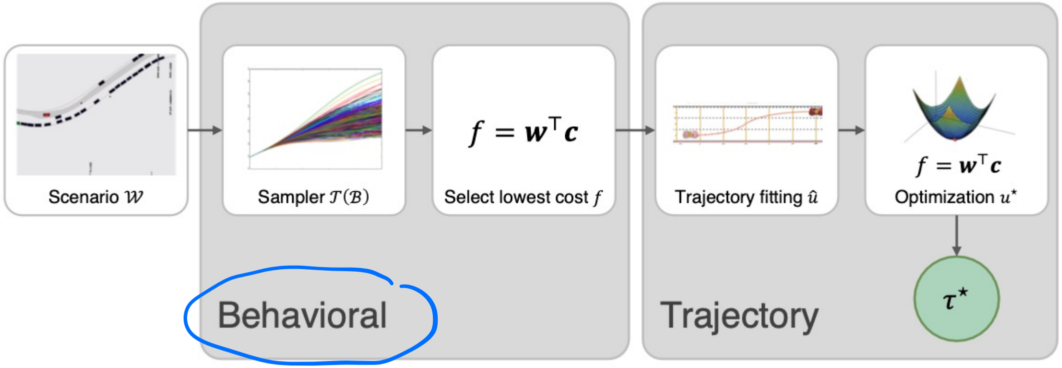
- Backward only depends on final quadratic approximation.

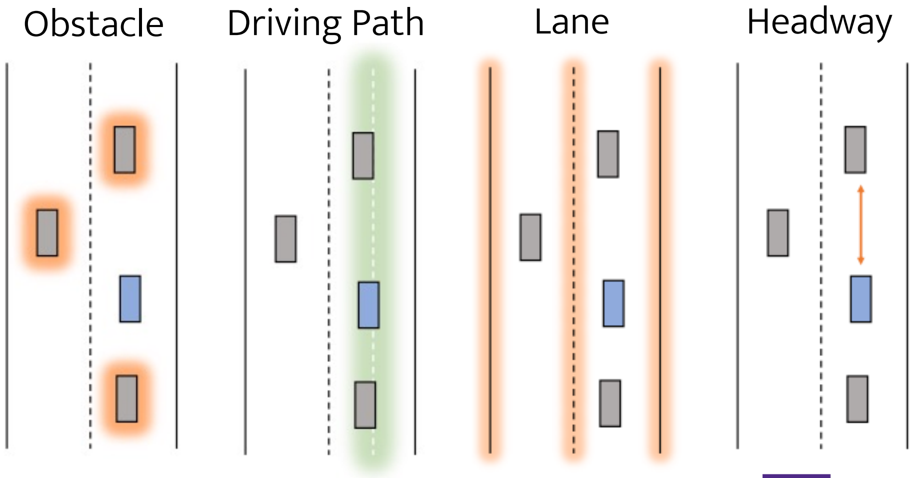
Amos et al. Differentiable MPC for End-to-end Planning and Control. NeurIPS 2018.

NYU

Behavioral vs. Trajectory Planning

- Gradient-based optimization provides a locally optimized trajectory.

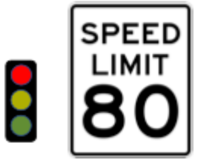


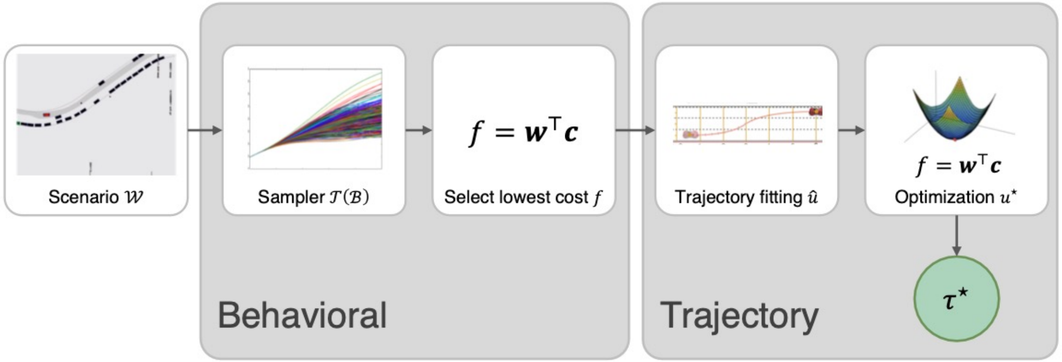


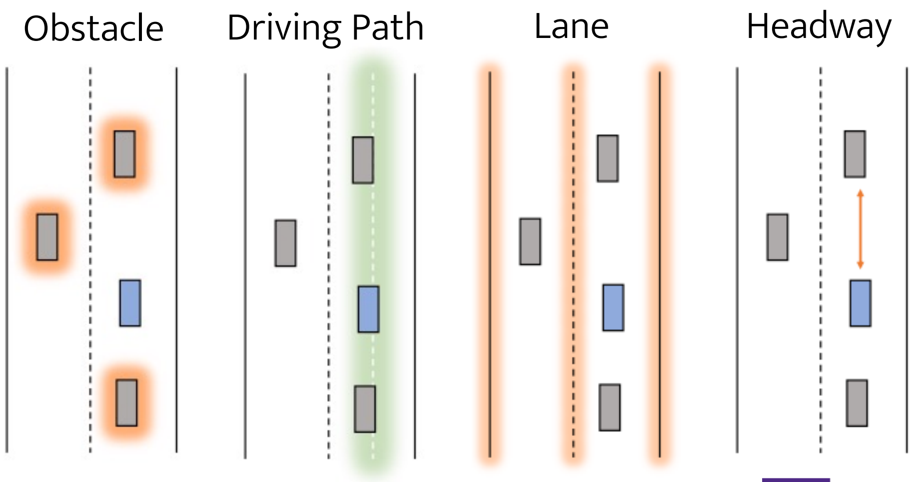
Sadat et al. Jointly Learnable Behavior and Trajectory Planning for Self-Driving Vehicles. IROS 2019.

Behavioral vs. Trajectory Planning

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



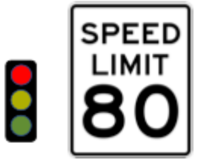




Sadat et al. Jointly Learnable Behavior and Trajectory Planning for Self-Driving Vehicles. IROS 2019.

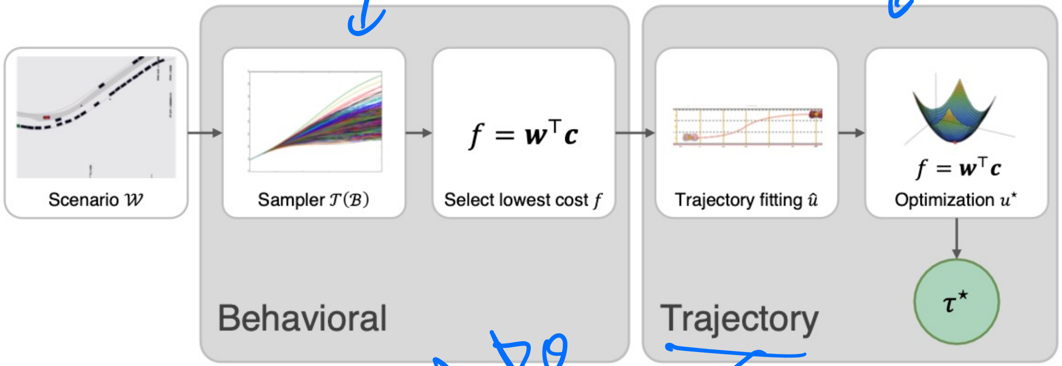
Behavioral vs. Trajectory Planning

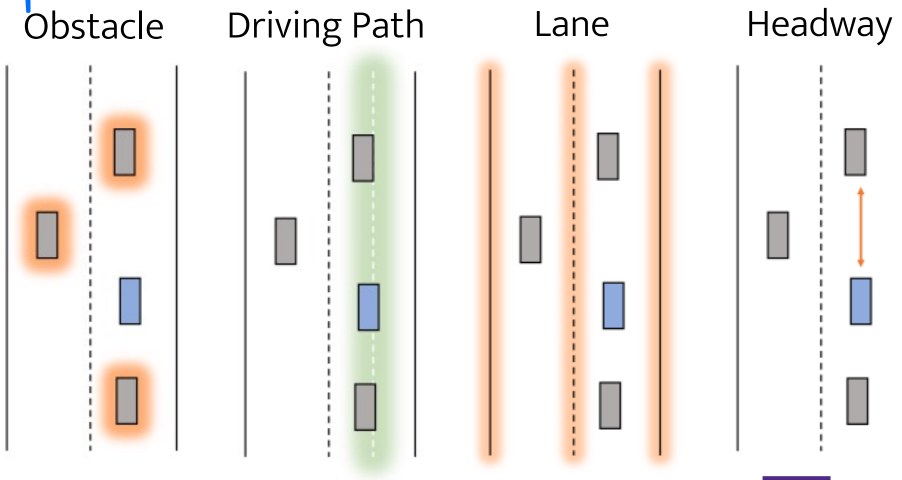
- Gradient-based optimization provides a locally optimized trajectory.
- Samples may be needed for reasoning global structure.
- Can learn together using the same learned costs.



max-margin

backprop.





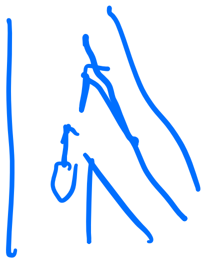
$\nabla\theta$

$\nabla\theta$

c

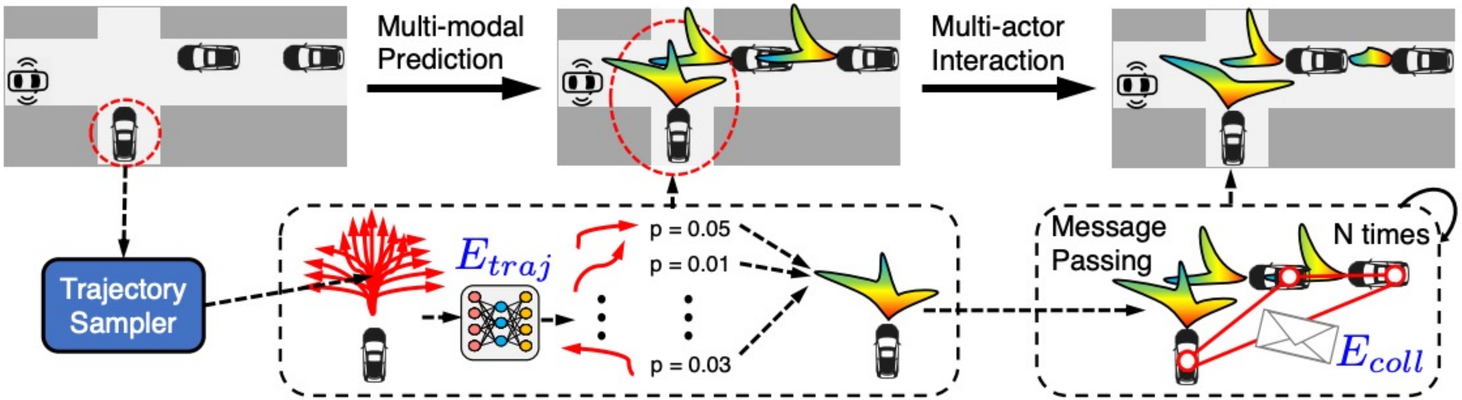
Sadat et al. Jointly Learnable Behavior and Trajectory Planning for Self-Driving Vehicles. IROS 2019.

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

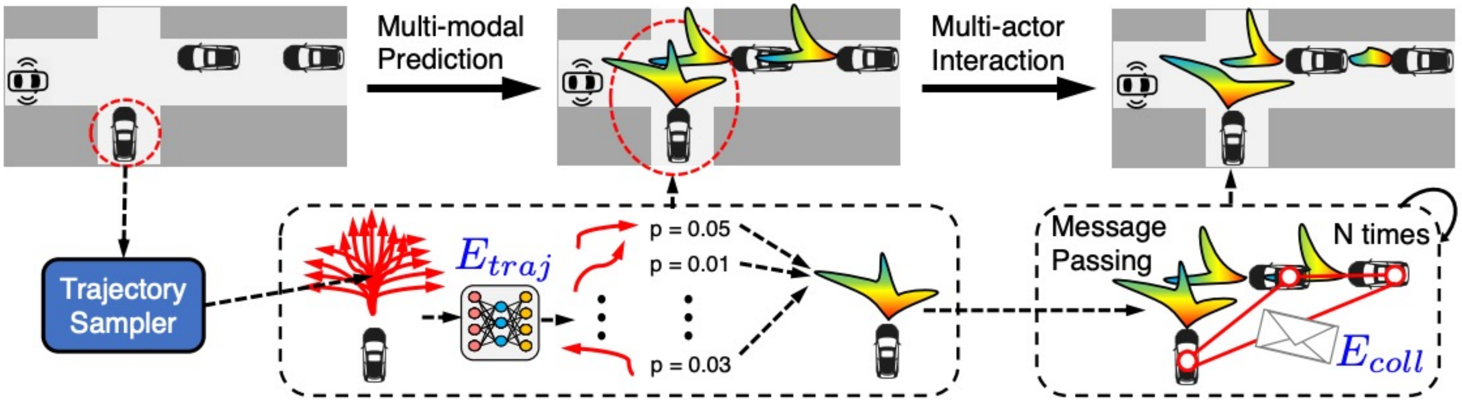
Planning with Social Reasoning

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

$$p(\mathbf{s}_1, \ldots, \mathbf{s}_N \mid \mathbf{X}) = \frac{1}{Z} \exp(-E_\theta(\mathbf{s_1}, \ldots, \mathbf{s}_N) \mid \mathbf{X})$$

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

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



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

NYU

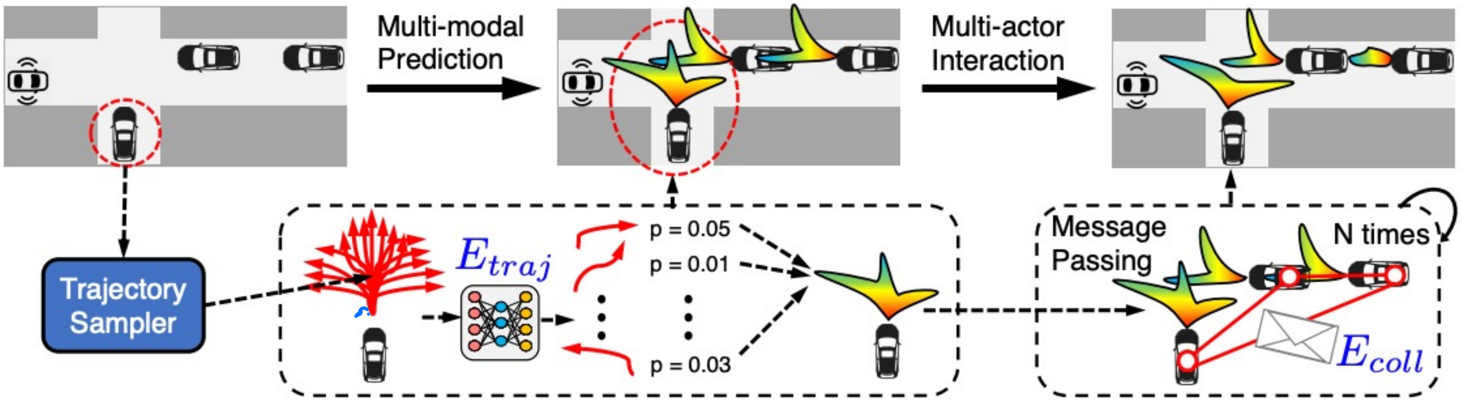
Planning with Social Reasoning

- 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.

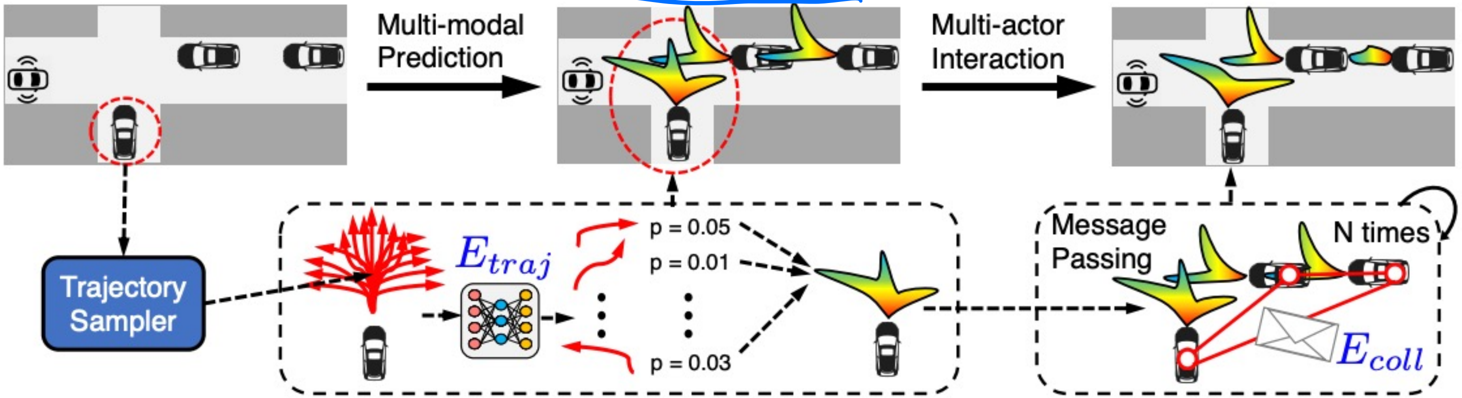
Planning with Social Reasoning

- 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.

Summary: End-to-End Planning

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

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

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

Module 5:
Continual Learning,
Few-Shot Learning, Meta-Learning

Why Continual Learning?

- The world is not a dataset that allows you to get IID samples.

Why Continual Learning?

- The world is not a dataset that allows you to get IID samples.
- The world keeps changing and evolving.

Why Continual Learning?

- The world is not a dataset that allows you to get IID samples.
- The world keeps changing and evolving.
- Online vs. Continual
  - Online means that samples arrive in a streaming / temporal partial order, but they may still come from a static distribution.

$$\theta_t = f(x_t, \theta_{t-1}) \quad x_{1:T} \sim \mathcal{X}$$

  - Example: Online reinforcement learning, trajectory roll out is online, but the environment is the same.
  - Continual learning means that there will be distribution shift.

NYU

What is Continual Learning?

- Distribution shift: Forgetting
  - Learning on A and then B, results in worse performance on A.

backward transfer.

NYU

What is Continual Learning?

- Distribution shift: Forgetting
  - Learning on A and then B, results in worse performance on A.
- Multi-task learning: Forward transfer
  - Learning Task A + B results in better learning in Task C compared to learning C alone.
  - Leverage the similarity between tasks.

What is Continual Learning?

- Distribution shift: Forgetting
  - Learning on A and then B, results in worse performance on A.
- Multi-task learning: Forward transfer
  - Learning Task A + B results in better learning in Task C compared to learning C alone.
  - Leverage the similarity between tasks.
- Compositionality
  - Learning A and B first, and then learning tasks with composed A+B.

NYU

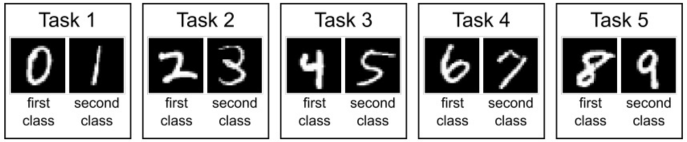
What is Continual Learning?

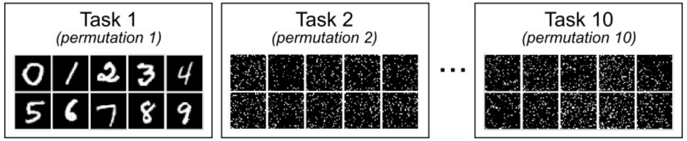
- Distribution shift: Forgetting
  - Learning on A and then B, results in worse performance on A.
- Multi-task learning: Forward transfer
  - Learning Task A + B results in better learning in Task C compared to learning C alone.
  - Leverage the similarity between tasks.
- Compositionality
  - Learning A and B first, and then learning tasks with composed A+B.
- Incremental/curriculum Learning
  - Learning A->B->C is easier than at random order.

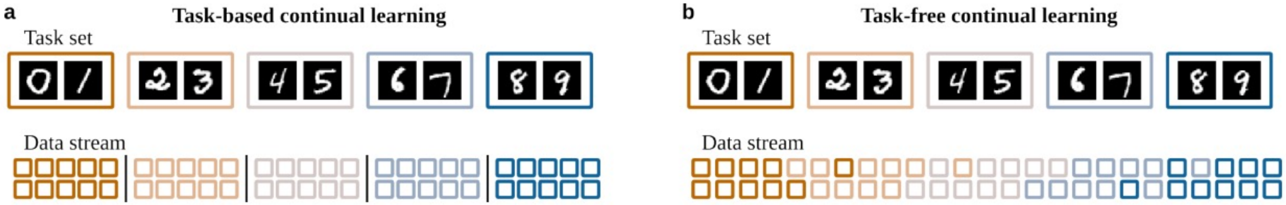
NYU

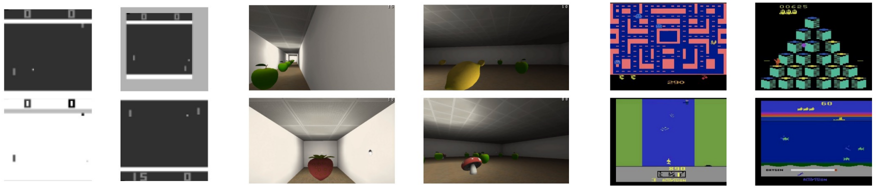
Continual Learning

- Learning a sequence of tasks without looking back.

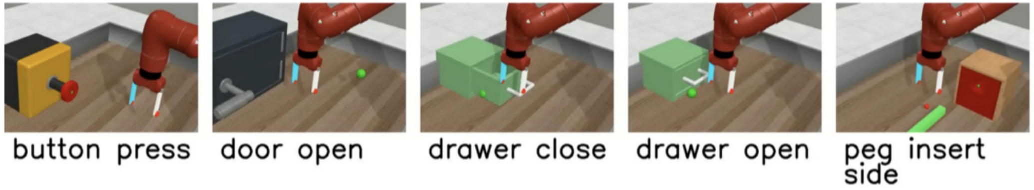








(a) Pong variants
(b) Labyrinth games
(c) Atari games

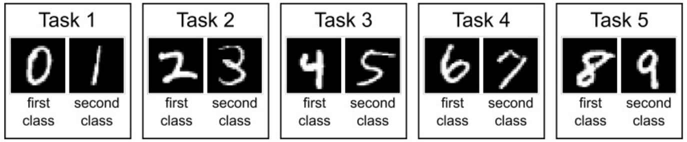


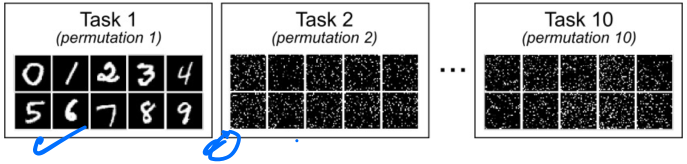
van de Ven et al. Continual learning and catastrophic forgetting. arXiv 2024.
van de Ven & Tolias. Three scenarios for continual learning. arXiv 2019.

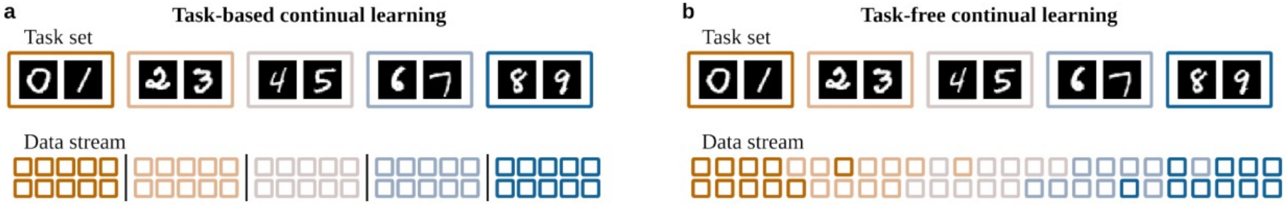
Continual Learning

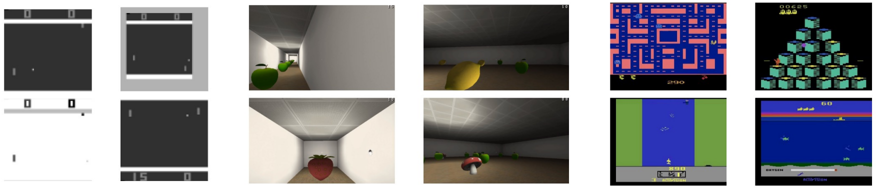
increment

- Learning a sequence of tasks without looking back.
- Goal is to do well on all of the tasks at the end.

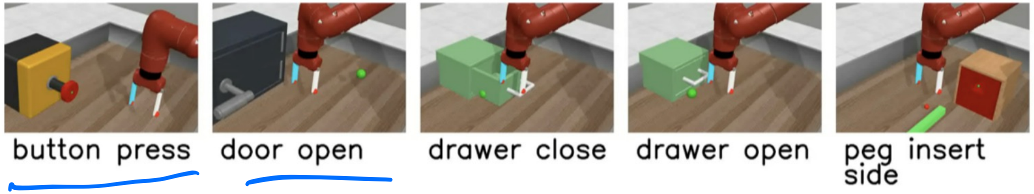








(a) Pong variants (b) Labyrinth games (c) Atari games

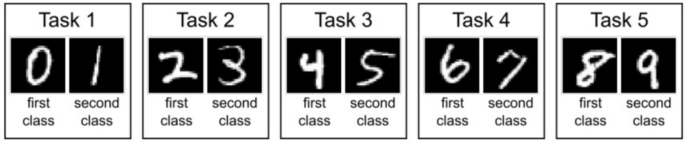


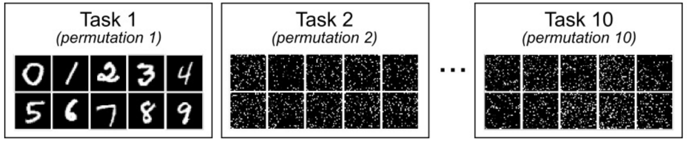
van de Ven et al. Continual learning and catastrophic forgetting. arXiv 2024.
van de Ven & Tolias. Three scenarios for continual learning. arXiv 2019.

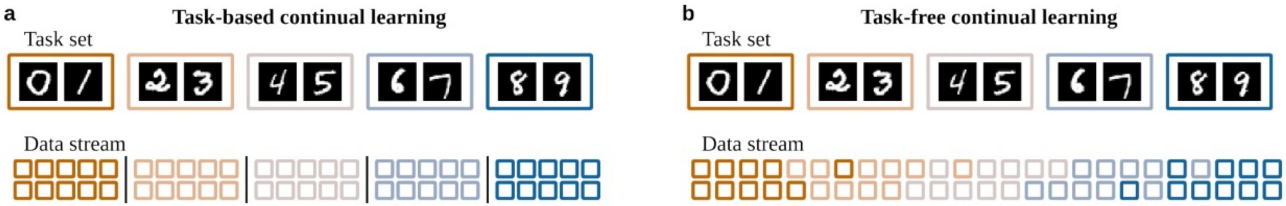
NYU

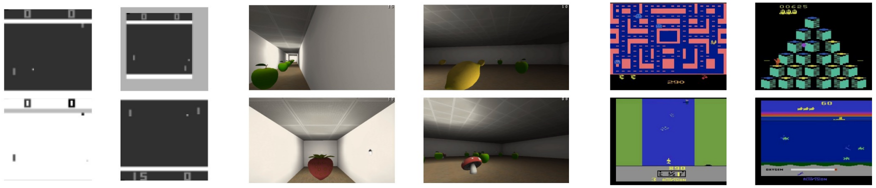
Continual Learning

- Learning a sequence of tasks without looking back.
- Goal is to do well on all of the tasks at the end.
- Task boundary

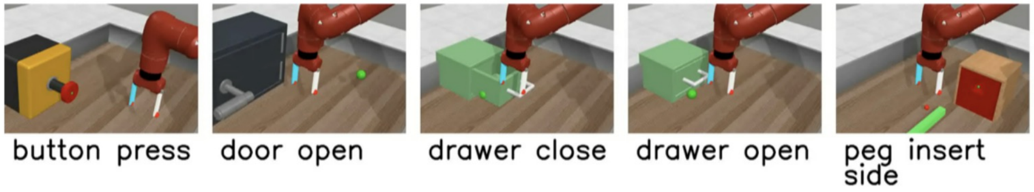








(a) Pong variants (b) Labyrinth games (c) Atari games

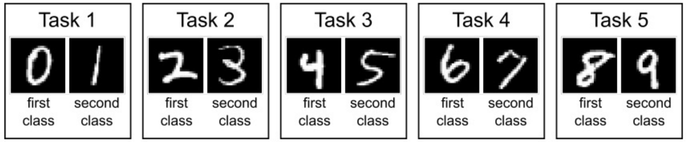


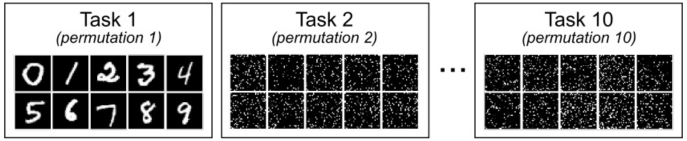
van de Ven et al. Continual learning and catastrophic forgetting. arXiv 2024.
van de Ven & Tolias. Three scenarios for continual learning. arXiv 2019.

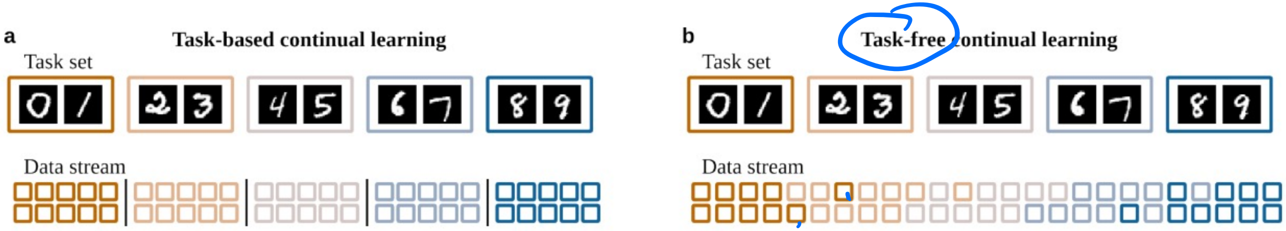
NYU

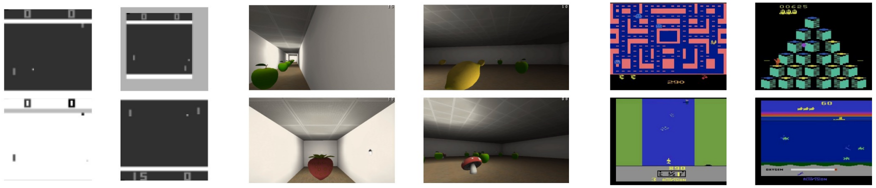
Continual Learning

- Learning a sequence of tasks without looking back.
- Goal is to do well on all of the tasks at the end.
- Task boundary
- Memory constraints

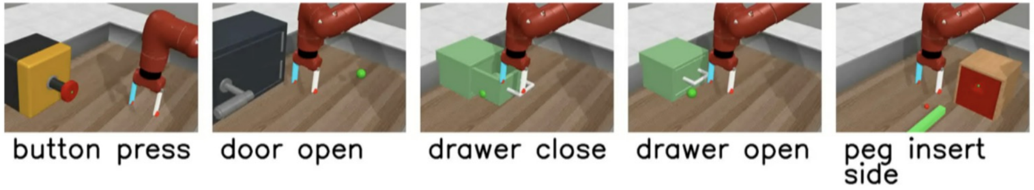








(a) Pong variants (b) Labyrinth games (c) Atari games



van de Ven et al. Continual learning and catastrophic forgetting. arXiv 2024.
van de Ven & Tolias. Three scenarios for continual learning. arXiv 2019.

Parameter Regularization

- Over-completeness Assumption. A multitude of models can reach equivalent performance.

$$\mathcal{S}_A = \{\theta \mid \ell_A(\theta) < \epsilon\}$$

$$\mathcal{S}_A \cap \mathcal{S}_B \neq \emptyset$$

Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. PNAS 2017.

Parameter Regularization

- Over-completeness Assumption. A multitude of models can reach equivalent performance.
- What is left is to efficiently find the intersection between A and B.

$$p(\theta \mid \mathcal{D}_A) = \mathcal{N}(\theta; \theta^*, \Sigma)$$

$$\mathcal{S}_A = \{\theta \mid \ell_A(\theta) < \epsilon\}$$

$$\mathcal{S}_A \cap \mathcal{S}_B \neq \emptyset$$

Parameter Regularization

- Over-completeness Assumption. A multitude of models can reach equivalent performance.

$$\mathcal{S}_A = \{\theta \mid \ell_A(\theta) < \epsilon\}$$

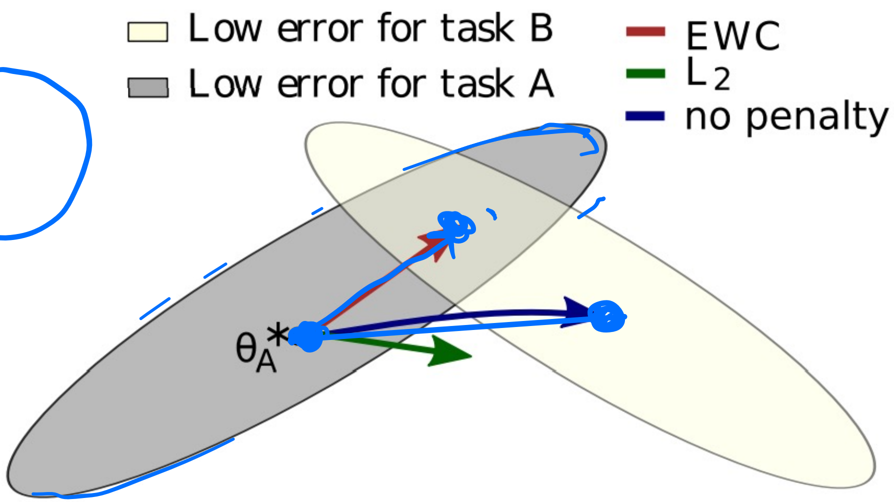
$$\mathcal{S}_A \cap \mathcal{S}_B \neq \emptyset$$

- What is left is to efficiently find the intersection between A and B.

$$p(\theta \mid \mathcal{D}_A) = \mathcal{N}(\theta; \theta^*, \Sigma)$$

- Elastic Weight Consolidation (EWC):

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta^*_{A,i})^2$$

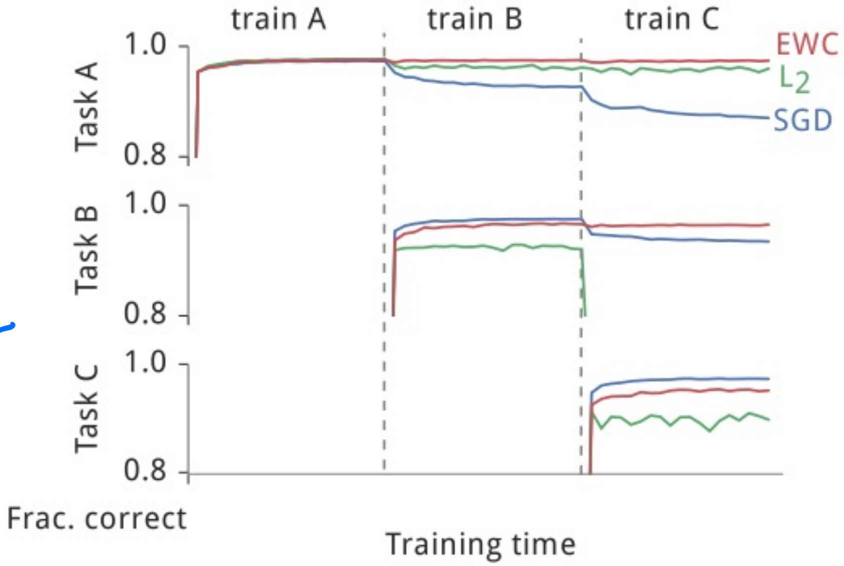


Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. PNAS 2017.

Computing Fisher

- At the end of each epoch, compute the gradient squared:

$$F_i = \left(\frac{d\mathcal{L}}{d\theta_i}\right)^2$$



Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. PNAS 2017.
Zenke et al. Continual Learning Through Synaptic Intelligence. ICML 2017.

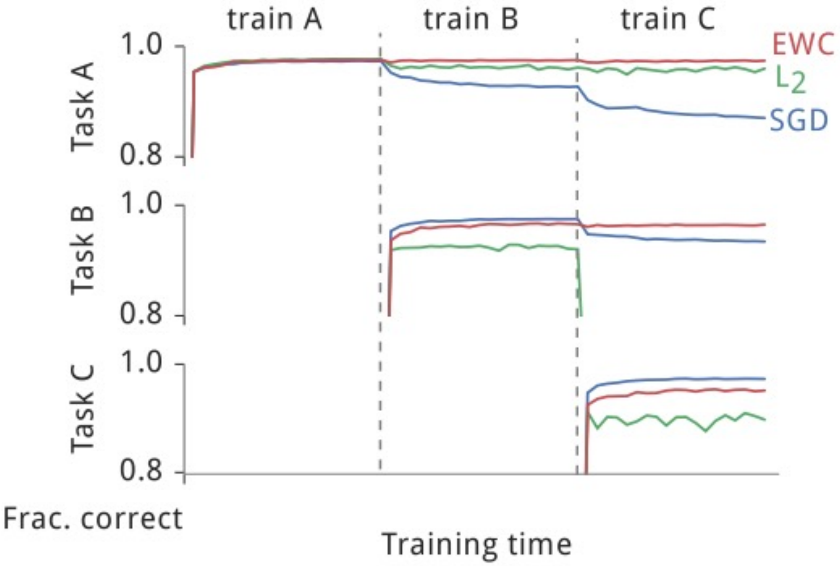
NYU

Computing Fisher

- At the end of each epoch, compute the gradient squared:

$$\boxed{F_i = \left(\frac{d\mathcal{L}}{d\theta_i}\right)^2}$$

- Measures the sensitivity on each parameter dimension.



Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. PNAS 2017.
Zenke et al. Continual Learning Through Synaptic Intelligence. ICML 2017.

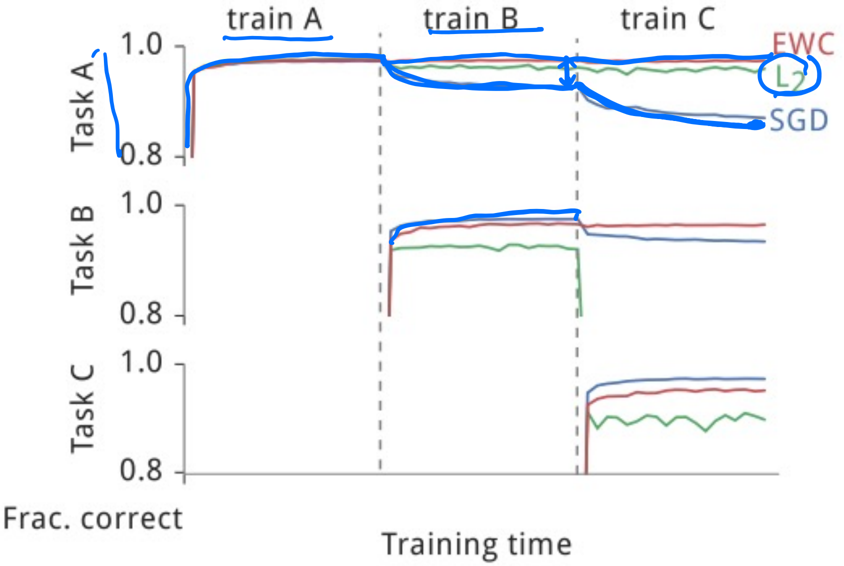
Computing Fisher

plasticity --
stability. ++

- At the end of each epoch, compute the gradient squared:

$$F_i = \left(\frac{d\mathcal{L}}{d\theta_i}\right)^2$$

- Measures the sensitivity on each parameter dimension.
- You can also accumulate an online estimate.



Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. PNAS 2017.
Zenke et al. Continual Learning Through Synaptic Intelligence. ICML 2017.

Variational Continual Learning (VCL)

- Bayesian formulation:

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathcal{D}_t \mid \boldsymbol{\theta}) \propto p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T-1}) p(\mathcal{D}_T \mid \boldsymbol{\theta}).$$

Variational Continual Learning (VCL)

- Bayesian formulation:

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathcal{D}_t \mid \boldsymbol{\theta}) \propto p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T-1}) p(\mathcal{D}_T \mid \boldsymbol{\theta}).$$

- Variational approach:

$$q_t(\boldsymbol{\theta}) = \operatorname*{argmin}_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\boldsymbol{\theta}) \parallel \frac{1}{Z_t} q_{t-1}(\boldsymbol{\theta}) p(\mathcal{D}_t \mid \boldsymbol{\theta}) \right).$$

Nguyen et al. Variational Continual Learning. ICLR 2018.

Variational Continual Learning (VCL)

- Bayesian formulation:

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathcal{D}_t \mid \boldsymbol{\theta}) \propto p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T-1}) p(\mathcal{D}_T \mid \boldsymbol{\theta}).$$

- Variational approach:

$$q_t(\boldsymbol{\theta}) = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \, \mathrm{KL}\left(q(\boldsymbol{\theta}) \parallel \frac{1}{Z_t} q_{t-1}(\boldsymbol{\theta}) p(\mathcal{D}_t \mid \boldsymbol{\theta}) \right).$$

- Loss: $\mathcal{L}(q_t(\boldsymbol{\theta})) = \mathbb{E}_{\boldsymbol{\theta} \sim q_t(\boldsymbol{\theta})}[-\log p(\mathbf{y}|\mathbf{x}, \theta)] + \mathrm{KL}(q_t(\boldsymbol{\theta} \parallel q_{t-1}(\boldsymbol{\theta})).$

Nguyen et al. Variational Continual Learning. ICLR 2018.

Variational Continual Learning (VCL)

- Bayesian formulation:

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathcal{D}_t \mid \boldsymbol{\theta}) \propto p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T-1}) p(\mathcal{D}_T \mid \boldsymbol{\theta}).$$

- Variational approach:

$$q_t(\boldsymbol{\theta}) = \operatorname*{argmin}_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\boldsymbol{\theta}) \parallel \frac{1}{Z_t} q_{t-1}(\boldsymbol{\theta}) p(\mathcal{D}_t \mid \boldsymbol{\theta}) \right).$$

- Loss:

$$\mathcal{L}(q_t(\boldsymbol{\theta})) = \mathbb{E}_{\boldsymbol{\theta} \sim q_t(\boldsymbol{\theta})}[-\log p(\mathbf{y}|\mathbf{x}, \theta)] + \mathrm{KL}(q_t(\boldsymbol{\theta} \parallel q_{t-1}(\boldsymbol{\theta})).$$

$$q_t(\boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(\theta_{t,d}; \mu_{t,d}, \sigma^2_{t,d}).$$

Nguyen et al. Variational Continual Learning. ICLR 2018.

NYU

Variational Continual Learning (VCL)

- Bayesian formulation:

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathcal{D}_t \mid \boldsymbol{\theta}) \propto p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T-1}) p(\mathcal{D}_T \mid \boldsymbol{\theta}).$$

- Variational approach:

$$q_t(\boldsymbol{\theta}) = \operatorname*{argmin}_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\boldsymbol{\theta}) \parallel \frac{1}{Z_t} q_{t-1}(\boldsymbol{\theta}) p(\mathcal{D}_t \mid \boldsymbol{\theta}) \right).$$

- Loss: $\mathcal{L}(q_t(\boldsymbol{\theta})) = \mathbb{E}_{\boldsymbol{\theta} \sim q_t(\boldsymbol{\theta})}[-\log p(\mathbf{y}|\mathbf{x}, \theta)] + \mathrm{KL}(q_t(\boldsymbol{\theta} \parallel q_{t-1}(\boldsymbol{\theta})).$

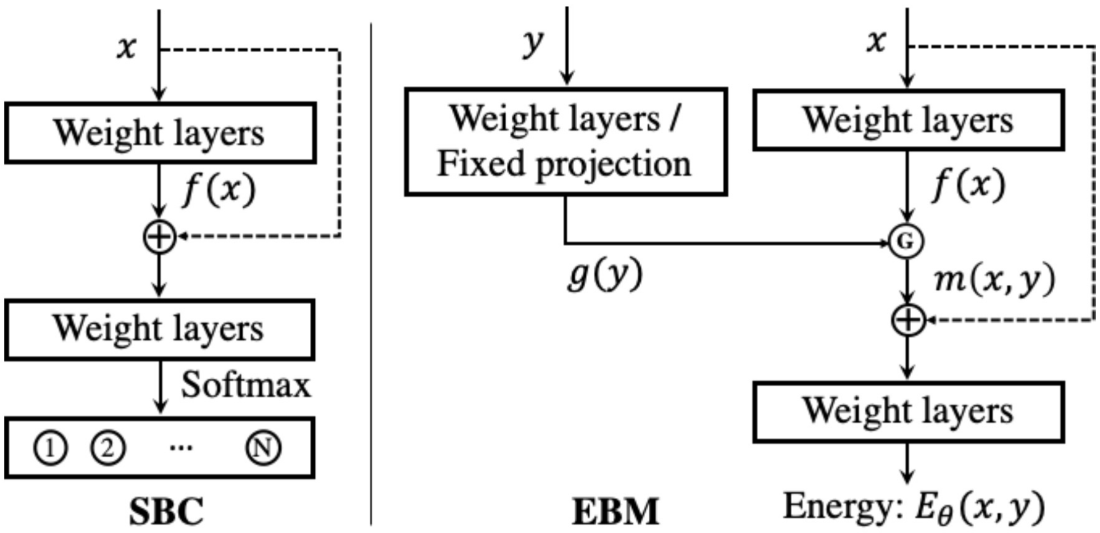
  $q_t(\boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(\theta_{t,d}; \mu_{t,d}, \sigma^2_{t,d}).$

- Compare to EWC: Maintains uncertainty throughout training.

Nguyen et al. Variational Continual Learning. ICLR 2018.

EBM for Continual Learning

- Softmax layer is known to be sensitive to distribution shift.

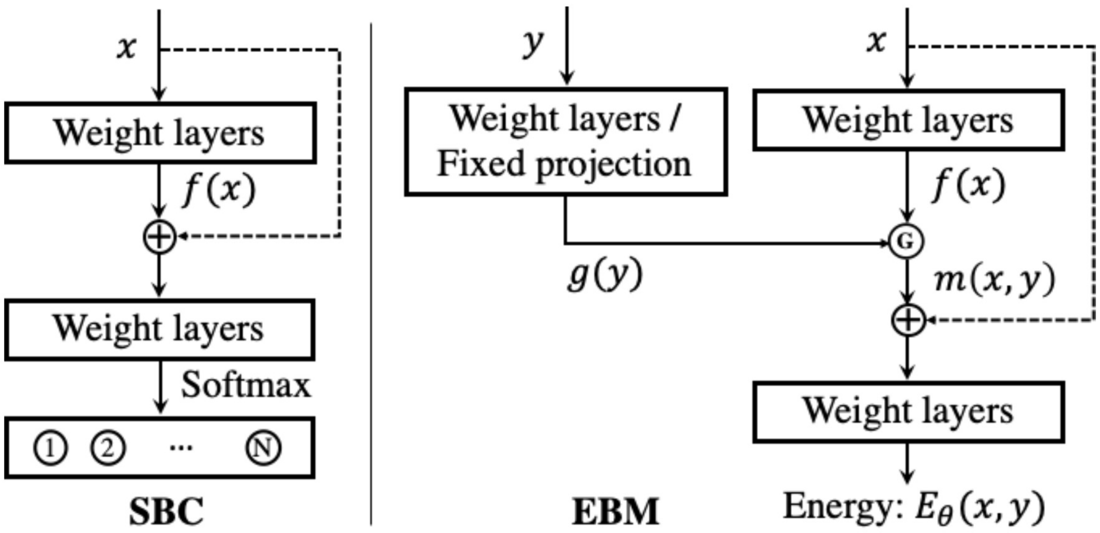


Li et al. Energy-Based Models for Continual Learning. CoLLAs 2022.

NYU

EBM for Continual Learning

- Softmax layer is known to be sensitive to distribution shift.
- A common approach is to use nearest mean classifier.

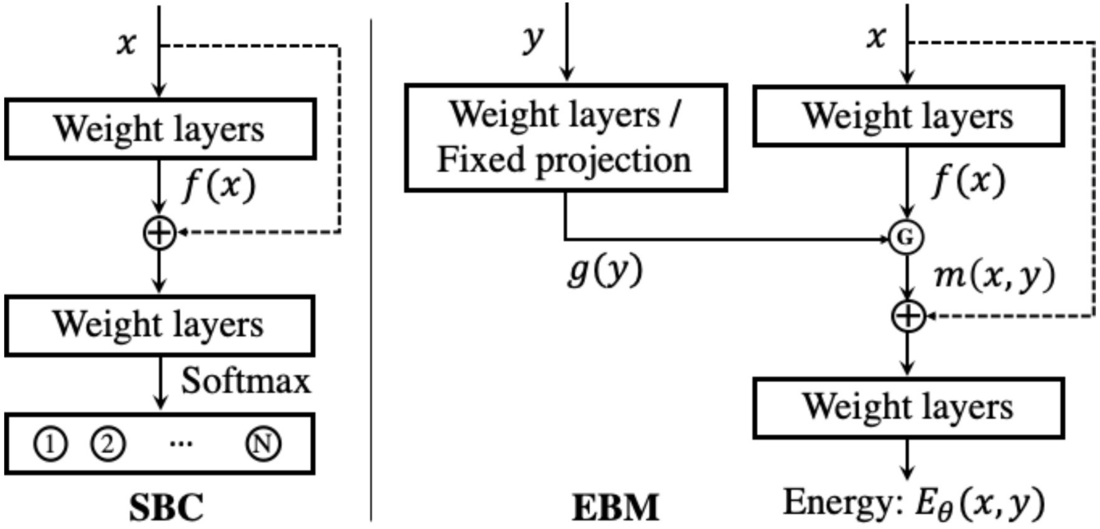


Li et al. Energy-Based Models for Continual Learning. CoLLAs 2022.

NYU

EBM for Continual Learning

- Softmax layer is known to be sensitive to distribution shift.
- A common approach is to use nearest mean classifier.
- Can be generalized to EBMs

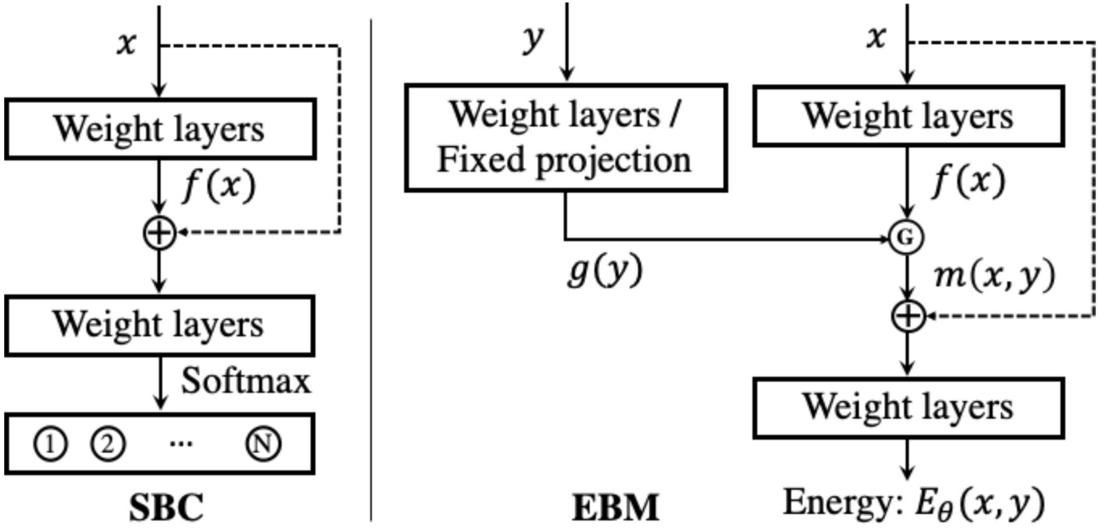


Li et al. Energy-Based Models for Continual Learning. CoLLAs 2022.

EBM for Continual Learning

- Softmax layer is known to be sensitive to distribution shift.
- A common approach is to use nearest mean classifier.
- Can be generalized to EBMs
- Energy between inputs and labels:

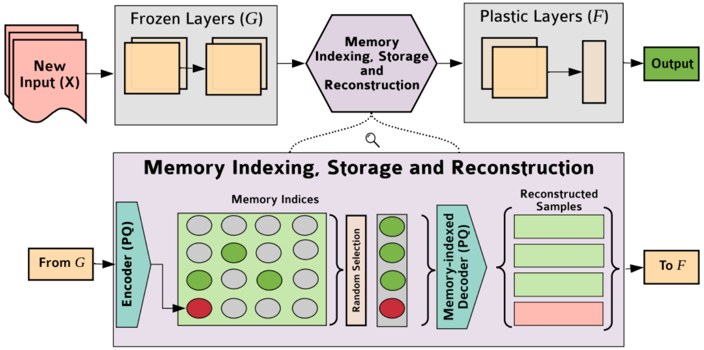
$$m(\mathbf{x}, y) = G(f(\mathbf{x}), g(y))$$



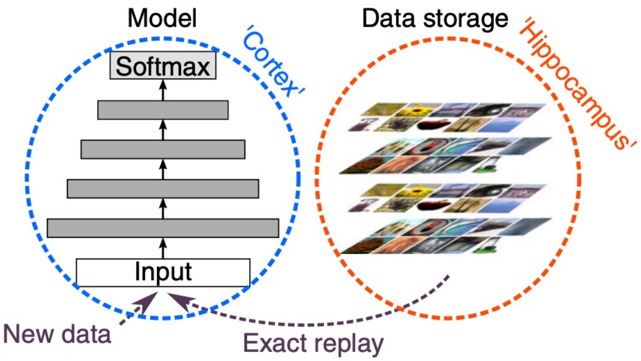
$$L_{CD}(\theta) = \mathbb{E}_{(\mathbf{x},y,y^-)\sim p_D}[E_\theta(\mathbf{x}, y) - E_\theta(\mathbf{x}, y^-)].$$

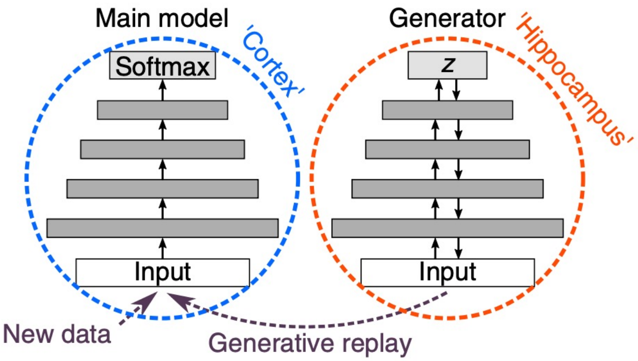
Li et al. Energy-Based Models for Continual Learning. CoLLAs 2022.

Replay

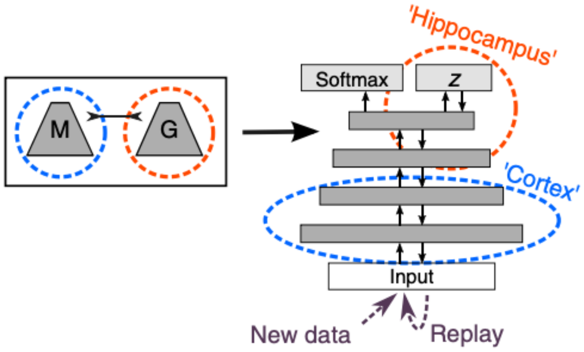


- Store raw data, representations, or train a generative model

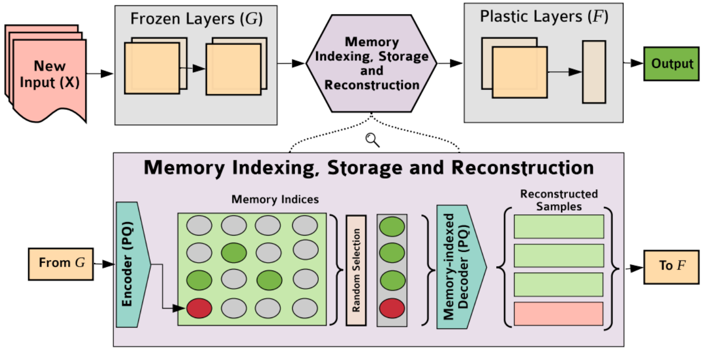




Feedback Connections

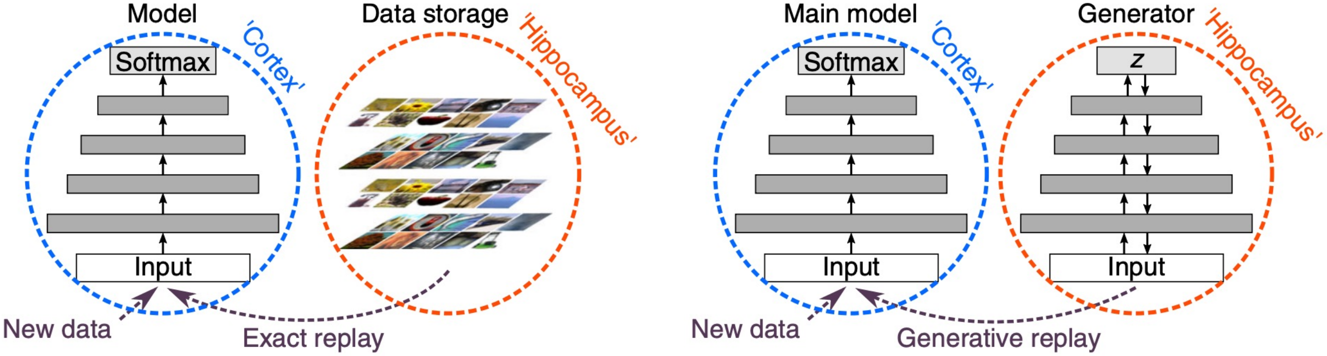


Rebuffi et al. iCaRL: Incremental Classifier and Representation Learning. CVPR 2017.
van de Ven et al. Brain-inspired replay for continual learning with artificial neural networks. Nature communications 2020.
Hayes et al. REMIND Your Neural Network to Prevent Catastrophic Forgetting. ECCV 2020.

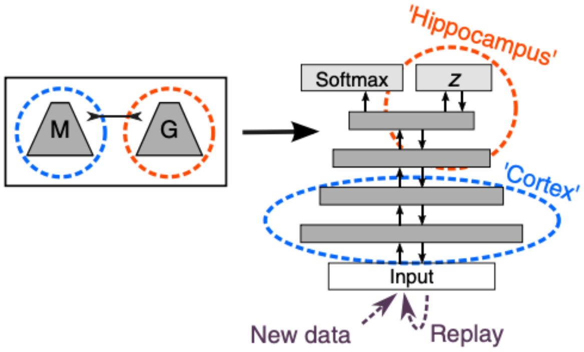


Replay

- Store raw data, representations, or train a generative model
- Coreset selection $\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x)$ $\quad p_k \leftarrow \operatorname*{argmin}_{x \in X} \left\| \mu - \frac{1}{k}[\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|.$



Feedback Connections



Rebuffi et al. iCaRL: Incremental Classifier and Representation Learning. CVPR 2017.
van de Ven et al. Brain-inspired replay for continual learning with artificial neural networks. Nature communications 2020.
Hayes et al. REMIND Your Neural Network to Prevent Catastrophic Forgetting. ECCV 2020.

NYU

Knowledge Distillation

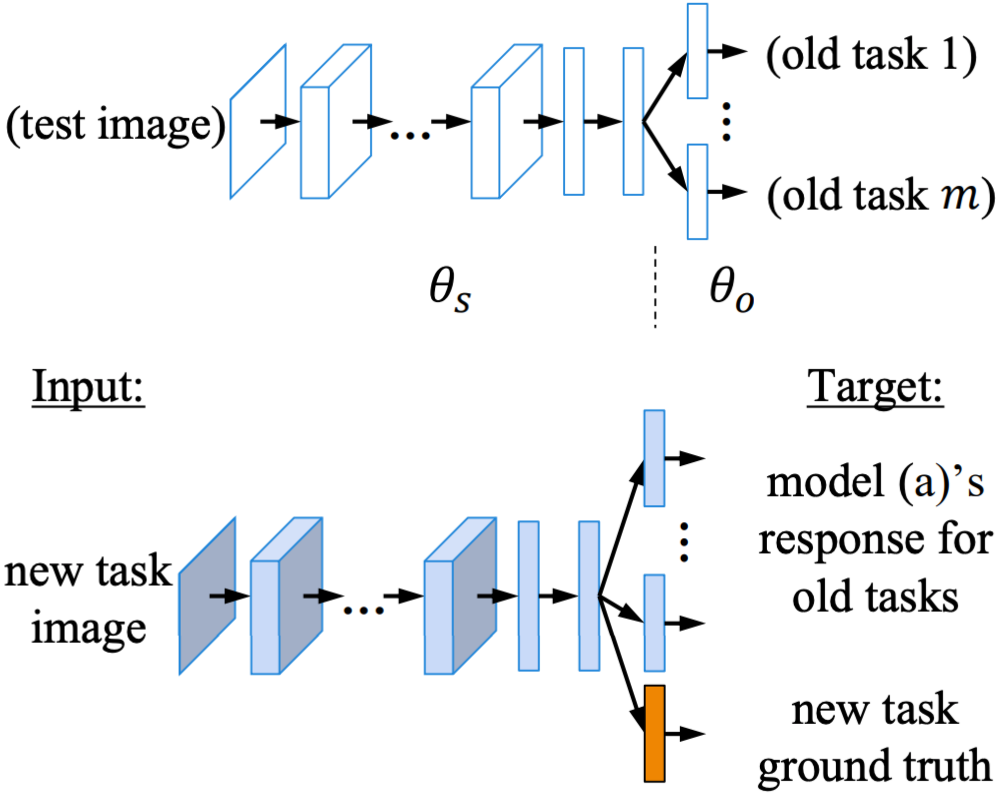
- Instead of saving the data points, we can also save the previous model checkpoint.

$$y_o = f(x_n; \theta_o).$$

$$\hat{y}_o = f(x_n; \theta_n).$$

$$\mathcal{L}(y_o, \hat{y}_o)$$

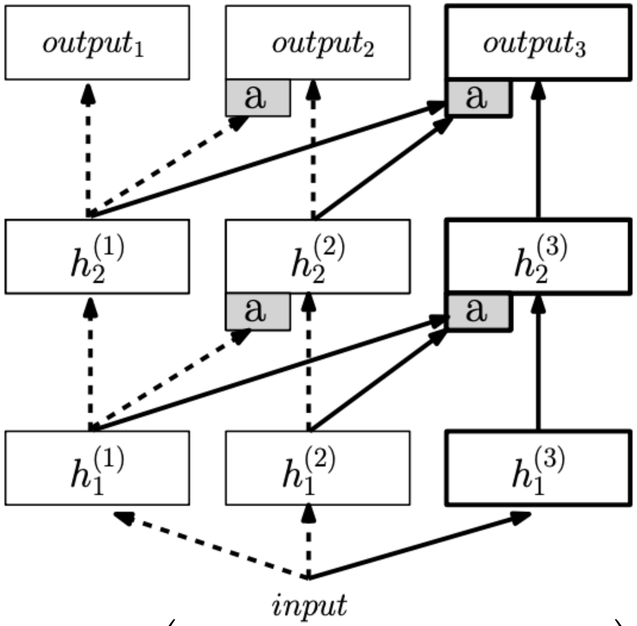
- Use new data points and old weights to "distill"



Li & Hoiem. Learning without Forgetting. ECCV 2016.

NYU

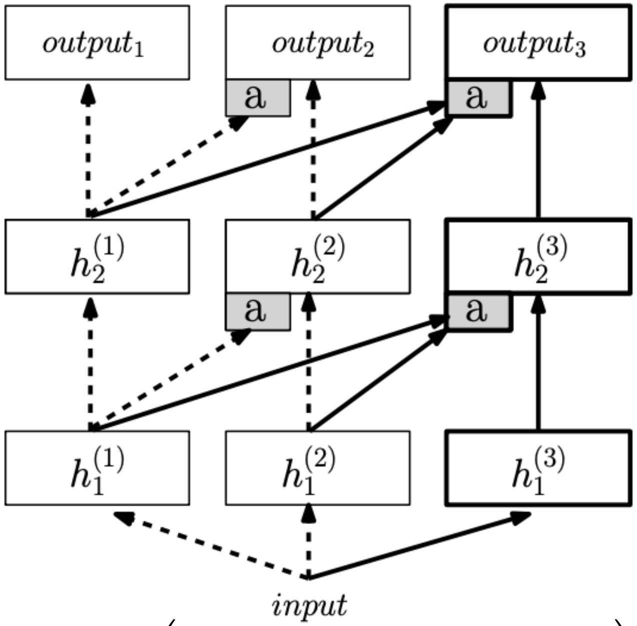
Architecture Expansion



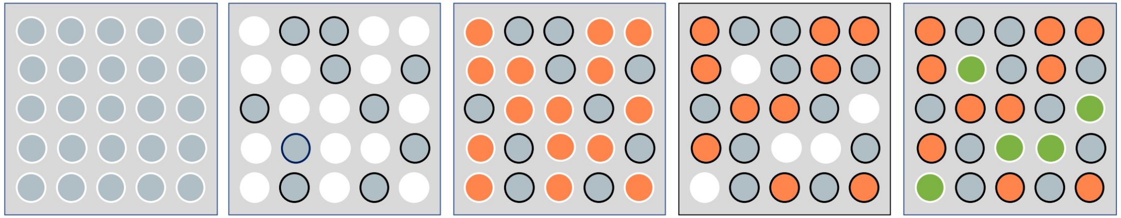
$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right).$$

Rusu et al. Progressive Neural Networks. NIPS 2016 Deep Learning Symposium.
PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. CVPR 2018.
Yoon et al. Lifelong Learning with Dynamically Expandable Networks. ICLR 2018.

Architecture Expansion



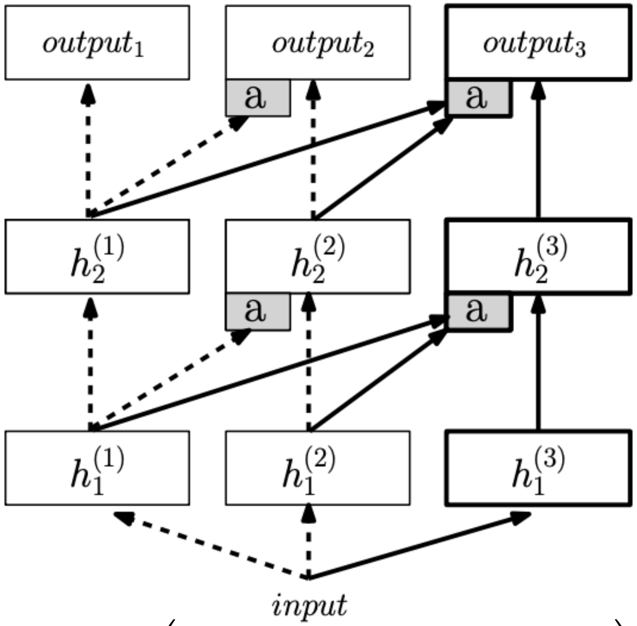
$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right).$$



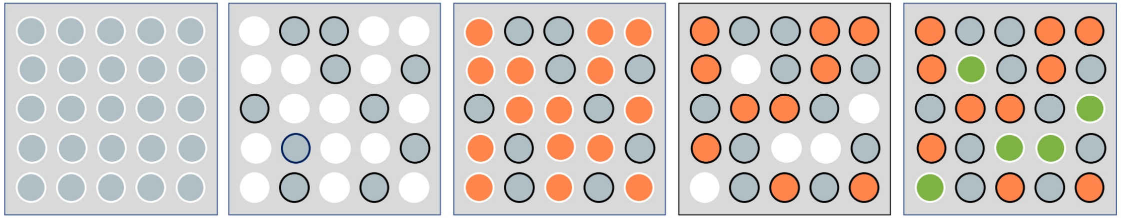
Rusu et al. Progressive Neural Networks. NIPS 2016 Deep Learning Symposium.
PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. CVPR 2018.
Yoon et al. Lifelong Learning with Dynamically Expandable Networks. ICLR 2018.

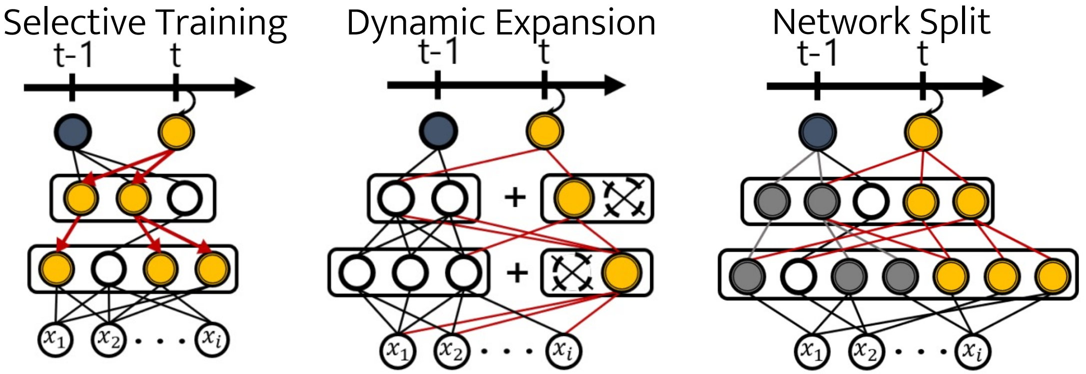
NYU

Architecture Expansion



$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right).$$





Selective Training

$$\mathcal{L}(\boldsymbol{W}_L^t, \boldsymbol{W}_{1:L-1}^{t-1}, \mathcal{D}_t) + \mu\|\boldsymbol{W}_L^t\|_1$$

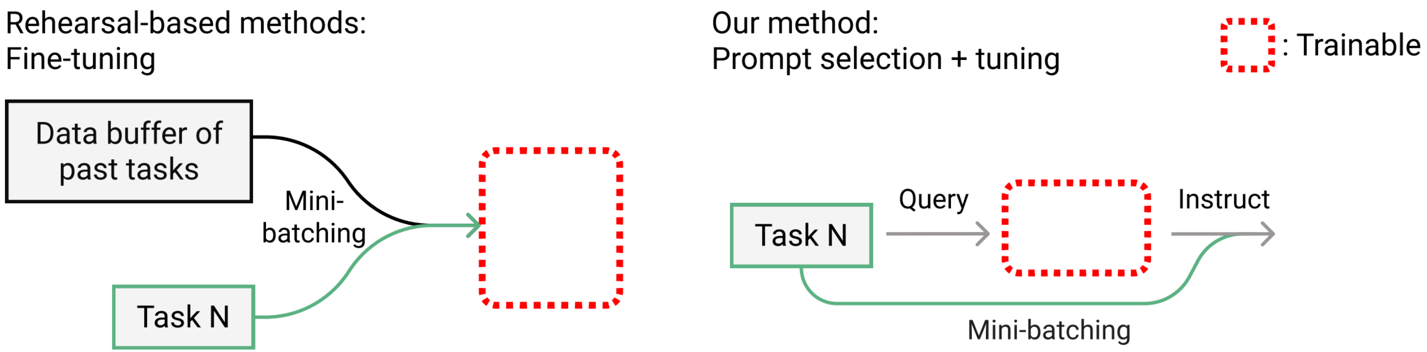
Dynamic Expansion | Group sparsity

$$\mathcal{L}(\boldsymbol{W}_l^{\mathcal{N}}, \boldsymbol{W}_l^{t-1}, \mathcal{D}_t) + \mu\|\boldsymbol{W}_l^{\mathcal{N}}\|_1 + \gamma\sum_g\|\boldsymbol{W}_{l,g}^{\mathcal{N}}\|_2$$

Rusu et al. Progressive Neural Networks. NIPS 2016 Deep Learning Symposium.
PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. CVPR 2018.
Yoon et al. Lifelong Learning with Dynamically Expandable Networks. ICLR 2018.

Adapting Pretrained Models

- Pretrained models have general knowledge that can be adapted to a continual stream of tasks.

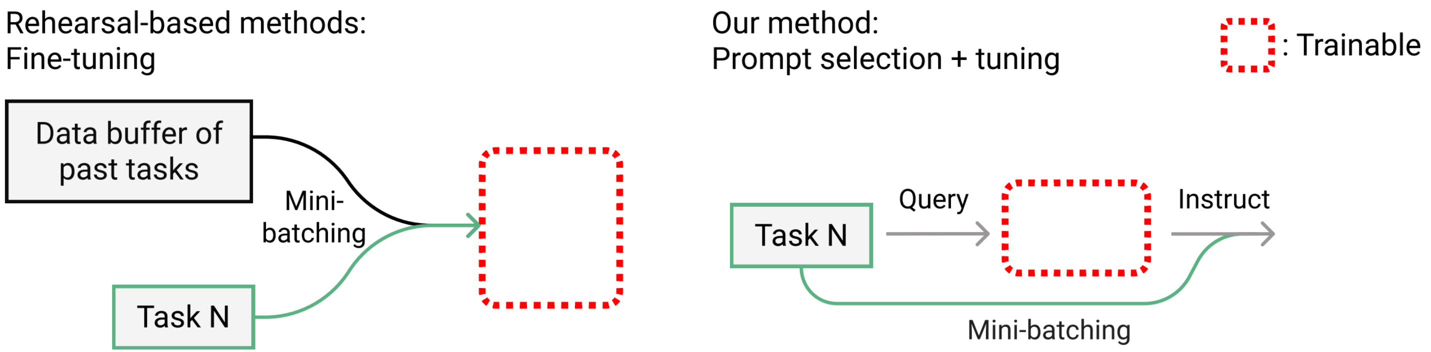


Wang et al. Learning to Prompt for Continual Learning. CVPR 2022.

NYU

Adapting Pretrained Models

- Pretrained models have general knowledge that can be adapted to a continual stream of tasks.
- Learn adaptation parameters for each task and store these as "task embeddings."

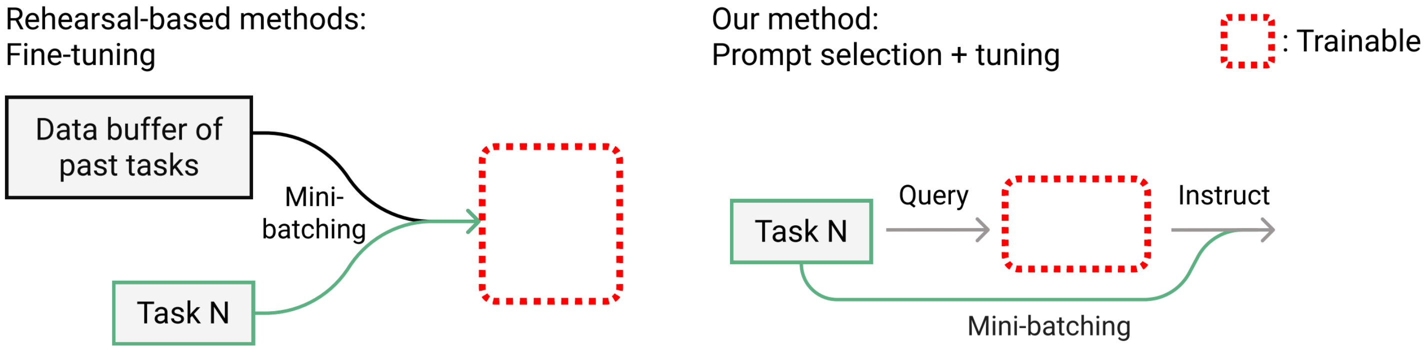


Wang et al. Learning to Prompt for Continual Learning. CVPR 2022.

NYU

Adapting Pretrained Models

- Pretrained models have general knowledge that can be adapted to a continual stream of tasks.
- Learn adaptation parameters for each task and store these as "task embeddings."
- Main model is frozen.



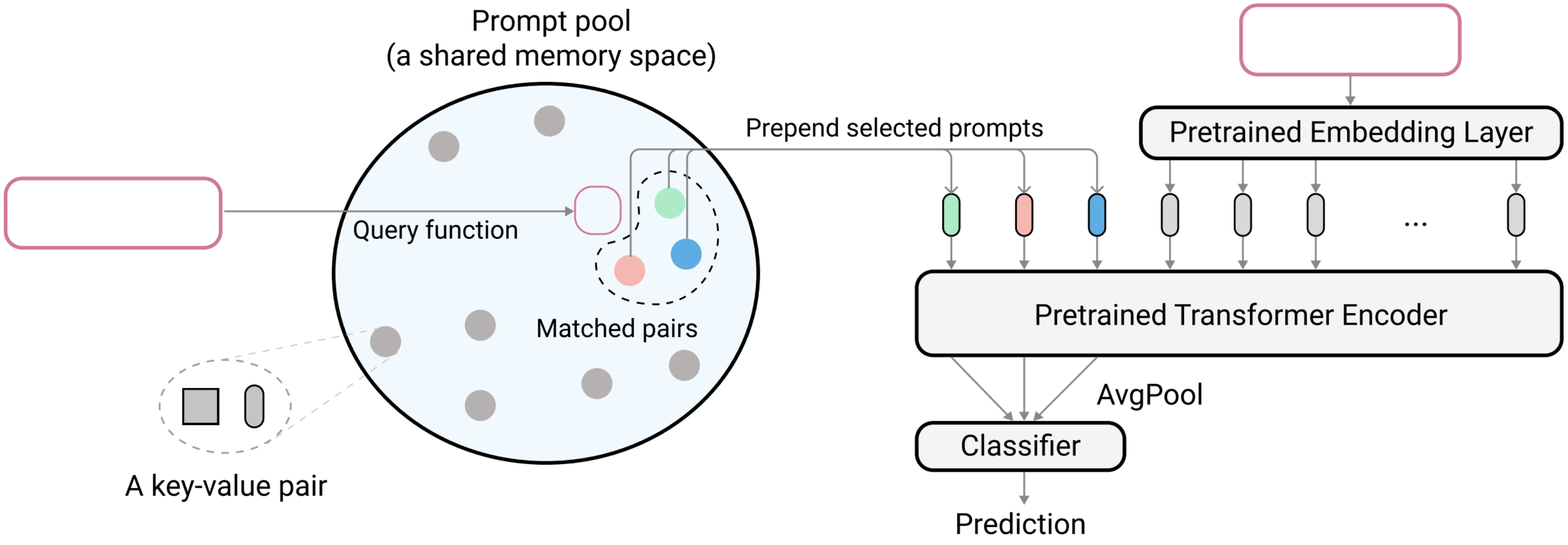
Wang et al. Learning to Prompt for Continual Learning. CVPR 2022.

NYU

Learning to Prompt

Prompt pool (slot memory)

$$\{(k_1, p_1), \ldots, (k_M, p_M)\}$$



Wang et al. Learning to Prompt for Continual Learning. CVPR 2022.

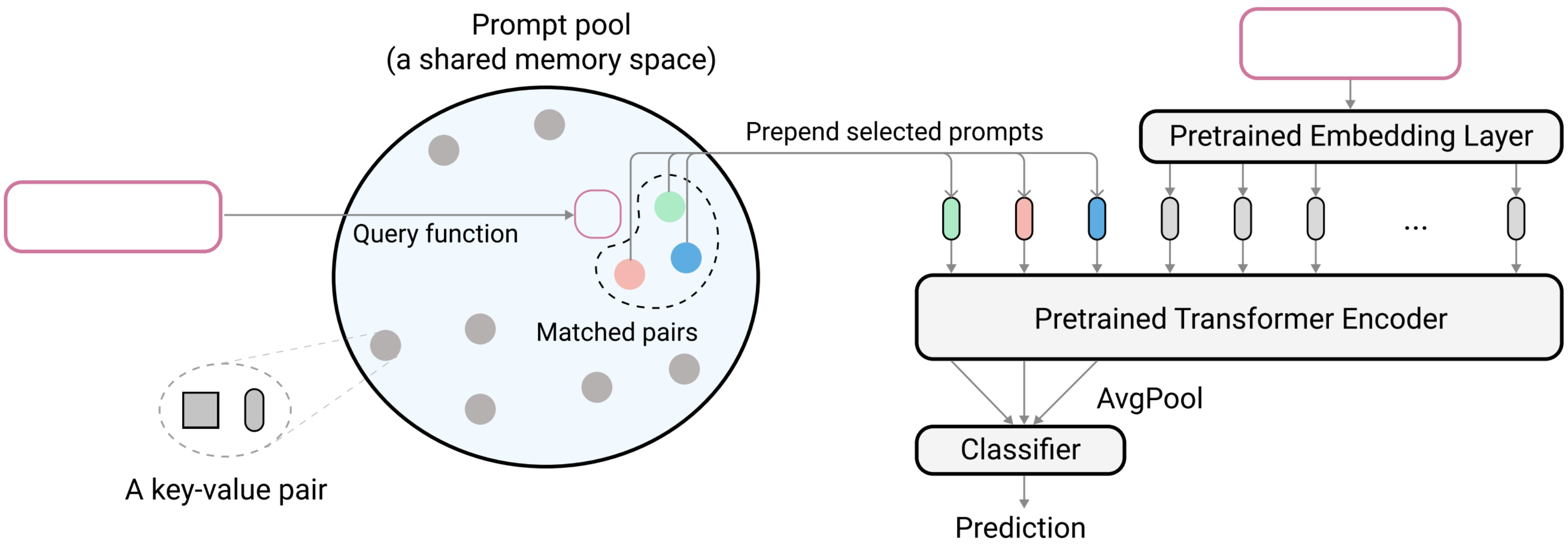
NYU

Learning to Prompt

Prompt pool (slot memory)

$$\{(k_1, p_1), \ldots, (k_M, p_M)\}$$

$$K_s = \operatorname{argmin}_{\mathcal{S}} \sum_{i \in \mathcal{S}} \gamma(q(x), k_i)$$



Wang et al. Learning to Prompt for Continual Learning. CVPR 2022.

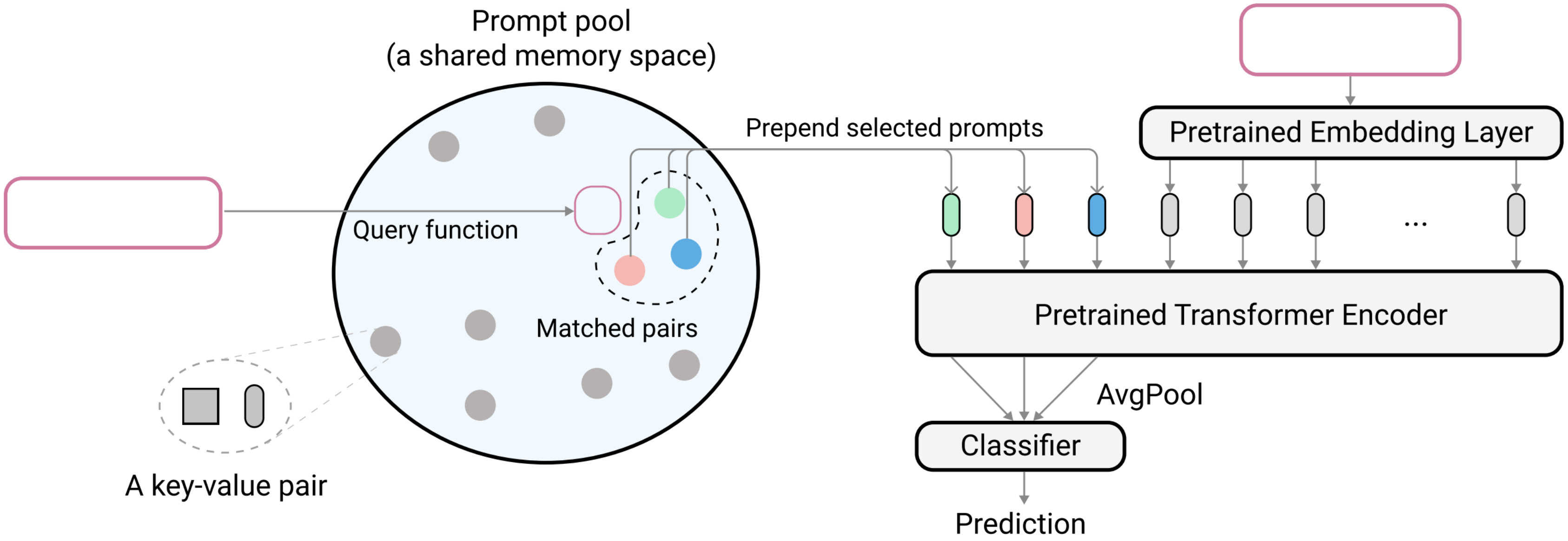
Learning to Prompt

Prompt pool (slot memory)

$$\{(k_1, p_1), \ldots, (k_M, p_M)\}$$

$$K_s = \text{argmin}_{\mathcal{S}} \sum_{i \in \mathcal{S}} \gamma(q(x), k_i)$$

$$\min_{P,K,\phi} \mathcal{L}(g_\phi(x), y) + \lambda \sum_{i \in K_s} \gamma(q(x), k_i)$$



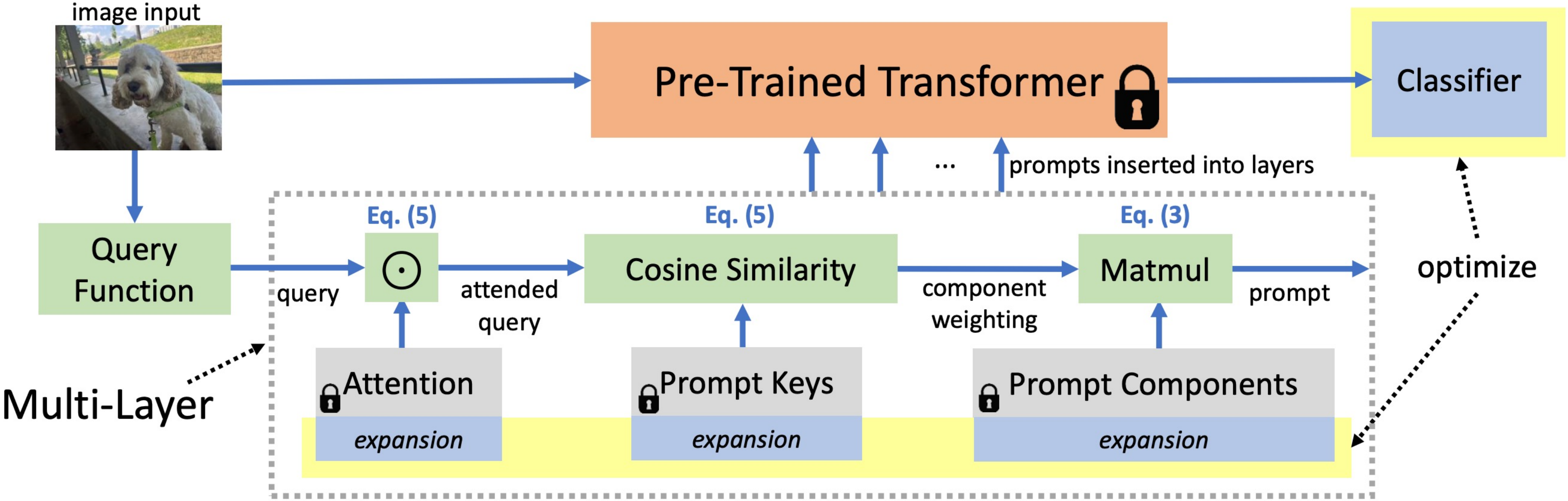
Wang et al. Learning to Prompt for Continual Learning. CVPR 2022.

NYU

Learned Prompt Query

- The query function can be end-to-end learned.

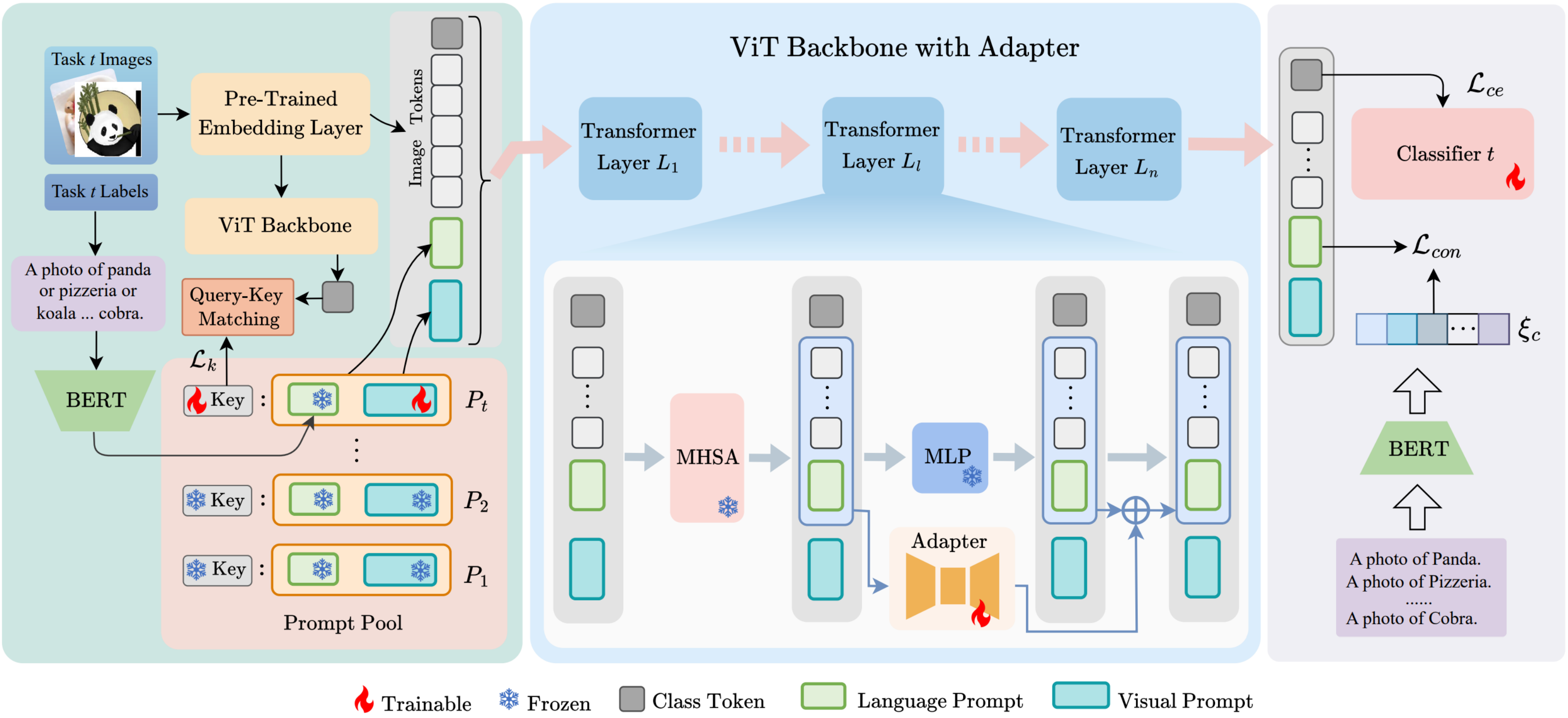
$$\alpha = \{\gamma(q(x) \odot A_1, K_1), \ldots, \gamma(q(x) \odot A_M, K_M)\}$$



Smith et al. CODA-Prompt: COntinual Decomposed Attention-based Prompting for Rehearsal-Free Continual Learning. CVPR 2023.

NYU

Multimodal Semantic Prompts



Yin et al. Adapter-Enhanced Semantic Prompting for Continual Learning. arXiv 2024.

Continual Learning and Memory

- Fragility of feedforward gradient descent of the entire networks

Continual Learning and Memory

- Fragility of feedforward gradient descent of the entire networks
- If we have representations ready, continual learning is just memorizing a sequence of new tasks.

Continual Learning and Memory

- Fragility of feedforward gradient descent of the entire networks
- If we have representations ready, continual learning is just memorizing a sequence of new tasks.
- In prompting approaches:
  - prompt pool = memory
  - pretrained network = representations

NYU

Continual Learning and Memory

- Fragility of feedforward gradient descent of the entire networks
- If we have representations ready, continual learning is just memorizing a sequence of new tasks.
- In prompting approaches:
  - prompt pool = memory
  - pretrained network = representations
- But what if representations also need to be built sequentially?

NYU

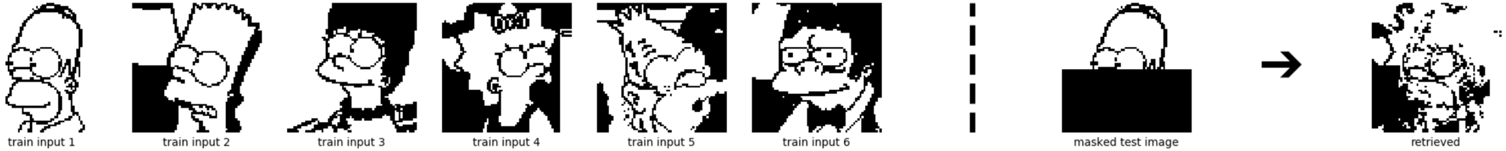
Continual Learning and Memory

- Fragility of feedforward gradient descent of the entire networks
- If we have representations ready, continual learning is just memorizing a sequence of new tasks.
- In prompting approaches:
  - prompt pool = memory
  - pretrained network = representations
- But what if representations also need to be built sequentially?
- It’s also plausible that representations are just “deeper memory.”

NYU

Associative Memory

- Memory aims to store content for easy retrieval

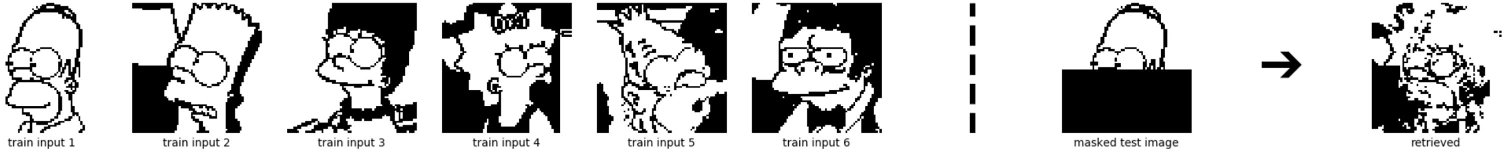


Krotov & Hopfield. Dense Associative Memory for Pattern Recognition. NIPS 2016.
https://ml-jku.github.io/hopfield-layers/

NYU

Associative Memory

- Memory aims to store content for easy retrieval
  - Associative memories (Hopfield Networks) can be viewed as energy-based models



Krotov & Hopfield. Dense Associative Memory for Pattern Recognition. NIPS 2016.
https://ml-jku.github.io/hopfield-layers/

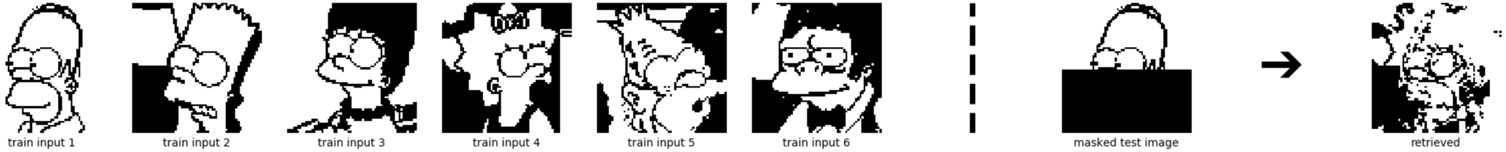
NYU

Associative Memory

- Memory aims to store content for easy retrieval
  - Associative memories (Hopfield Networks) can be viewed as energy-based models

$$E = -\frac{1}{2} \sum_{i,j=1}^{N} s_i W_{ij} s_j , \qquad W_{ij} = \sum_{k=1}^{K} \xi_i^k \xi_j^k .$$

"Superposition" of k slots
Hebbian learning



Krotov & Hopfield. Dense Associative Memory for Pattern Recognition. NIPS 2016.
https://ml-jku.github.io/hopfield-layers/

NYU

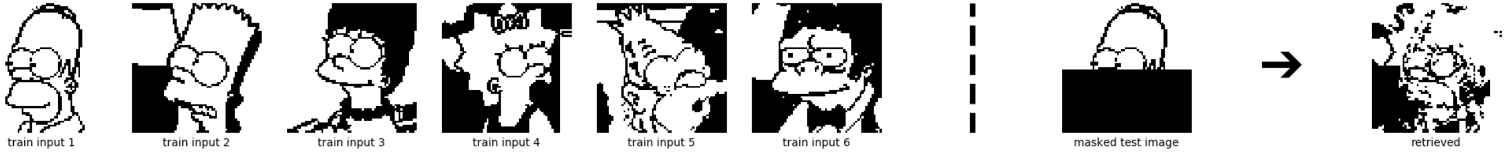
Associative Memory

- Memory aims to store content for easy retrieval
  - Associative memories (Hopfield Networks) can be viewed as energy-based models

$$E = -\frac{1}{2} \sum_{i,j=1}^{N} s_i W_{ij} s_j , \qquad W_{ij} = \sum_{k=1}^{K} \xi_i^k \xi_j^k .$$

"Superposition" of k slots
Hebbian learning

  - When presented with a new pattern the network should respond with a stored memory which most closely resembles the input.



Krotov & Hopfield. Dense Associative Memory for Pattern Recognition. NIPS 2016.
https://ml-jku.github.io/hopfield-layers/

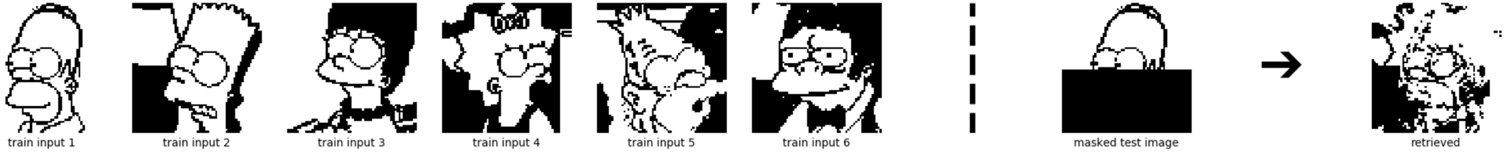
Associative Memory

- Memory aims to store content for easy retrieval
  - Associative memories (Hopfield Networks) can be viewed as energy-based models

$$E = -\frac{1}{2}\sum_{i,j=1}^{N} s_i W_{ij} s_j, \qquad W_{ij} = \sum_{k=1}^{K} \xi_i^k \xi_j^k .$$

”Superposition” of k slots
Hebbian learning

  - When presented with a new pattern the network should respond with a stored memory which most closely resembles the input.
  - Retrieval: $s_i = \mathrm{sign}(\sum_j W_{ij} s_j)$ Storage: $C \approx \frac{d}{2\log(d)} = 0.14d.$

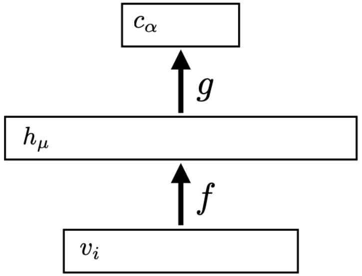


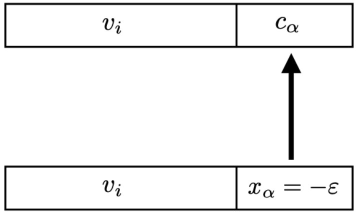
Krotov & Hopfield. Dense Associative Memory for Pattern Recognition. NIPS 2016.
https://ml-jku.github.io/hopfield-layers/

Memory Duality

- Duality with a feedforward network.

$$E = -\sum_k F(\sum_i \xi_i^k s_i)$$





Salakhutdinov & Hinton. Deep Boltzmann Machines. AISTATS 2009.
Krotov. Hierarchical Associative Memory. arXiv 2021.

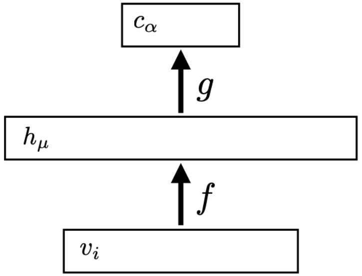
NYU

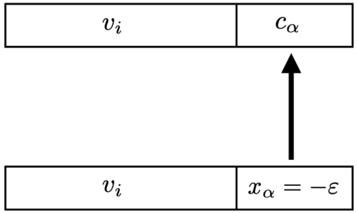
Memory Duality

- Duality with a feedforward network.

$$E = -\sum_k F(\sum_i \xi_i^k s_i)$$

- Non-linearity allows us to store more patterns.





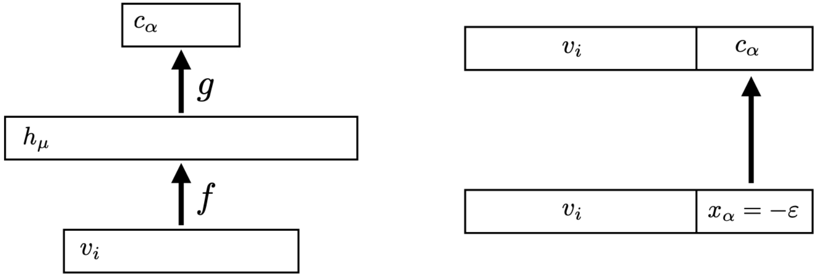
Salakhutdinov & Hinton. Deep Boltzmann Machines. AISTATS 2009.
Krotov. Hierarchical Associative Memory. arXiv 2021.

Memory Duality

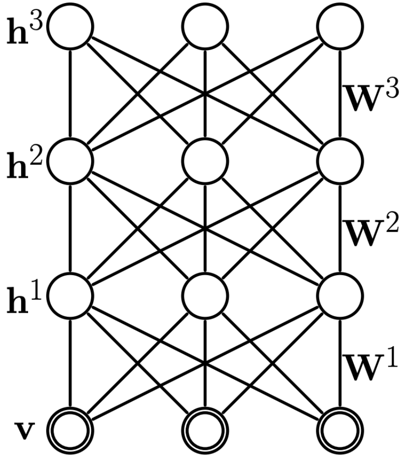
- Duality with a feedforward network.

$$E = -\sum_k F\left(\sum_i \xi_i^k s_i\right)$$

- Non-linearity allows us to store more patterns.
- Deep Boltzmann machines



Deep Boltzmann Machine



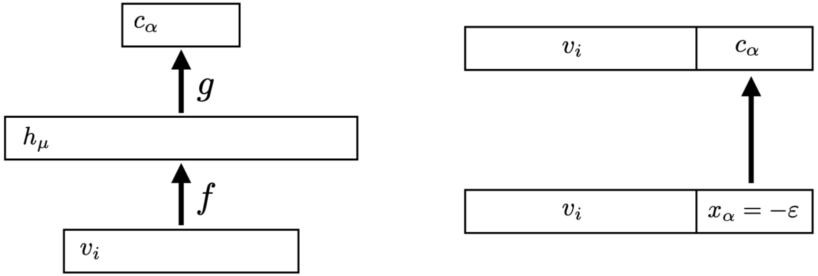
Salakhutdinov & Hinton. Deep Boltzmann Machines. AISTATS 2009.
Krotov. Hierarchical Associative Memory. arXiv 2021.

Memory Duality

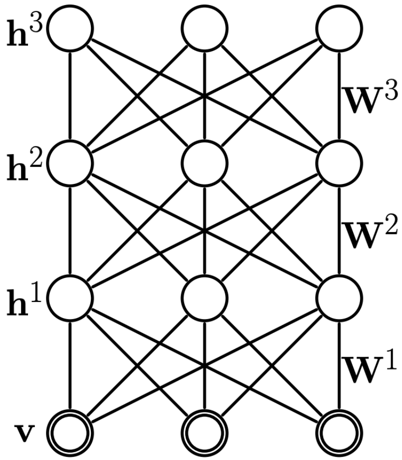
- Duality with a feedforward network.

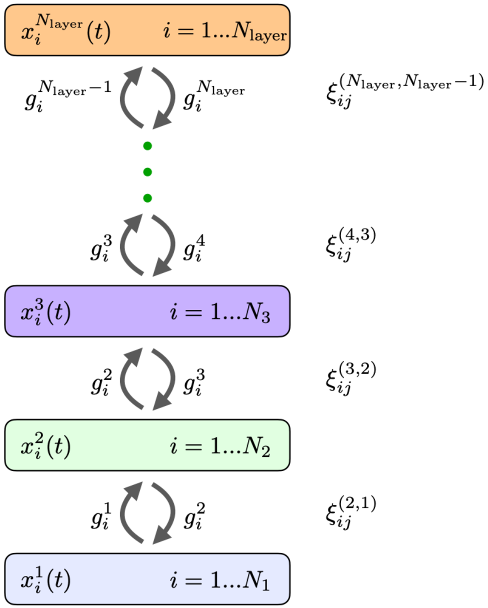
$$E = -\sum_k F\left(\sum_i \xi_i^k s_i\right)$$

- Non-linearity allows us to store more patterns.
- Deep Boltzmann machines
- Hierarchical associative memory



Deep Boltzmann Machine





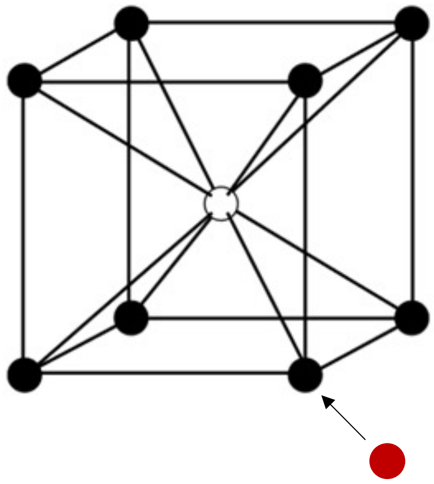
Salakhutdinov & Hinton. Deep Boltzmann Machines. AISTATS 2009.
Krotov. Hierarchical Associative Memory. arXiv 2021.

Relation to Transformers

- General form:

$$E = -\sum_k F(\sum_i \xi_i^k s_i).$$

Relation to Transformers



- General form:

$$E = -\sum_k F\Big(\sum_i \xi_i^k s_i\Big).$$

- When $F(z) = z^2$ it gives the classic HN.

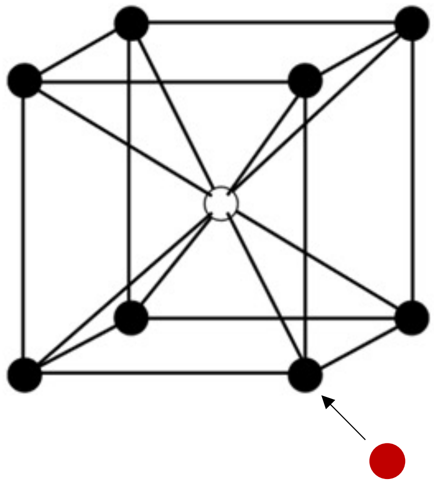
$$\nabla_{s_i} E = -\sum_j W_{ij} s_j$$

$$s_i \leftarrow \mathrm{sign}\big(\textstyle\sum_j W_{ij} s_j\big)$$

Ramsauer et al. Hopfield Networks is All You Need. ICLR 2021.

NYU

Relation to Transformers



- General form:

$$E = -\sum_k F(\sum_i \xi_i^k s_i).$$

- When $F(z) = z^2$ it gives the classic HN.
- Transformer-like attention operation:

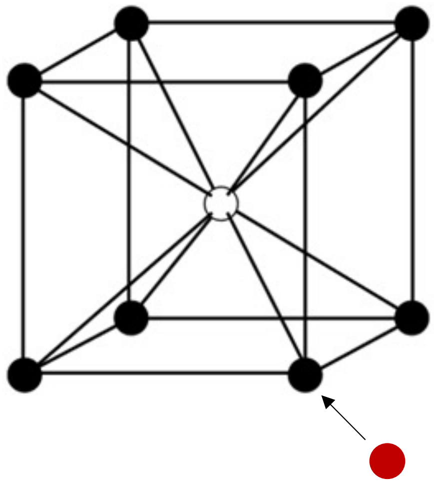
$$\boldsymbol{Z} \leftarrow \mathrm{softmax}(\beta \boldsymbol{X} \boldsymbol{W}_q \boldsymbol{W}_k^\top \boldsymbol{Y}^\top) \boldsymbol{Y}_i \boldsymbol{W}_v.$$

$$\nabla_{s_i} E = -\sum_j W_{ij} s_j$$

$$s_i \leftarrow \mathrm{sign}(\textstyle\sum_j W_{ij} s_j)$$

NYU

Relation to Transformers



- General form:

$$E = -\sum_k F(\sum_i \xi_i^k s_i).$$

- When $F(z) = z^2$ it gives the classic HN.
- Transformer-like attention operation:

$$\boldsymbol{Z} \leftarrow \mathrm{softmax}(\beta \boldsymbol{X} \boldsymbol{W}_q \boldsymbol{W}_k^\top \boldsymbol{Y}^\top) \boldsymbol{Y}_i \boldsymbol{W}_v.$$

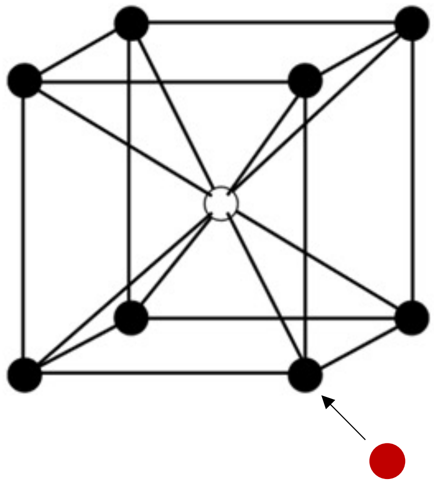
$$\boldsymbol{S} \leftarrow \mathrm{softmax}(\beta \boldsymbol{S} \boldsymbol{\Xi}^\top) \boldsymbol{\Xi}.$$

$$\nabla_{s_i} E = -\sum_j W_{ij} s_j$$

$$s_i \leftarrow \mathrm{sign}(\textstyle\sum_j W_{ij} s_j)$$

Ramsauer et al. Hopfield Networks is All You Need. ICLR 2021.

Relation to Transformers



- General form:

$$E = -\sum_k F(\sum_i \xi_i^k s_i).$$

- When $F(z) = z^2$ it gives the classic HN.
- Transformer-like attention operation:

$$\boldsymbol{Z} \leftarrow \mathrm{softmax}(\beta \boldsymbol{X} \boldsymbol{W}_q \boldsymbol{W}_k^\top \boldsymbol{Y}^\top) \boldsymbol{Y}_i \boldsymbol{W}_v.$$

$$\boldsymbol{S} \leftarrow \mathrm{softmax}(\beta \boldsymbol{S} \boldsymbol{\Xi}^\top) \boldsymbol{\Xi}.$$

$$E = -\mathrm{logsumexp}(\beta, \boldsymbol{\Xi}^\top \boldsymbol{s}) + \frac{1}{2} \boldsymbol{s}^\top \boldsymbol{s} + \beta^{-1} \log N + \frac{1}{2} M^2.$$

$$\nabla_{s_i} E = -\sum_j W_{ij} s_j$$

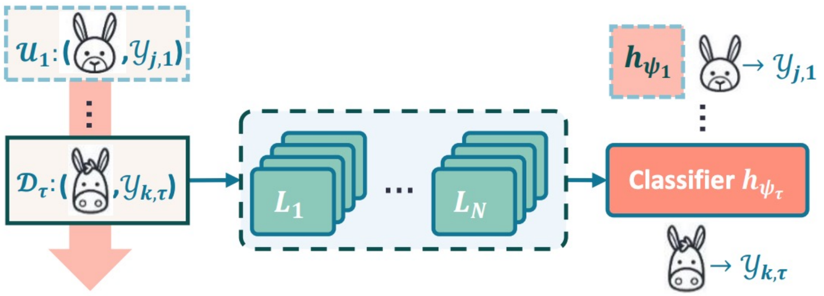
$$s_i \leftarrow \mathrm{sign}(\textstyle\sum_j W_{ij} s_j)$$

Ramsauer et al. Hopfield Networks is All You Need. ICLR 2021.

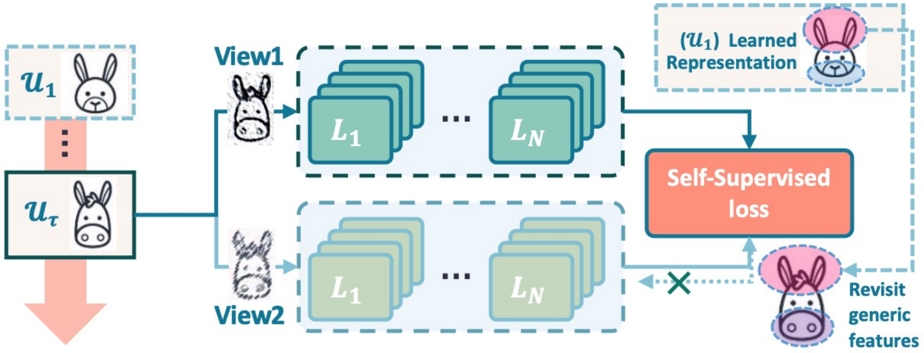
NYU

Continual Self-Supervised Learning

- Learning from a stream of unlabeled inputs.



SUPERVISED CONTINUAL LEARNING (SCL)



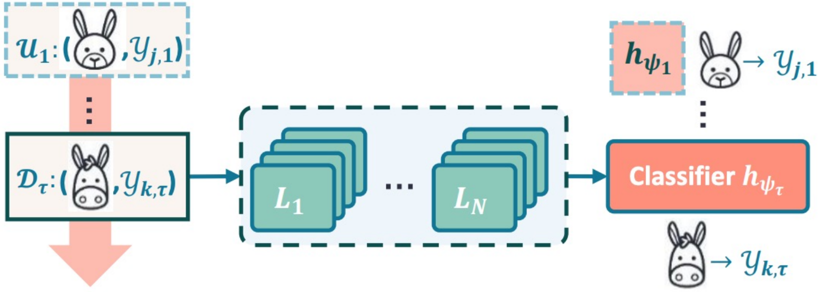
UNSUPERVISED CONTINUAL LEARNING (UCL)

Madaan et al. Representational Continuity for Unsupervised Continual Learning. ICLR 2022.

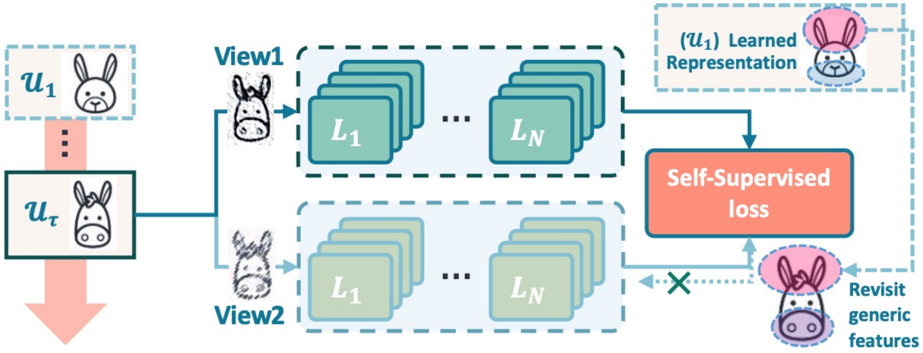
NYU

Continual Self-Supervised Learning

- Learning from a stream of unlabeled inputs.
- Bring SSL to the dynamic world.



SUPERVISED CONTINUAL LEARNING (SCL)

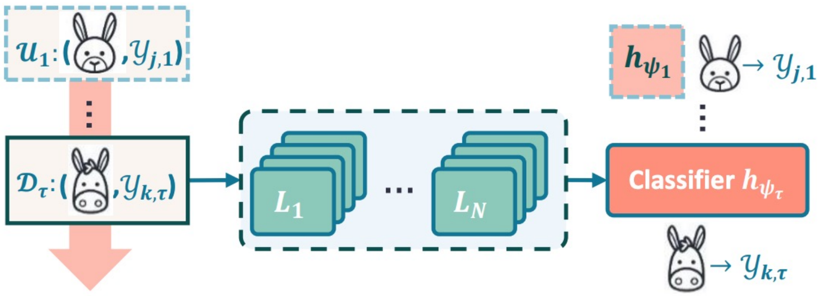


UNSUPERVISED CONTINUAL LEARNING (UCL)

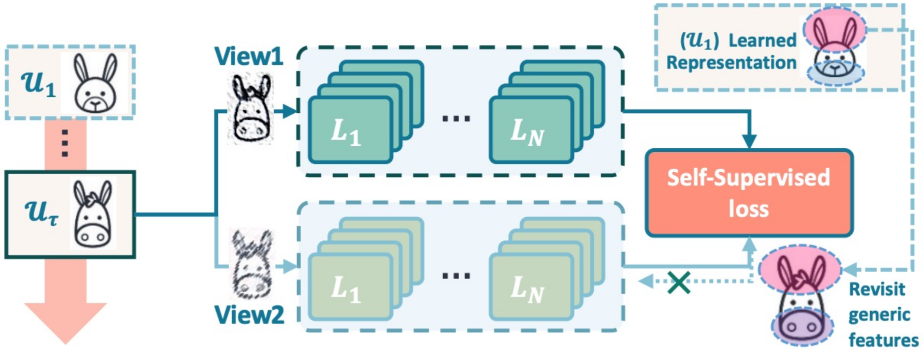
Madaan et al. Representational Continuity for Unsupervised Continual Learning. ICLR 2022.

Continual Self-Supervised Learning

- Learning from a stream of unlabeled inputs.
- Bring SSL to the dynamic world.
- SSL can still suffer from distributional shifts.



SUPERVISED CONTINUAL LEARNING (SCL)



UNSUPERVISED CONTINUAL LEARNING (UCL)

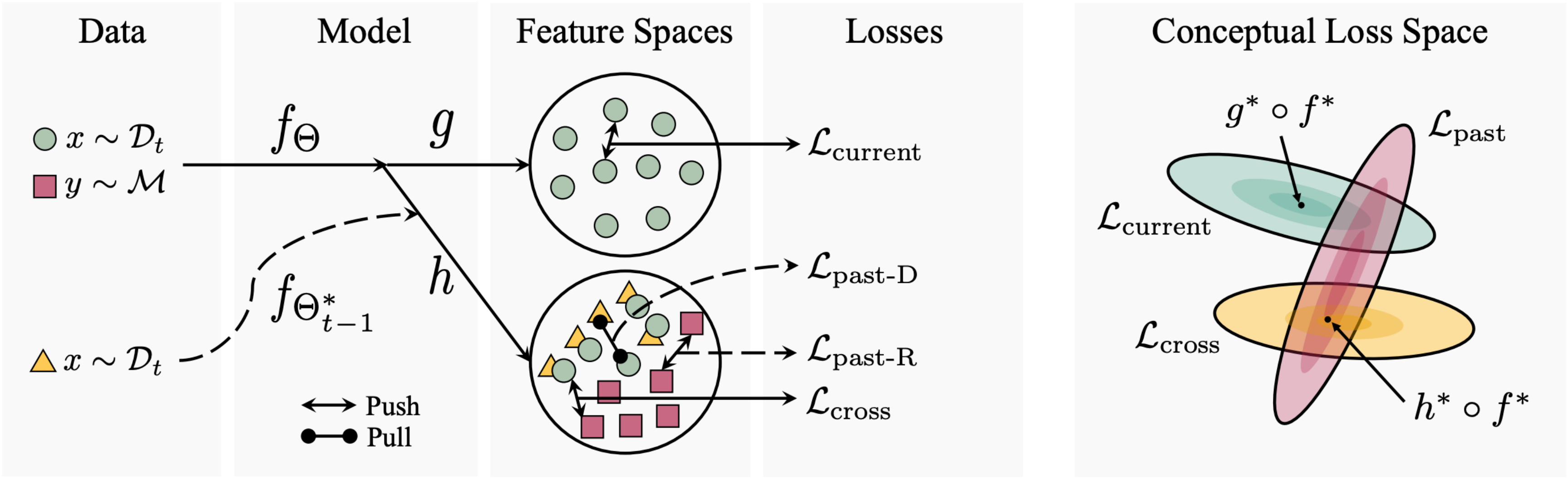
Madaan et al. Representational Continuity for Unsupervised Continual Learning. ICLR 2022.

NYU

Continual Self-Supervised Learning

- Integrating learning objectives of both past and present.

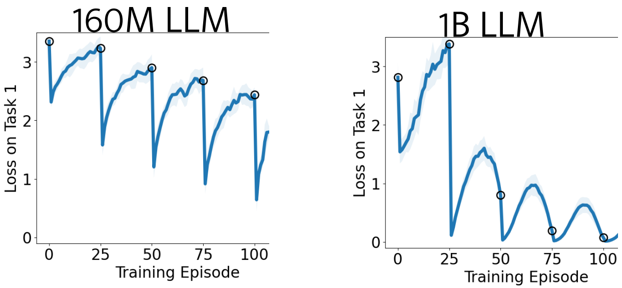
$$\underbrace{\mathcal{L}(X; g \circ f_t)}_{\text{Current}} + \underbrace{\mathcal{L}(X, X; h \circ f_{t-1}, h \circ f_t)}_{\text{Past (Distillation)}} + \underbrace{\mathcal{L}(Y; h \circ f_t)}_{\text{Past (Replay)}} + \underbrace{\mathcal{L}(X, Y; h \circ f_t)}_{\text{Cross-Consolidation}}$$



Zhang et al. Integrating Present and Past in Unsupervised Continual Learning. CoLLAs 2024.

Outlooks

- Understand continual learning at scale

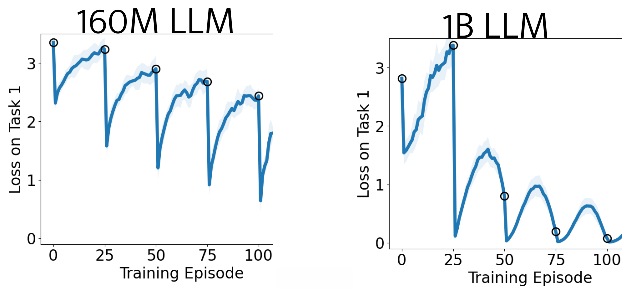


[Mayo et al. 2023]

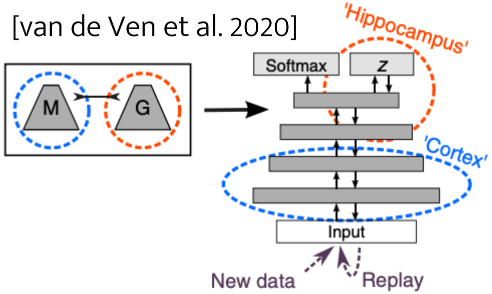
Outlooks

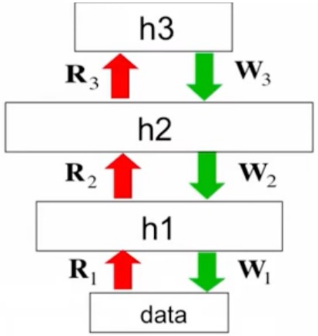
- Understand continual learning at scale

- Unified learning architecture, objective and replay, role of sleep



[van de Ven et al. 2020]



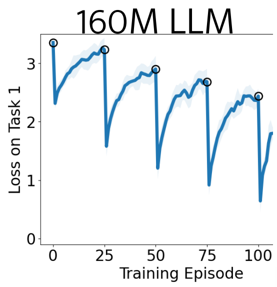


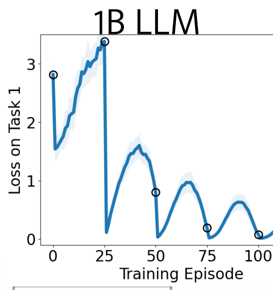
[Hinton et al. 1995]

[Mayo et al. 2023]

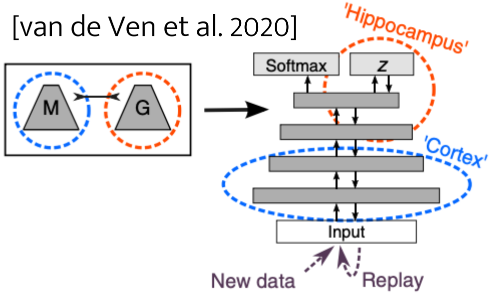
Outlooks

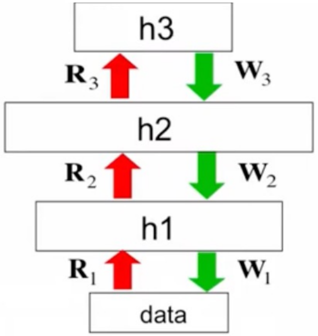
- Understand continual learning at scale
- Unified learning architecture, objective and replay, role of sleep
- Continual learning with real world structure





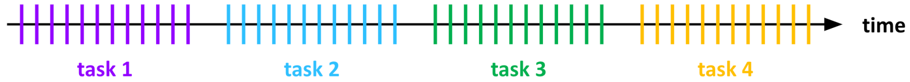
[van de Ven et al. 2020]





[Hinton et al. 1995]

Typical setting in continual, few-shot, transfer, and meta-learning



Typical setting of human learning [Mayo et al. 2023]

Tasks are interspersed and recur
No opportunity to master one before confronted with another

Summary: Continual Learning

- Regularization, Distillation, Architecture Expansion/Isolation

Summary: Continual Learning

- Regularization, Distillation, Architecture Expansion/Isolation
- Frozen representation: prompt learning

Summary: Continual Learning

- Regularization, Distillation, Architecture Expansion/Isolation
- Frozen representation: prompt learning
- Integration of memory and representations

Summary: Continual Learning

- Regularization, Distillation, Architecture Expansion/Isolation
- Frozen representation: prompt learning
- Integration of memory and representations
- Combination with self-supervised learning

Summary: Continual Learning

- Regularization, Distillation, Architecture Expansion/Isolation
- Frozen representation: prompt learning
- Integration of memory and representations
- Combination with self-supervised learning
- Exploration of multimodal continual learning from embodied environments