DS-GA.3001 Embodied Learning and Vision

Mengye Ren

NYU

Spring 2025

embodied-learning-vision-course.github.io

Lecture Slides for Note Taking

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.

What is Continual Learning?

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

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.

NYU

What is Continual Learning?

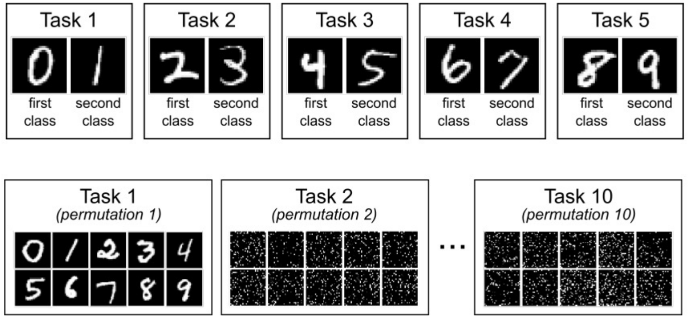
- Distribution shift: Forgetting
  - Learning on A and then B, results in worse performance on A.
- Multi-task learning: Forward transfer
  - Learning Task A + B results in better learning in Task C compared to learning C alone.
  - Leverage the similarity between tasks.
- Compositionality
  - Learning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

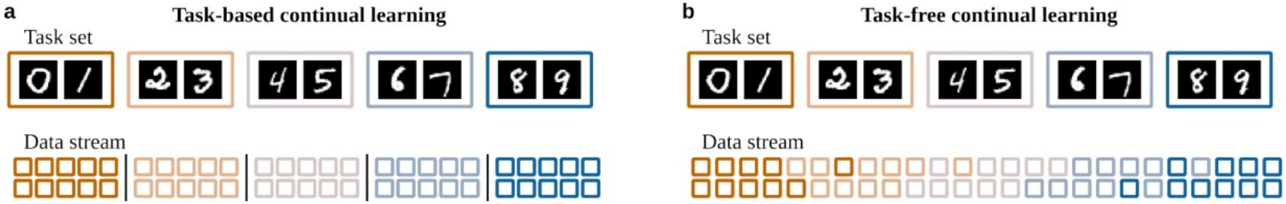
What is Continual Learning?

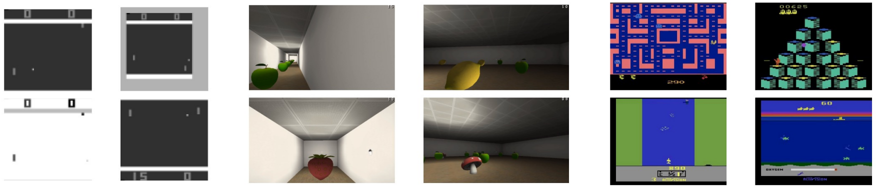
- Distribution shift: Forgetting
  - Learning on A and then B, results in worse performance on A.
- Multi-task learning: Forward transfer
  - Learning Task A + B results in better learning in Task C compared to learning C alone.
  - Leverage the similarity between tasks.
- Compositionality
  - Learning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

Continual Learning

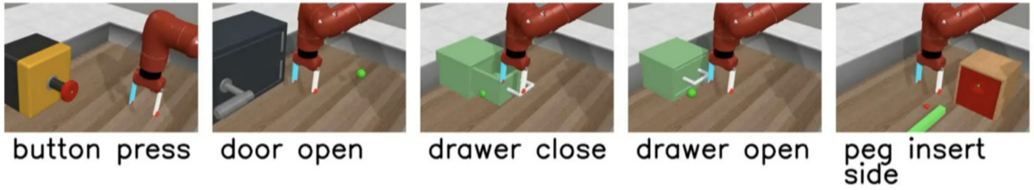
- Learning a sequence of tasks without looking back.







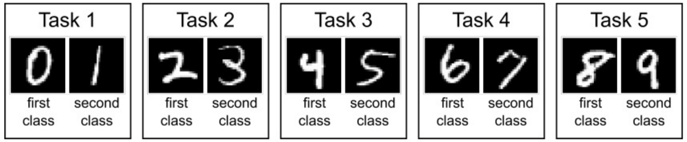
(a) Pong variants (b) Labyrinth games (c) Atari games

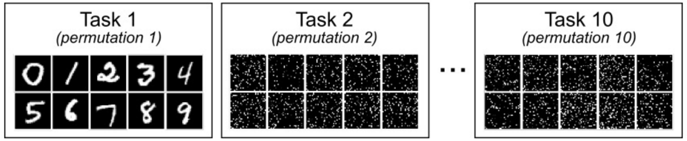


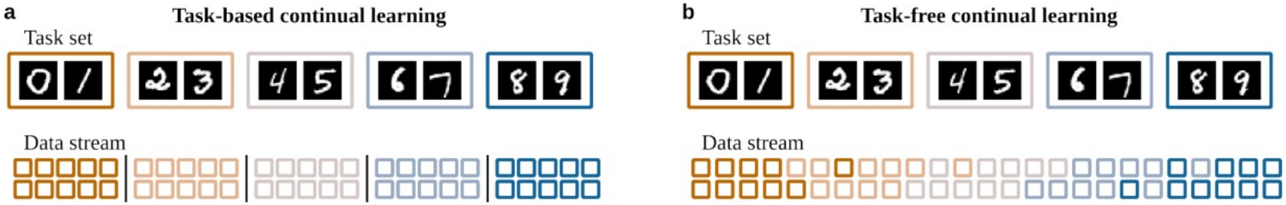
van de Ven et al. Continual learning and catastrophic forgetting. arXiv 2024.
van de Ven & Tolias. Three scenarios for continual learning. arXiv 2019.

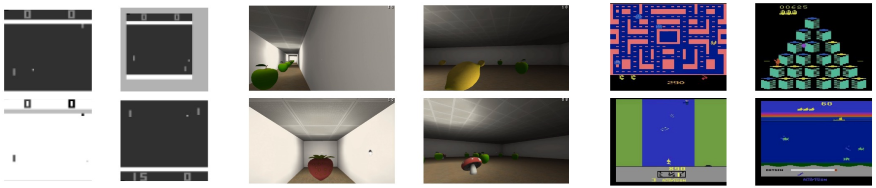
Continual Learning

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

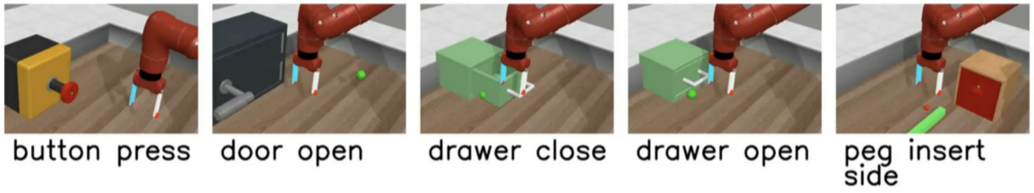








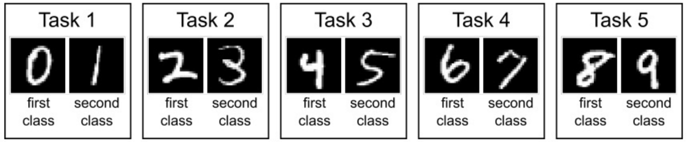
(a) Pong variants (b) Labyrinth games (c) Atari games

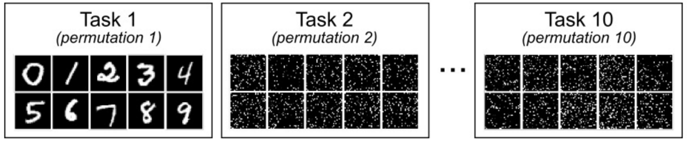


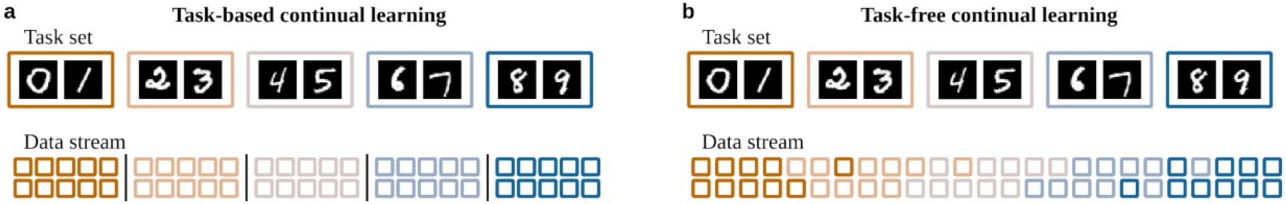
van de Ven et al. Continual learning and catastrophic forgetting. arXiv 2024.
van de Ven & Tolias. Three scenarios for continual learning. arXiv 2019.

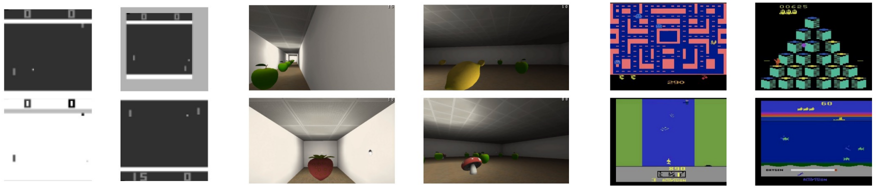
Continual Learning

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

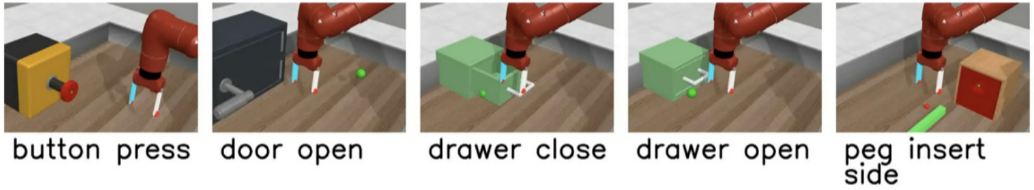








(a) Pong variants (b) Labyrinth games (c) Atari games

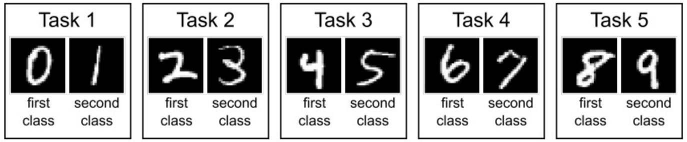


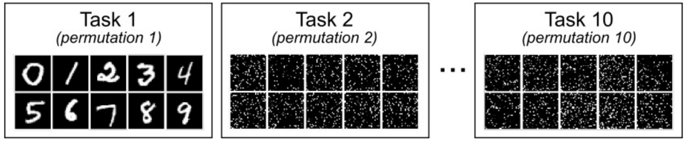
van de Ven et al. Continual learning and catastrophic forgetting. arXiv 2024.
van de Ven & Tolias. Three scenarios for continual learning. arXiv 2019.

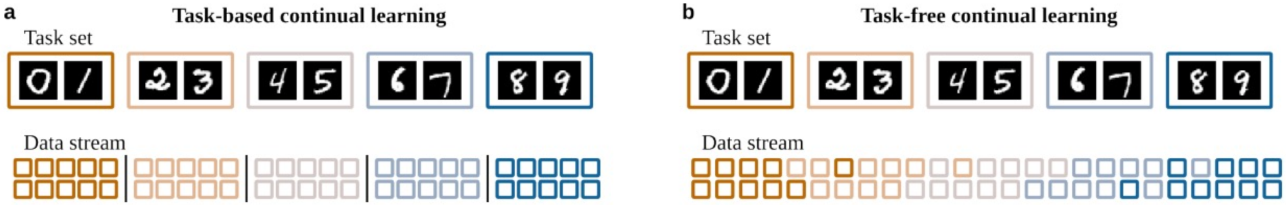
NYU

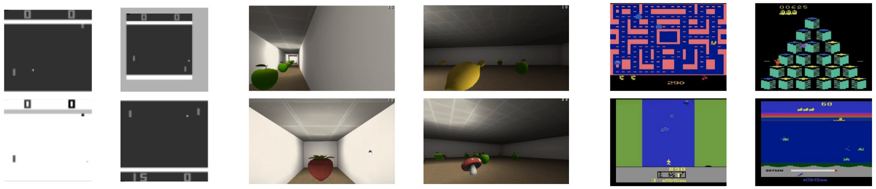
Continual Learning

- 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

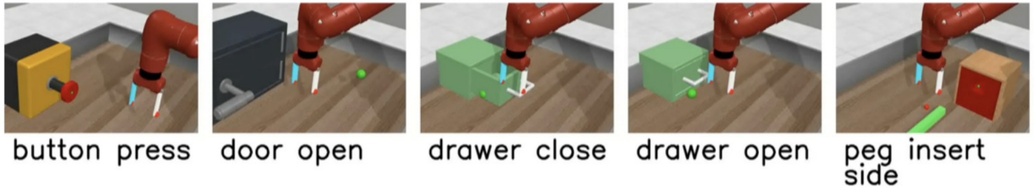








(a) Pong variants (b) Labyrinth games (c) Atari games



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

NYU

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.

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

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

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

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

NYU

Parameter Regularization

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

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

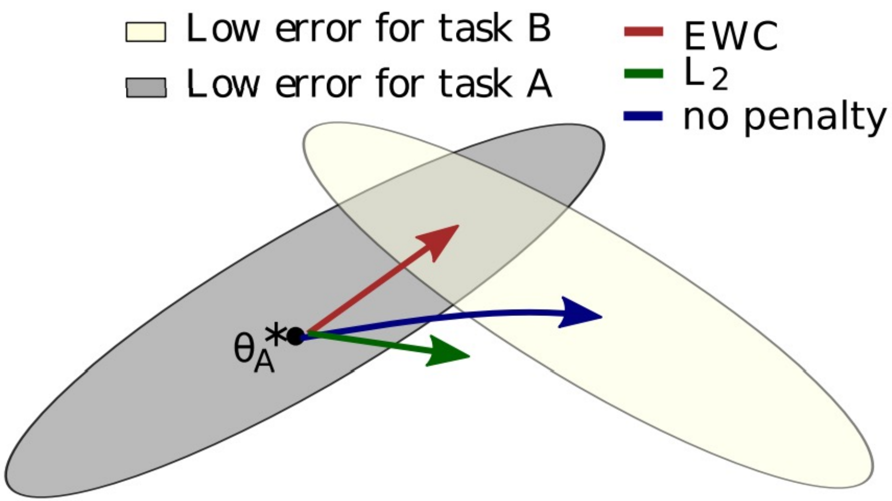
$$\mathcal{S}_A \cap \mathcal{S}_B \neq \emptyset$$

- 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):

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

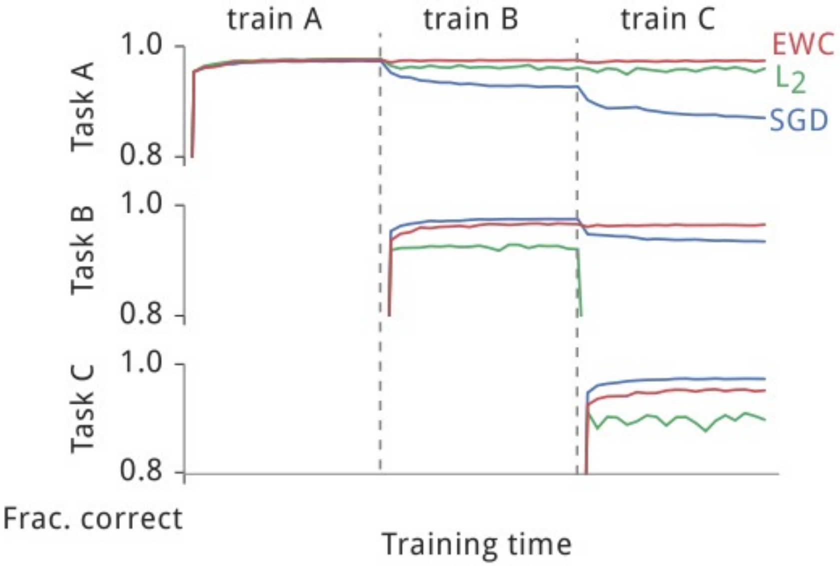


Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. PNAS 2017.

Computing Fisher

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

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



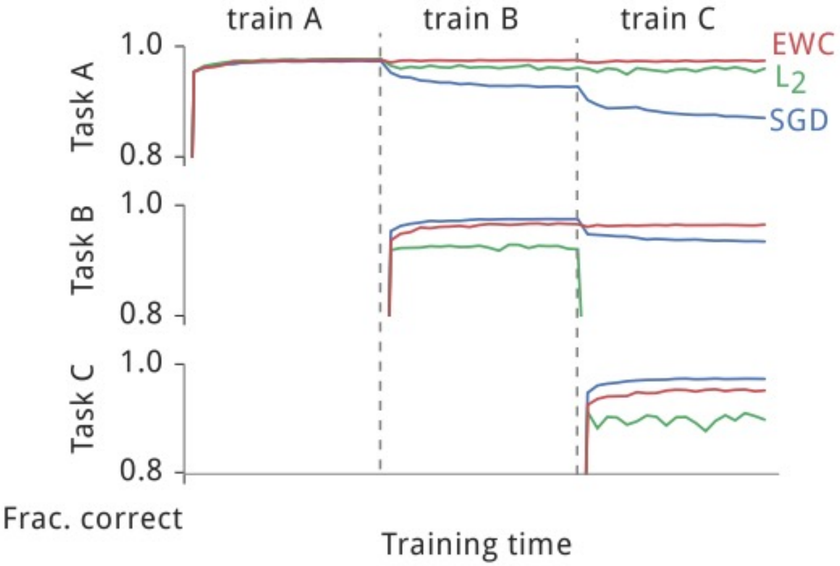
Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. PNAS 2017.
Zenke et al. Continual Learning Through Synaptic Intelligence. ICML 2017.

NYU

Computing Fisher

- At the end of each epoch, compute the gradient squared:
$$F_i = \left(\frac{d\mathcal{L}}{d\theta_i}\right)^2$$

- 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.

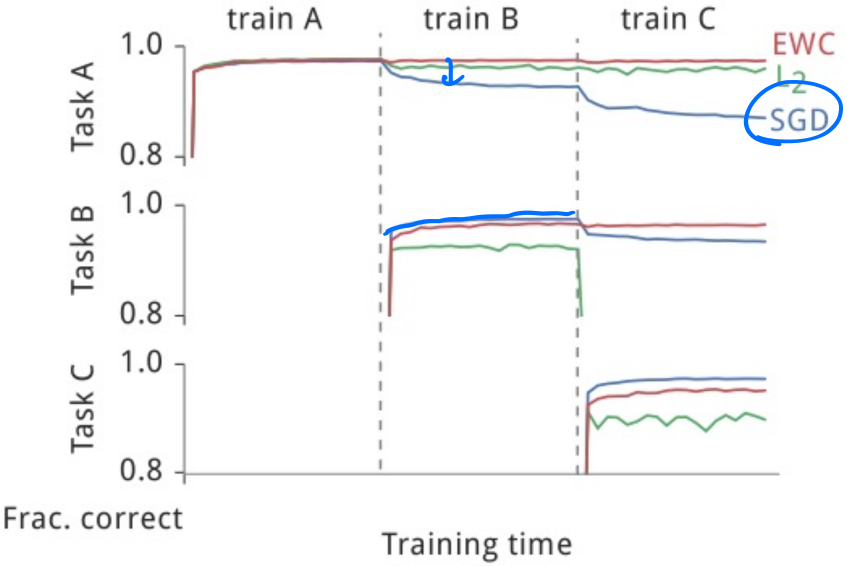
NYU

Computing Fisher

- 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:

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

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:

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

likelihood for the current dataset.

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:

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

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

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:

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

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

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

Nguyen et al. Variational Continual Learning. ICLR 2018.

NYU

Variational Continual Learning (VCL)

- Bayesian formulation:

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

θ_t

F_i

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

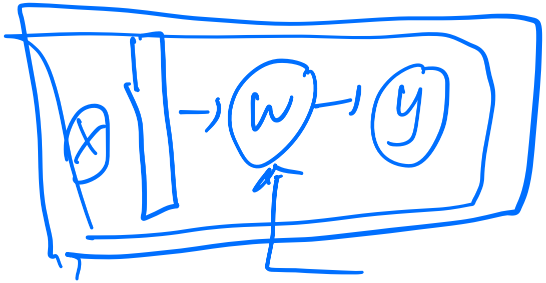
$$q_t(\boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(\theta_{t,d}; \mu_{t,d}, \sigma^2_{t,d}).$$

d^2

- Compare to EWC: Maintains uncertainty throughout training.

Nguyen et al. Variational Continual Learning. ICLR 2018.

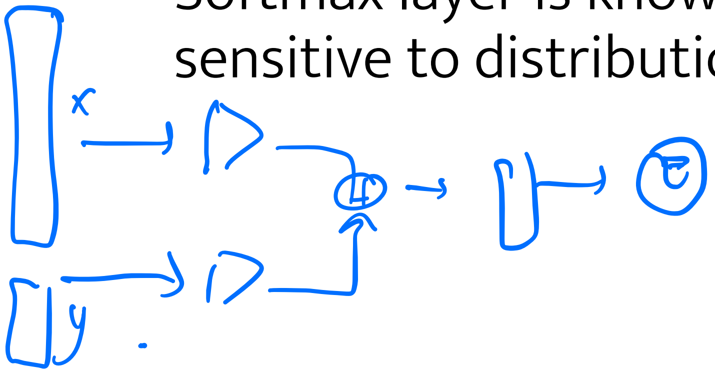
NYU

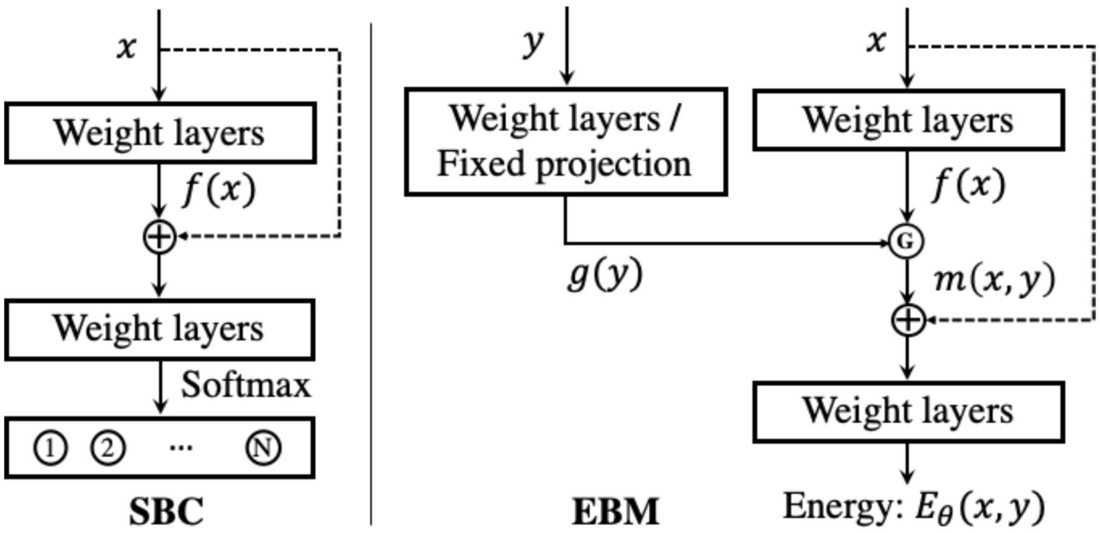


EBM for Continual Learning

$E(x, y).$

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



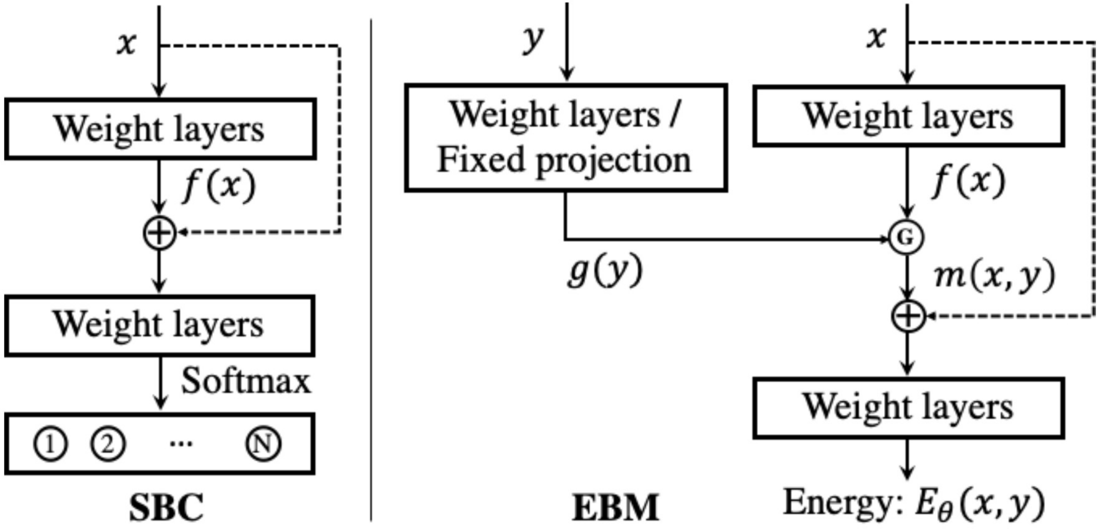


Li et al. Energy-Based Models for Continual Learning. CoLLAs 2022.

NYU

EBM for Continual Learning

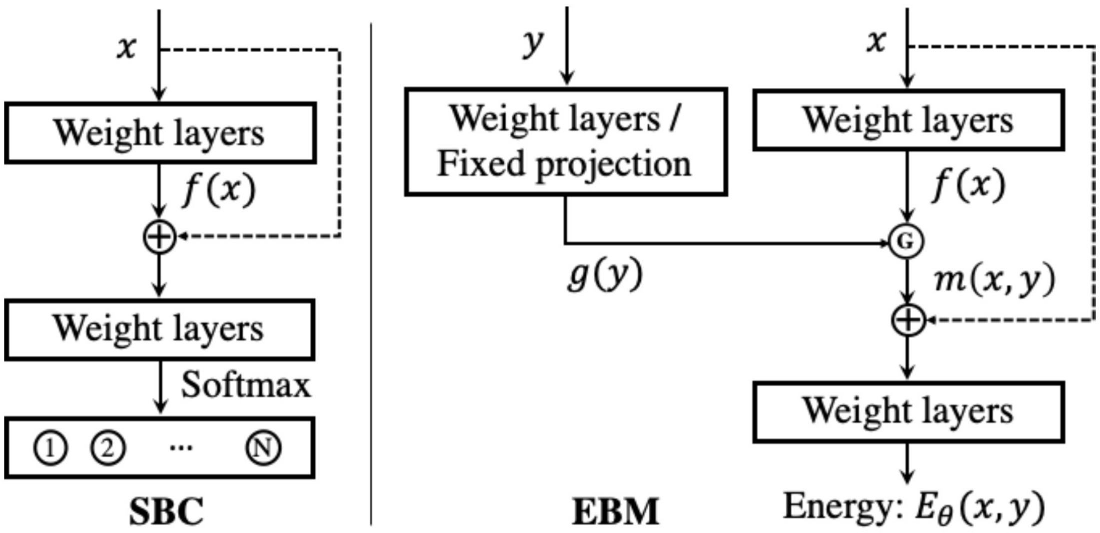
- Softmax layer is known to be sensitive to distribution shift.
- A common approach is to use nearest mean classifier.



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

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



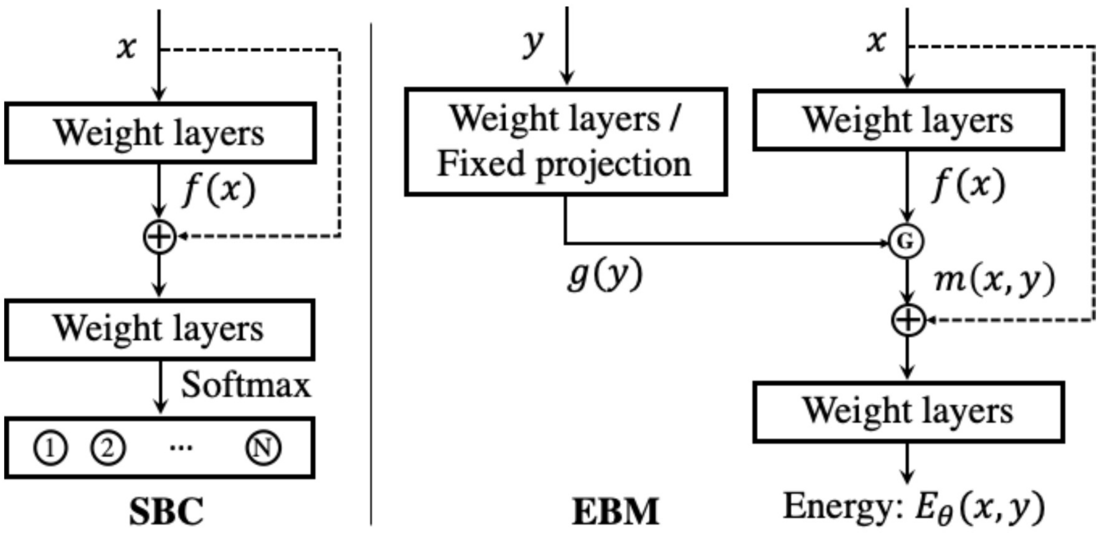
Li et al. Energy-Based Models for Continual Learning. CoLLAs 2022.

EBM for Continual Learning

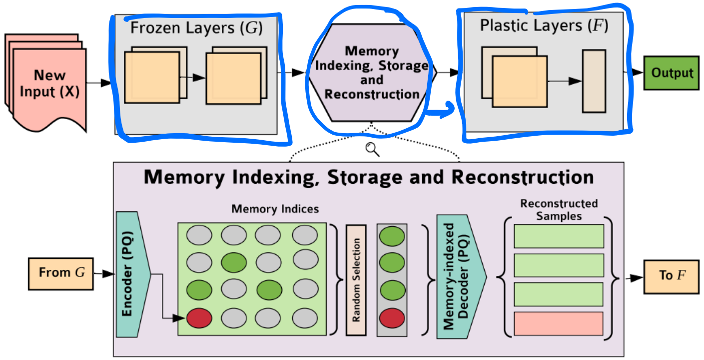
- Softmax layer is known to be sensitive to distribution shift.
- A common approach is to use nearest mean classifier.
- Can be generalized to EBMs
- Energy between inputs and labels:

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

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



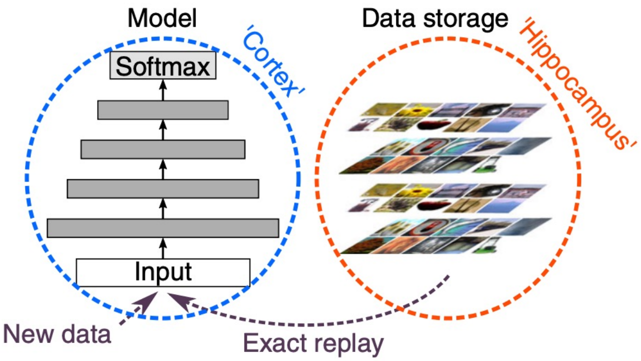
Li et al. Energy-Based Models for Continual Learning. CoLLAs 2022.

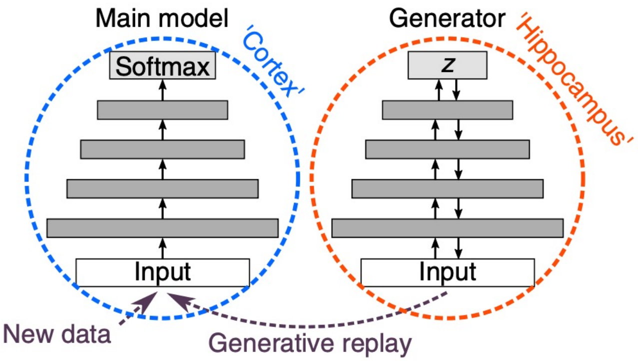


Replay

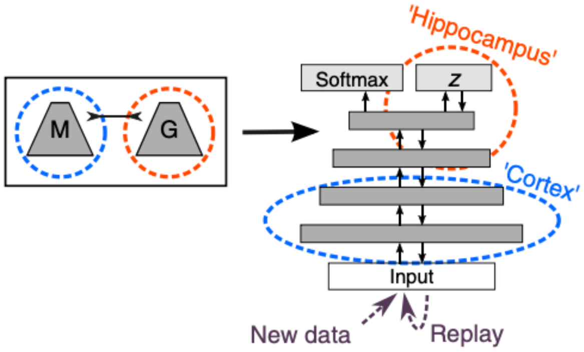
- Store raw data, representations, or train a generative model

Generative replay.





Feedback Connections



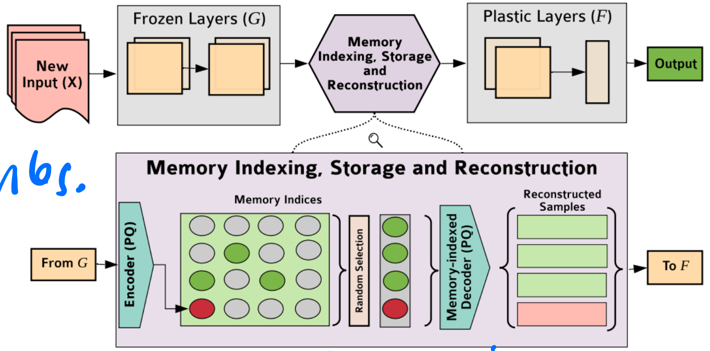
Rebuffi et al. iCaRL: Incremental Classifier and Representation Learning. CVPR 2017.
van de Ven et al. Brain-inspired replay for continual learning with artificial neural networks. Nature communications 2020.
Hayes et al. REMIND Your Neural Network to Prevent Catastrophic Forgetting. ECCV 2020.

NYU

reservoir sampling.

beginning high prob. storing samples.

Replay



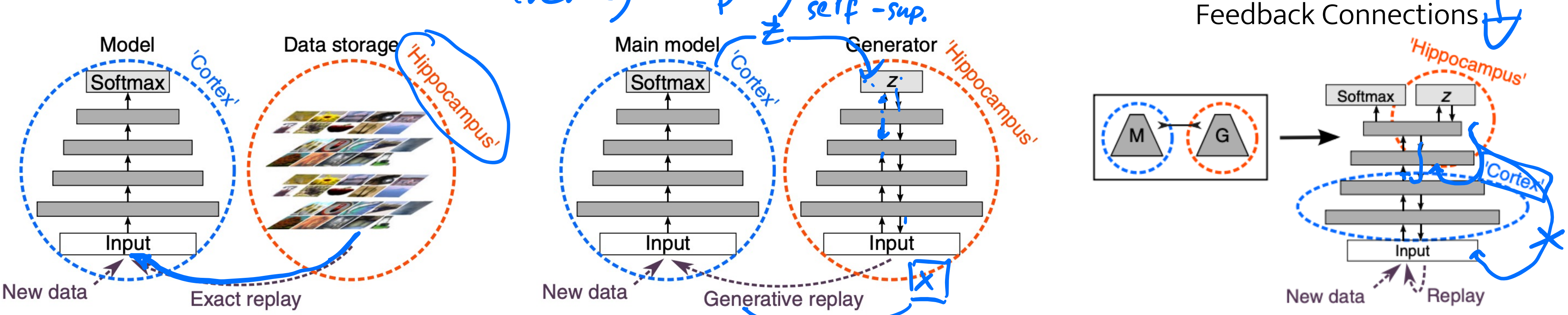
Closer to avg representation.

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

average rep

classification

self-sup.



reconstruction.

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.

train

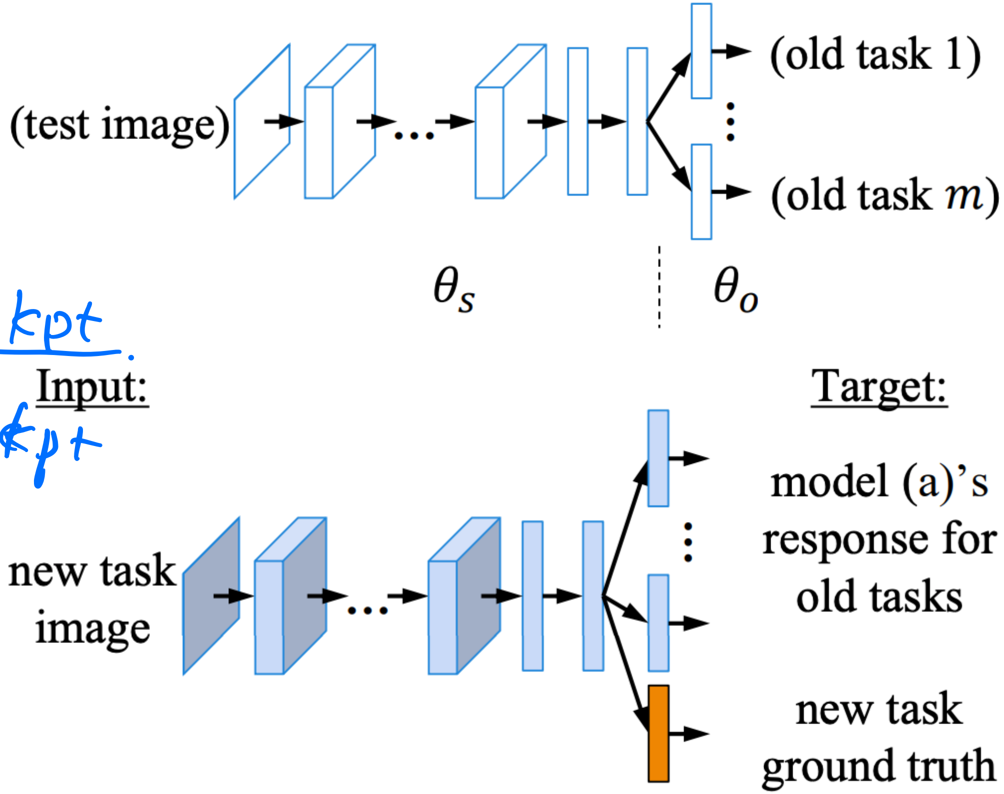
$y_o = f(x_n; \theta_o)$. old ckpt

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

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

new data

$y_n = f(x_n, \theta_n)$



Li & Hoiem. Learning without Forgetting. ECCV 2016.

Knowledge Distillation

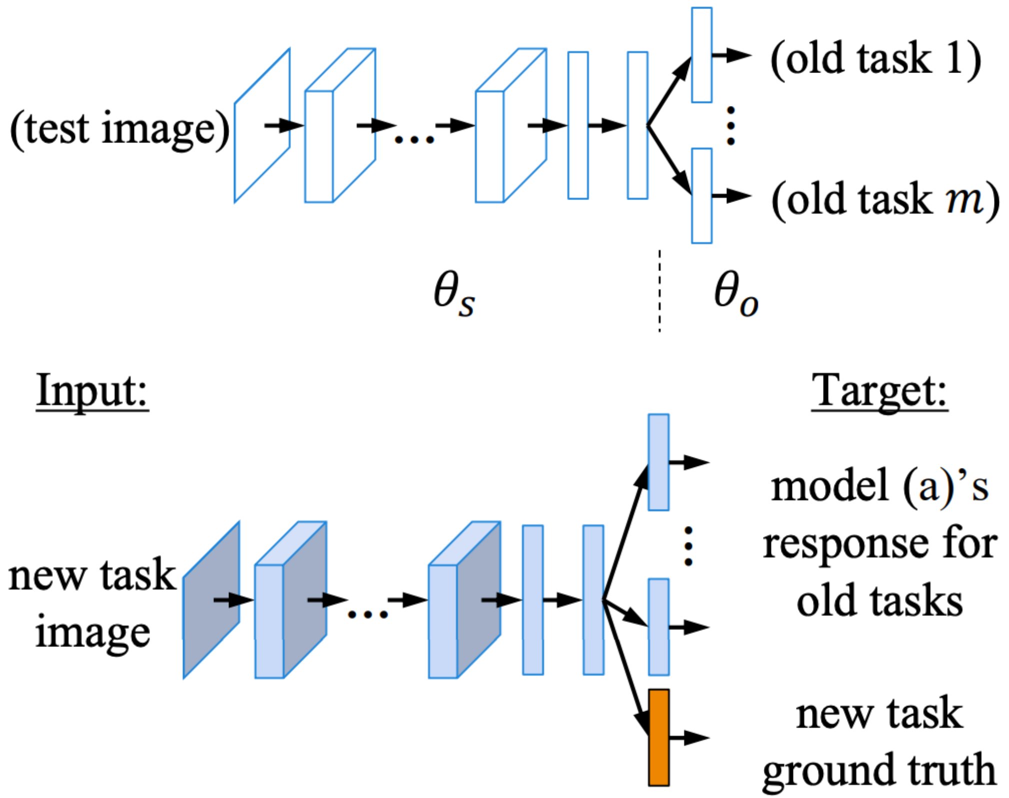
- Instead of saving the data points, we can also save the previous model checkpoint.

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

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

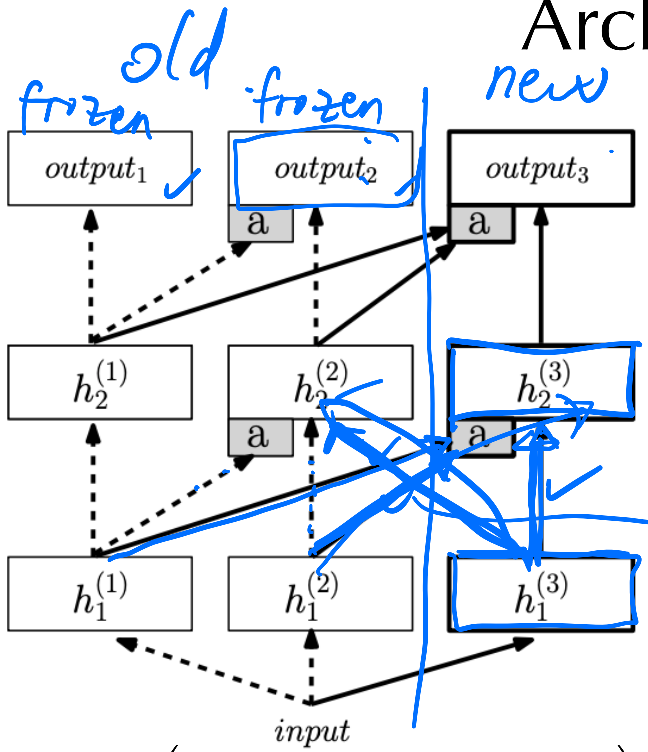
$$\mathcal{L}(y_o, \hat{y}_o)$$

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



Li & Hoiem. Learning without Forgetting. ECCV 2016.

Architecture Expansion

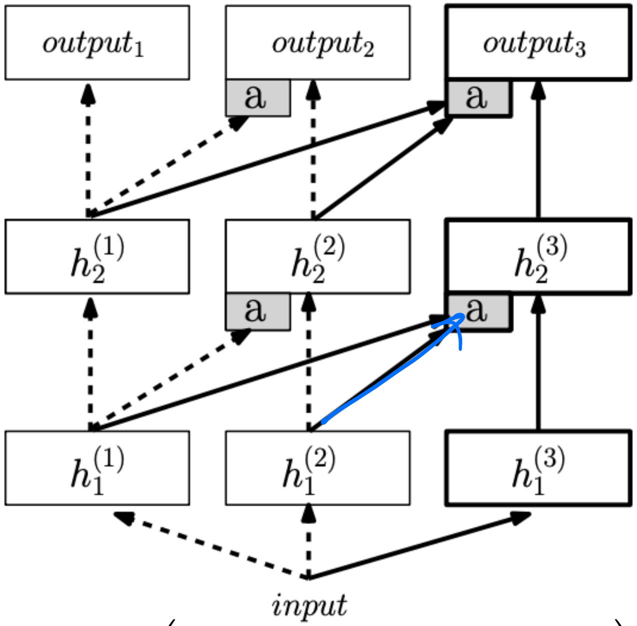


leverage knowledge from previous tasks.

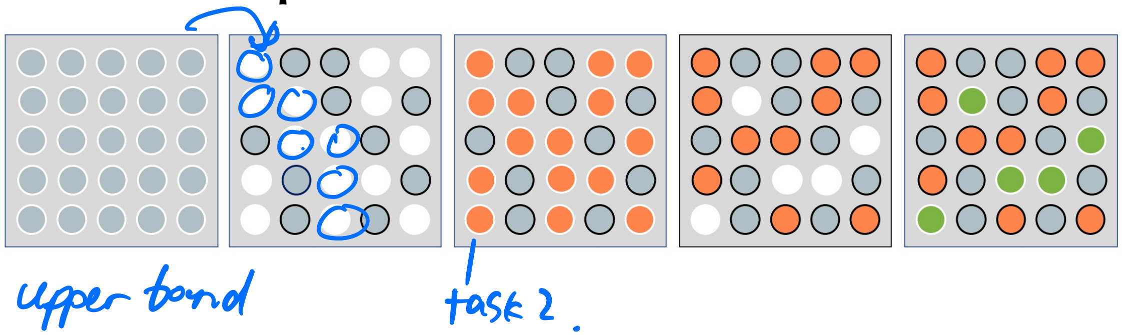
$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right).$$

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

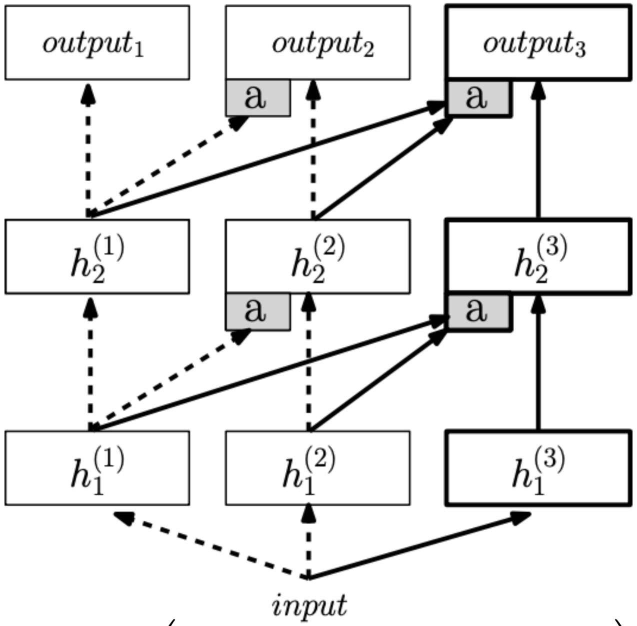


$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right).$$

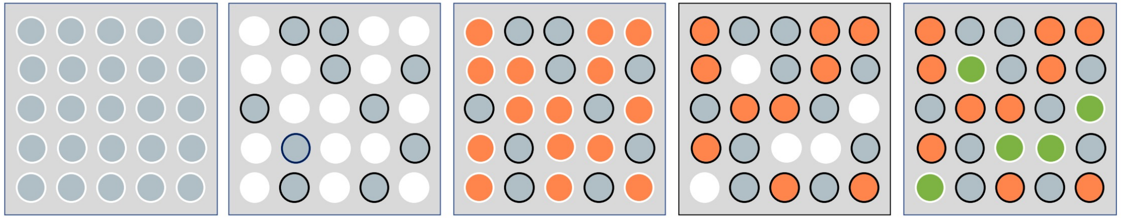


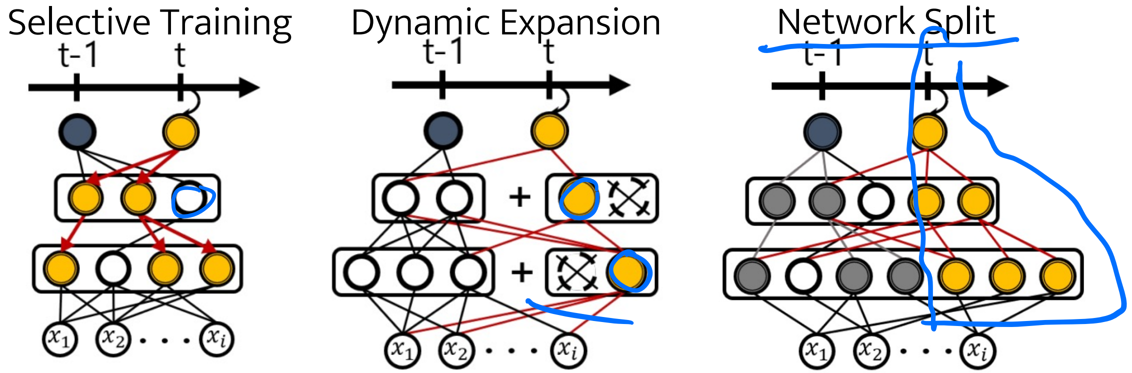
Rusu et al. Progressive Neural Networks. NIPS 2016 Deep Learning Symposium.
PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. CVPR 2018.
Yoon et al. Lifelong Learning with Dynamically Expandable Networks. ICLR 2018.

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

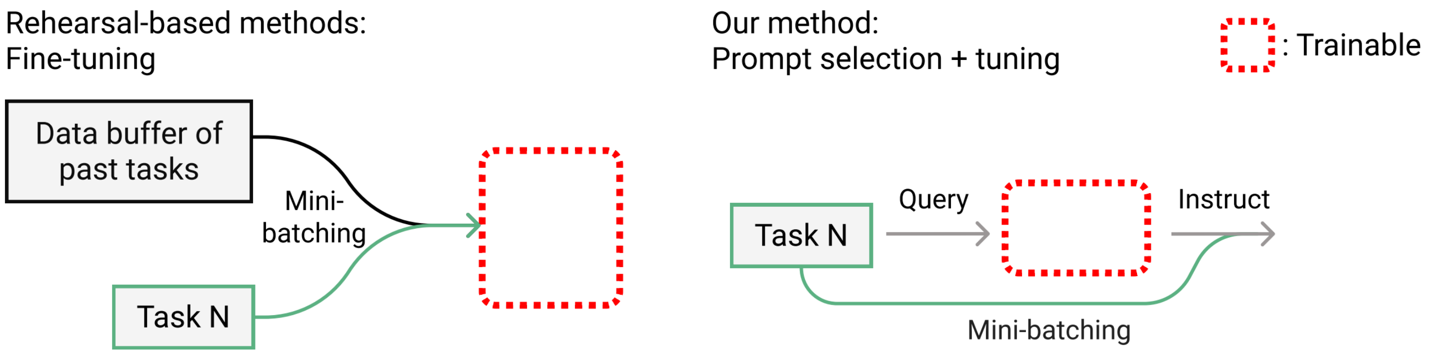
Group sparsity

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

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

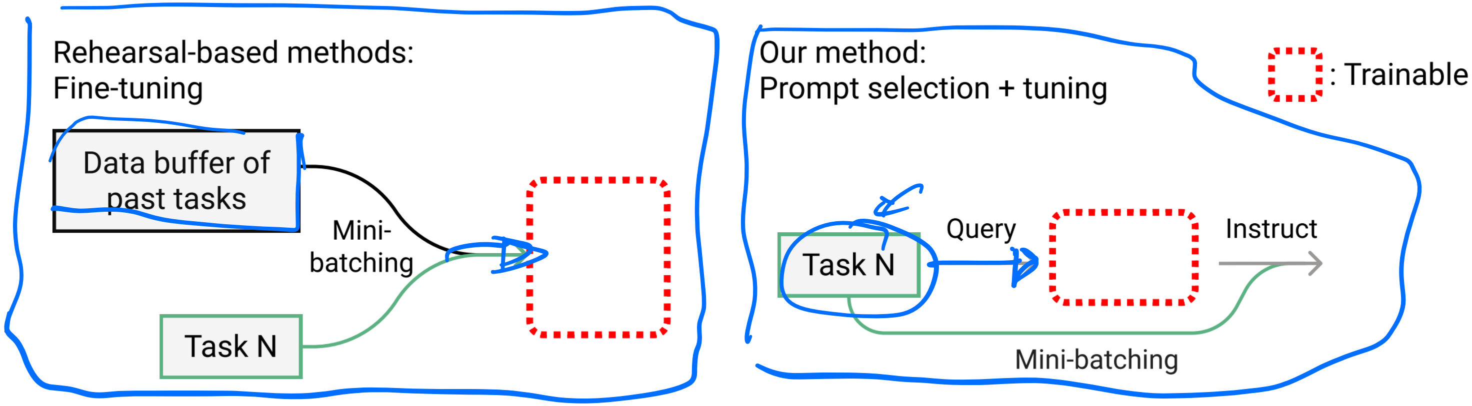
- Pretrained models have general knowledge that can be adapted to a continual stream of tasks.



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

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."

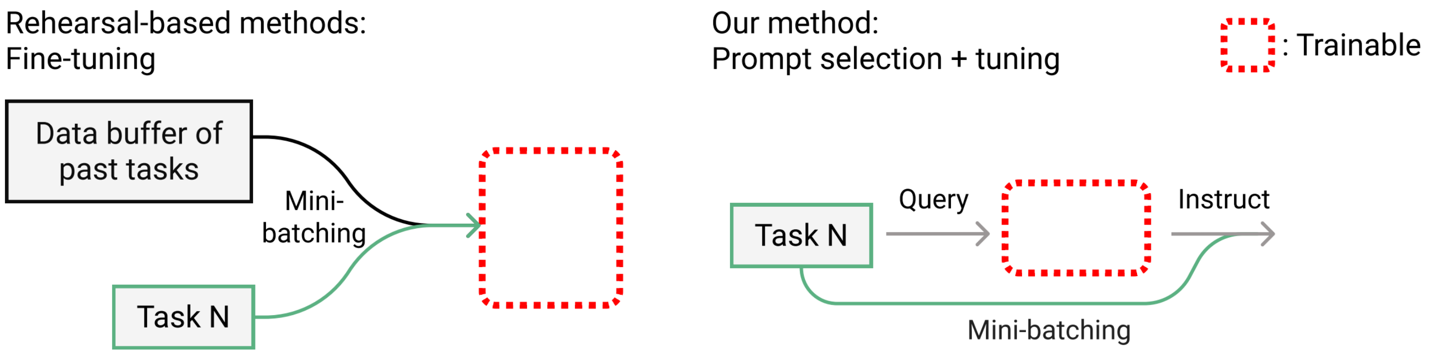


Wang et al. Learning to Prompt for Continual Learning. CVPR 2022.

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.



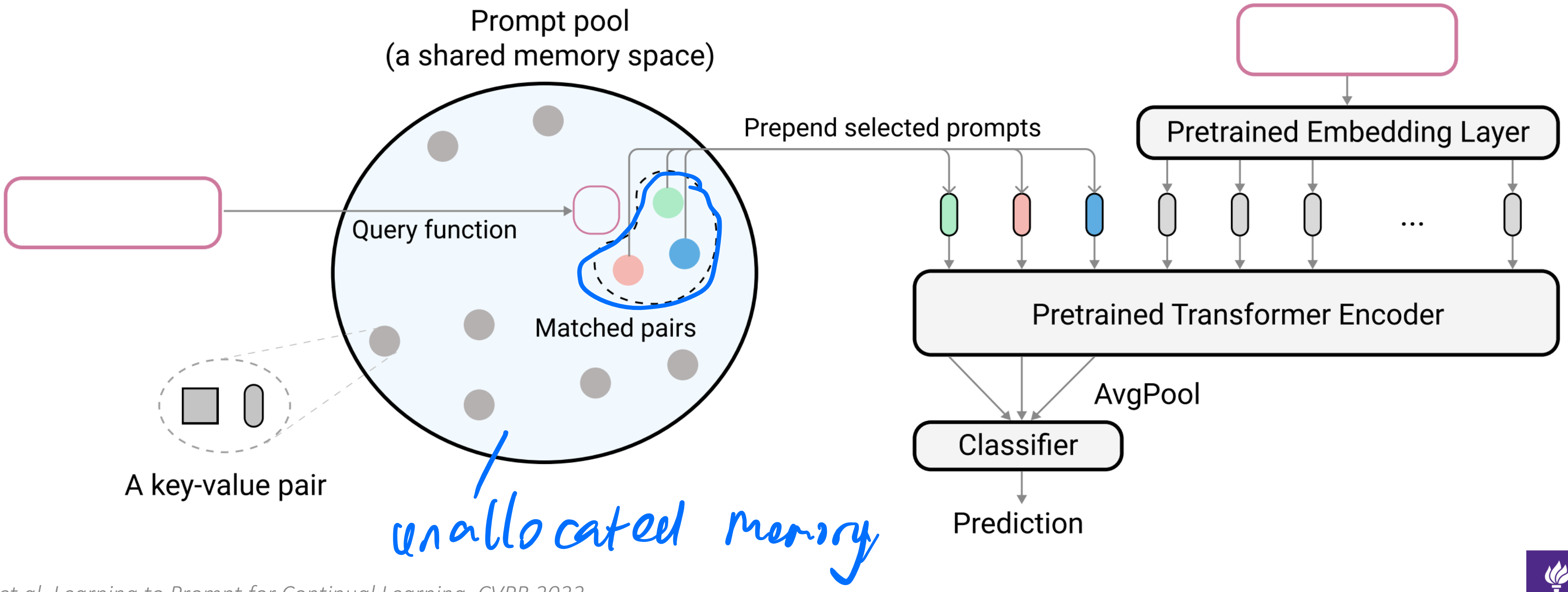
Wang et al. Learning to Prompt for Continual Learning. CVPR 2022.

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.

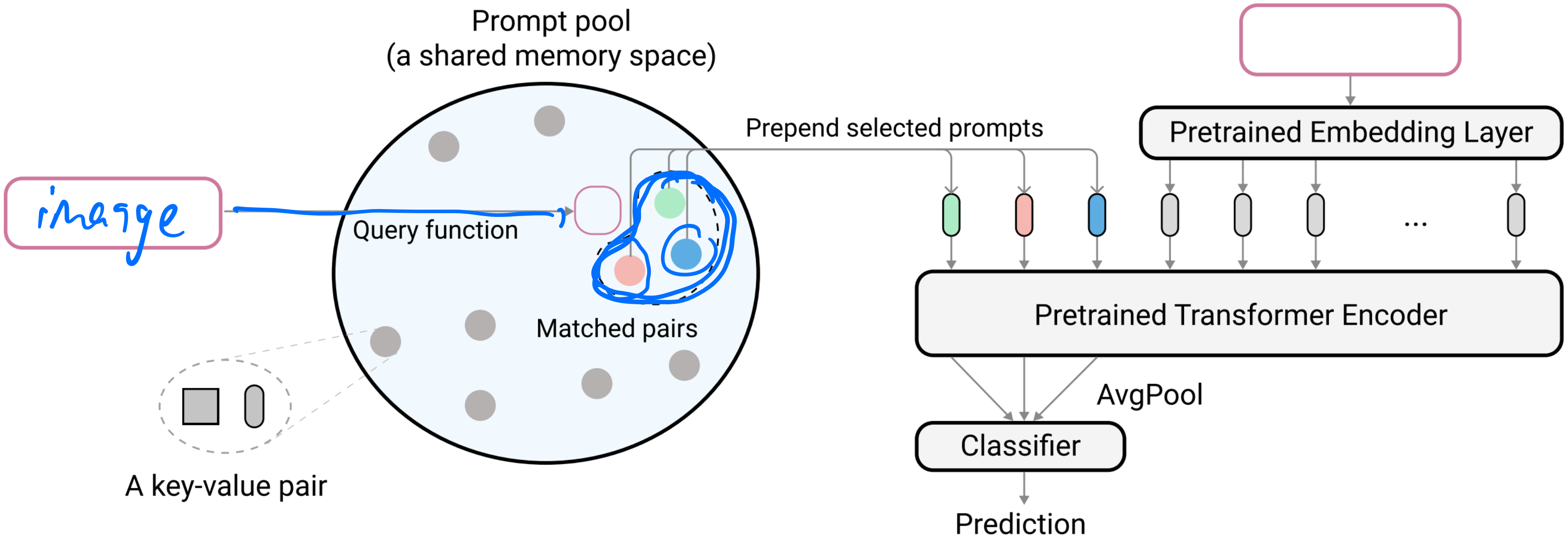
NYU

Learning to Prompt

Prompt pool (slot memory)

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

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



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

NYU

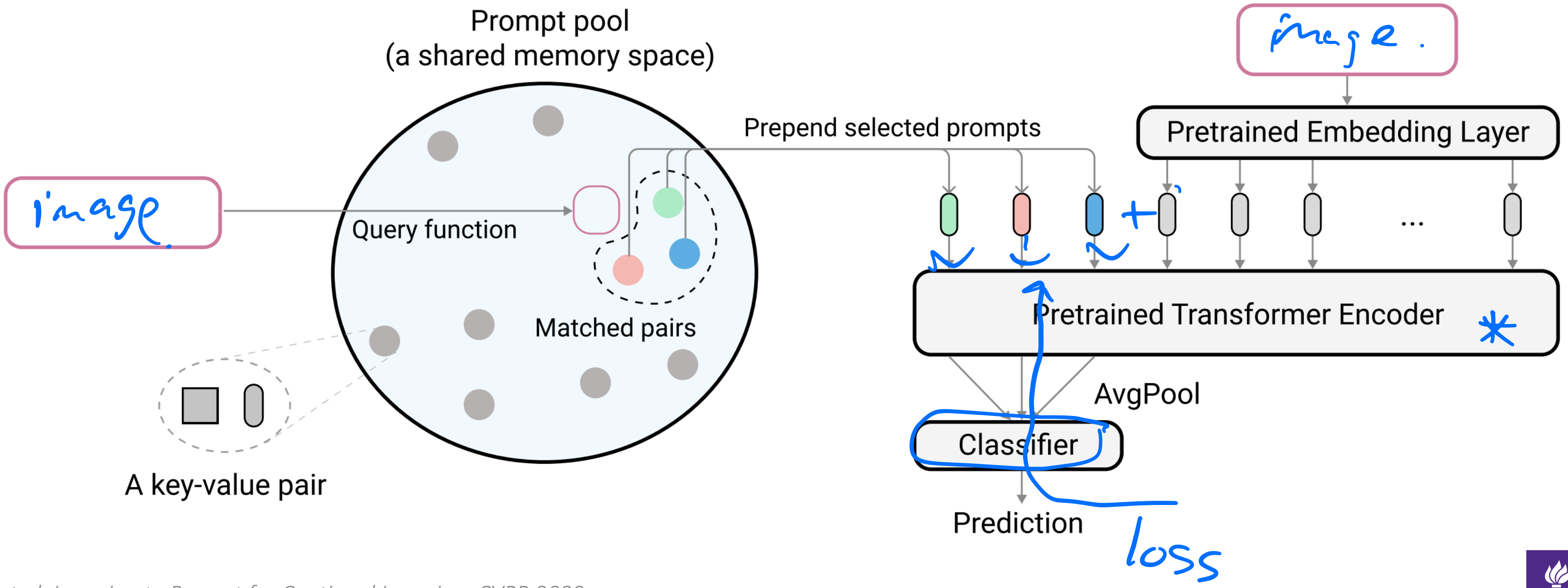
Learning to Prompt

Prompt pool (slot memory)

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

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

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

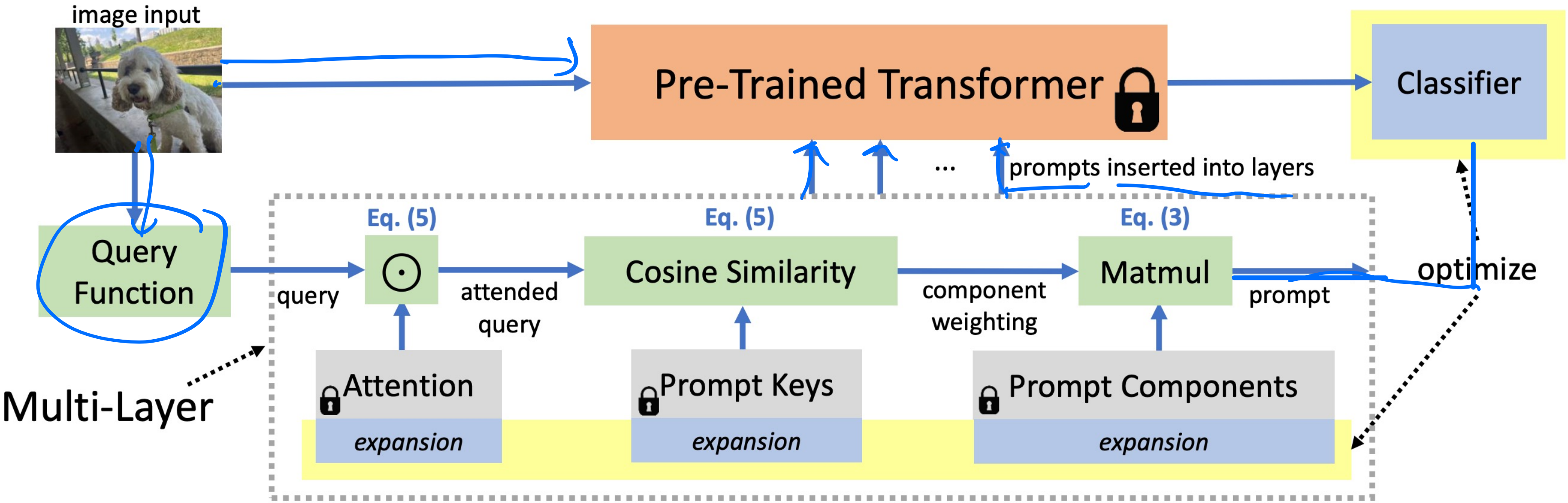


Wang et al. Learning to Prompt for Continual Learning. CVPR 2022.

Learned Prompt Query

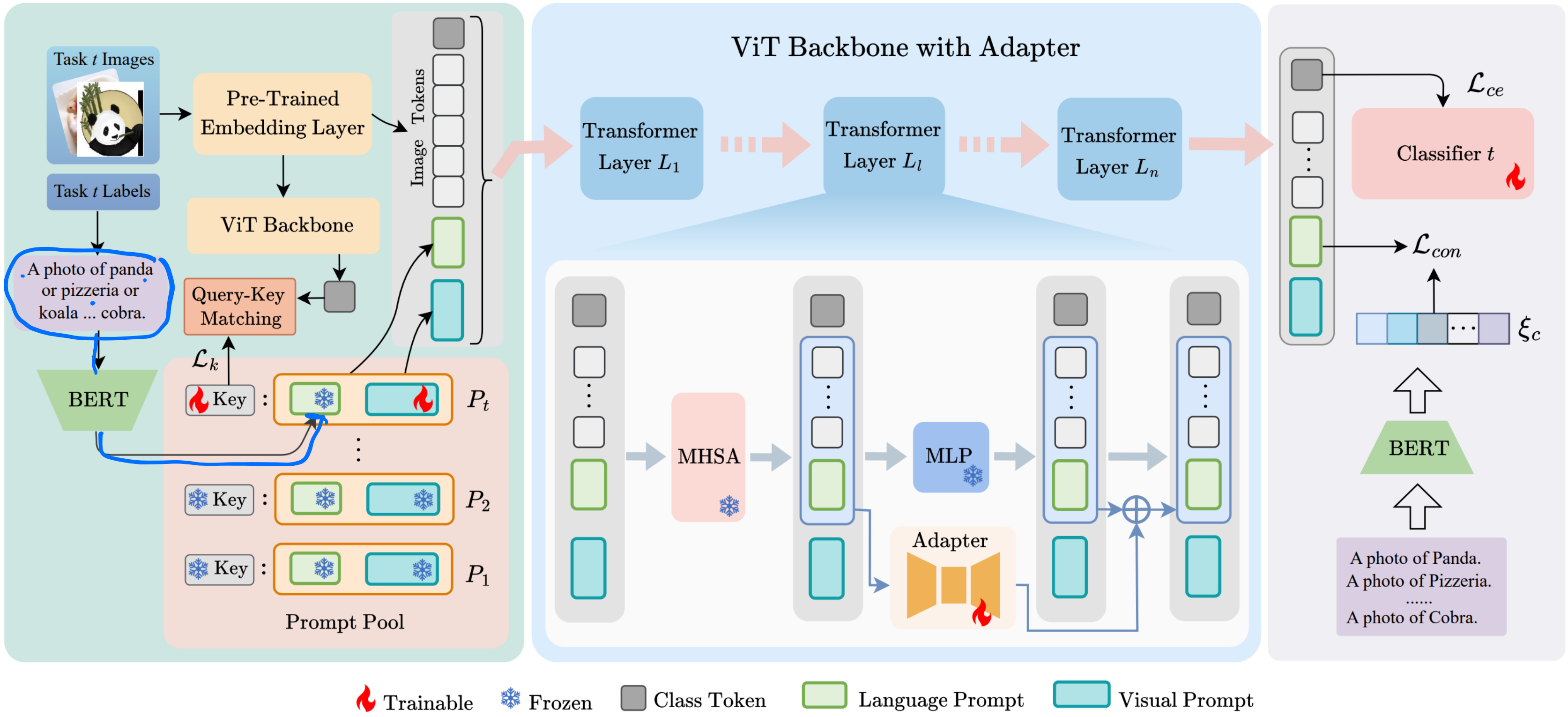
- The query function can be end-to-end learned.

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



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

Multimodal Semantic Prompts



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

Continual Learning and Memory

forgetting.

- 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

memory.

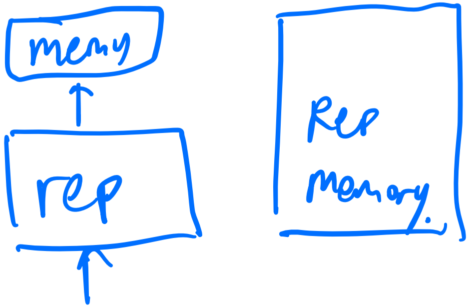
NYU

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?

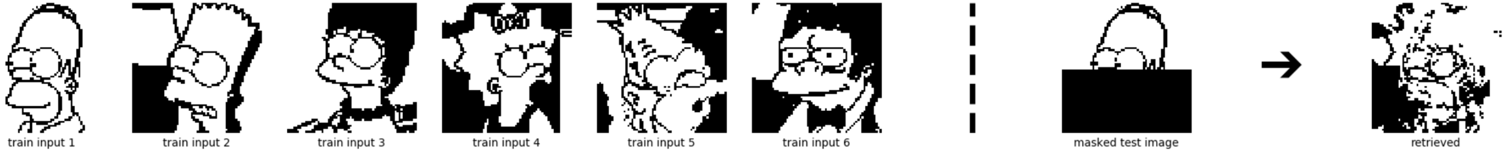
Continual Learning and Memory

- 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.”



Associative Memory

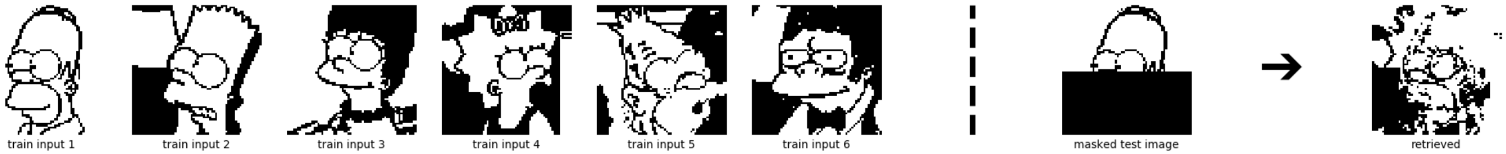
- Memory aims to store content for easy retrieval



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

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/

Associative Memory

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

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

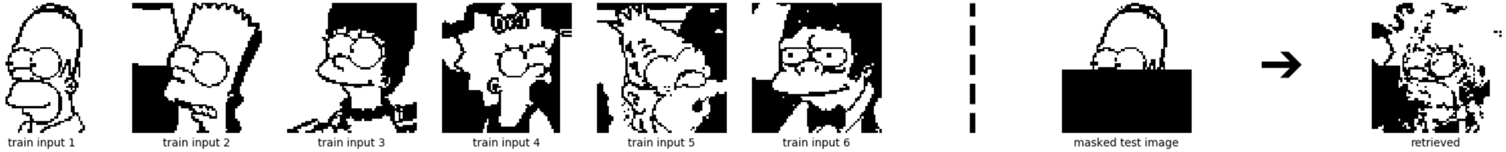
"Superposition" of k slots
Hebbian learning

$s = \xi$

stored patterns.

retrieval.

ξ stored. pattern.



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

NYU

Associative Memory

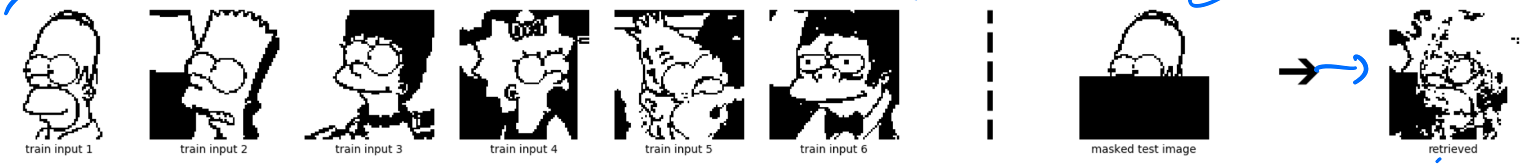
- Memory aims to store content for easy retrieval
  - Associative memories (Hopfield Networks) can be viewed as energy-based models

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

"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.

Stored patterns



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

binary.

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

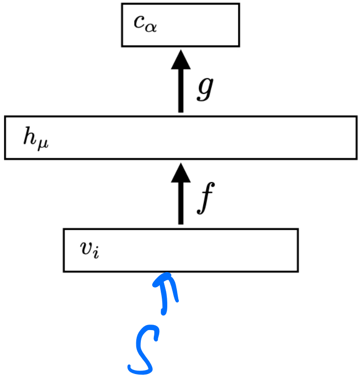
100.

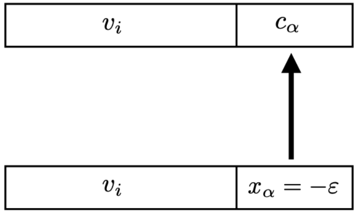
Krotov & Hopfield. *Dense Associative Memory for Pattern Recognition.* NIPS 2016.
https://ml-jku.github.io/hopfield-layers/

Memory Duality

- Duality with a feedforward network.

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





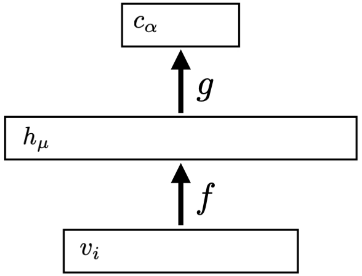
Salakhutdinov & Hinton. Deep Boltzmann Machines. AISTATS 2009.
Krotov. Hierarchical Associative Memory. arXiv 2021.

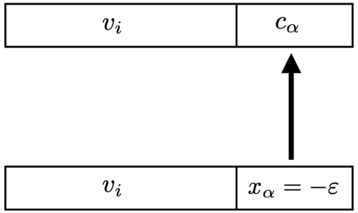
Memory Duality

- 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.





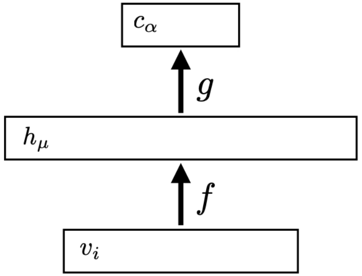
Salakhutdinov & Hinton. Deep Boltzmann Machines. AISTATS 2009.
Krotov. Hierarchical Associative Memory. arXiv 2021.

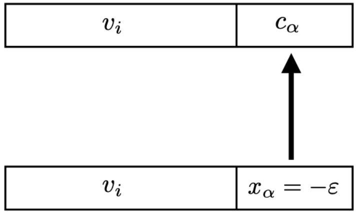
Memory Duality

- Duality with a feedforward network.

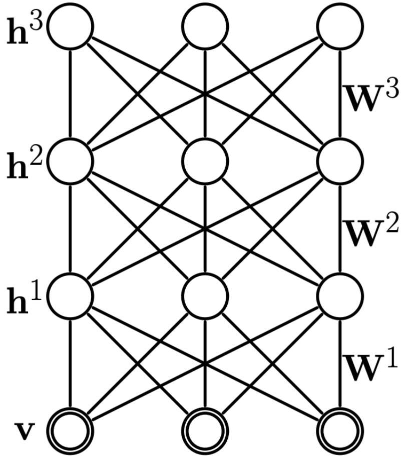
$$E = -\sum_k F\left(\sum_i \xi_i^k s_i\right)$$

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





Deep Boltzmann Machine



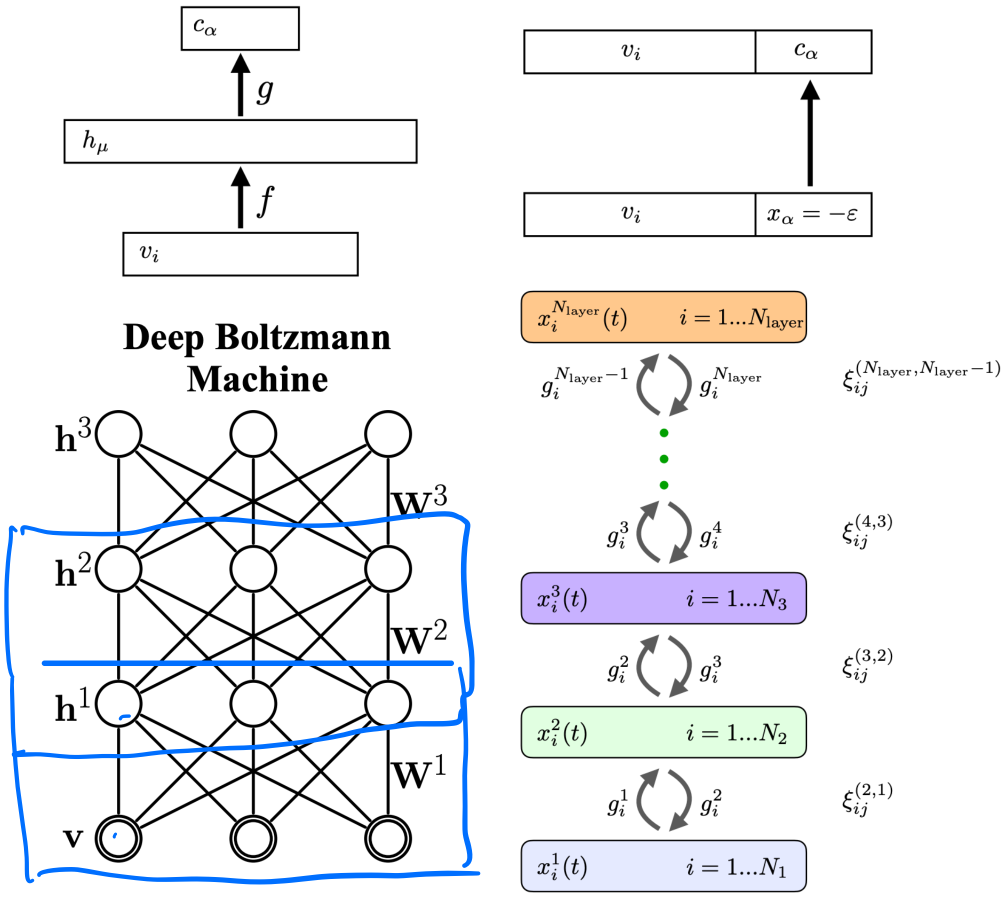
Salakhutdinov & Hinton. Deep Boltzmann Machines. AISTATS 2009.
Krotov. Hierarchical Associative Memory. arXiv 2021.

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



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

Relation to Transformers

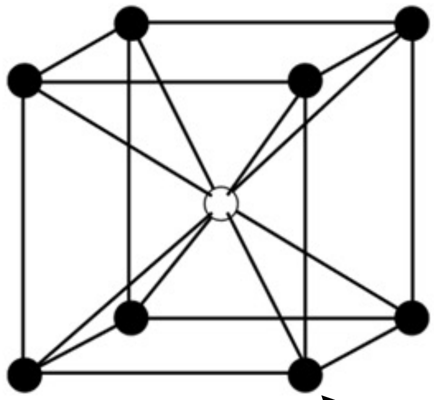
- General form:

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

Stored patterns

retrieved vector

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.

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

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

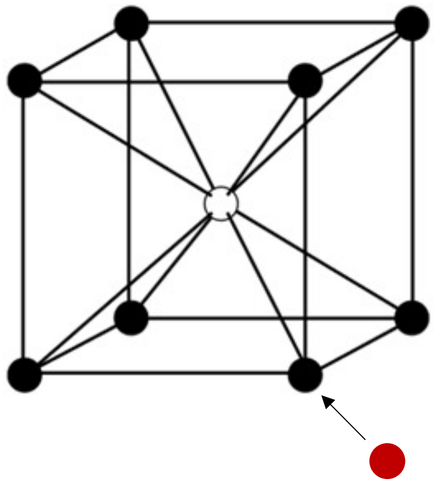
Ramsauer et al. Hopfield Networks is All You Need. ICLR 2021.

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:

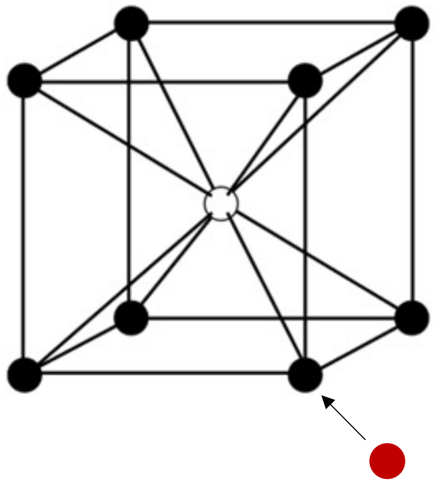


$$\nabla_{s_i} E = -\sum_j W_{ij} s_j$$

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

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:

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

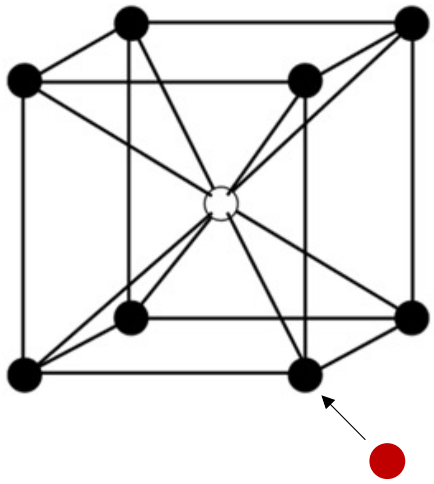
$$\nabla_{s_i} E = -\sum_j W_{ij} s_j$$

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

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

NYU

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.
- Transformer-like attention operation:

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

$$\boldsymbol{S} \leftarrow \underline{\mathrm{softmax}(\beta \boldsymbol{S}\boldsymbol{\Xi}^\top)\boldsymbol{\Xi}}. \quad \to \nabla E$$

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

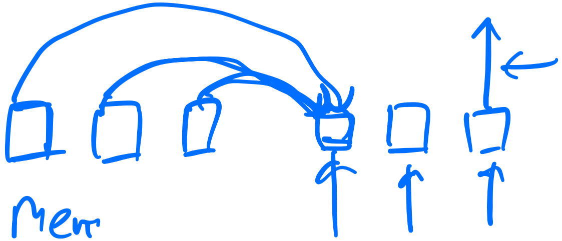
$$s_i \leftarrow \underline{\mathrm{sign}(\textstyle\sum_j W_{ij} s_j)}$$

$\xi\xi^\top$

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

NYU

Relation to Transformers



- General form:

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

Mem

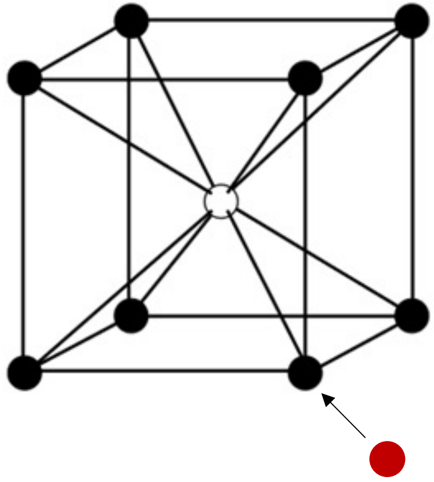
- When $F(z) = z^2$ it gives the classic HN.
- Transformer-like attention operation:

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

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

1 step.

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



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

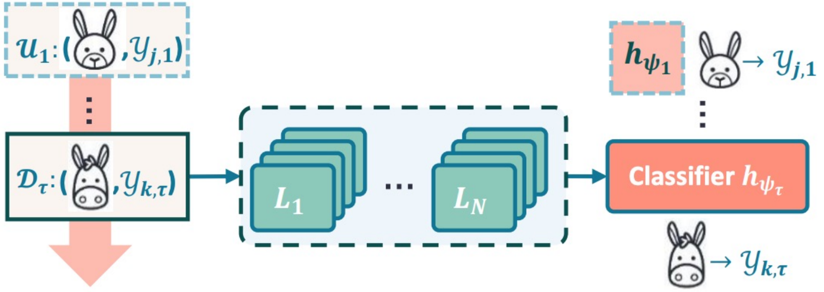
$$s_i \leftarrow \mathrm{sign}\big(\textstyle\sum_j W_{ij} s_j\big)$$

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

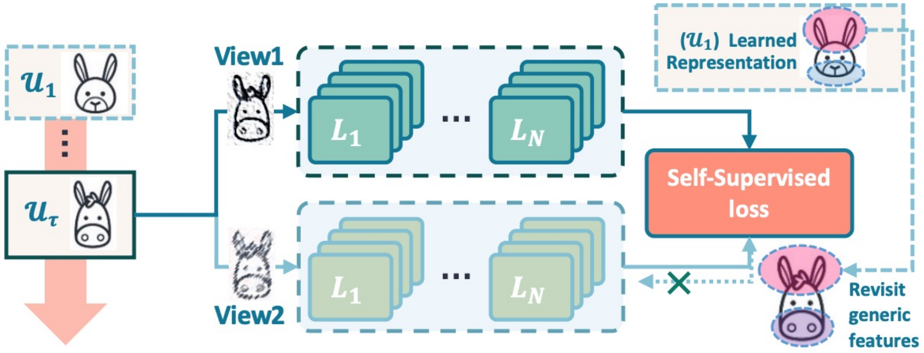
NYU

Continual Self-Supervised Learning

- Learning from a stream of unlabeled inputs.



SUPERVISED CONTINUAL LEARNING (SCL)

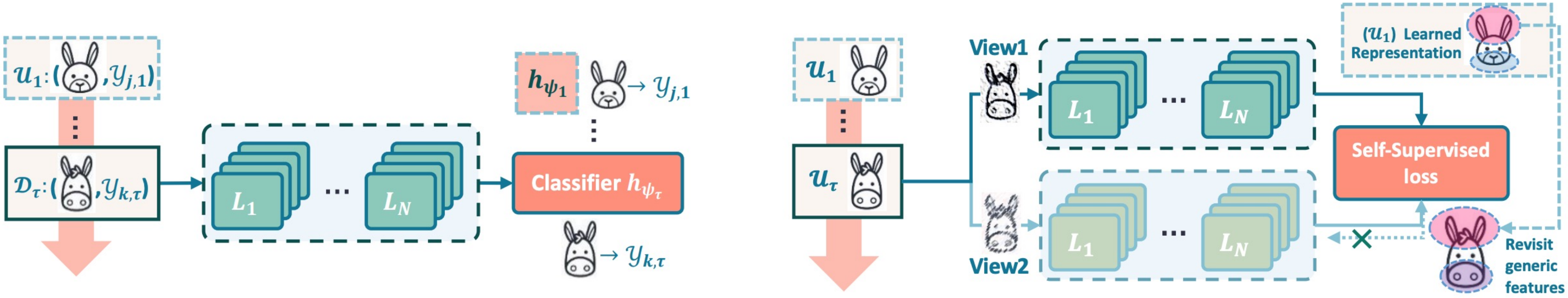


UNSUPERVISED CONTINUAL LEARNING (UCL)

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

Continual Self-Supervised Learning

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



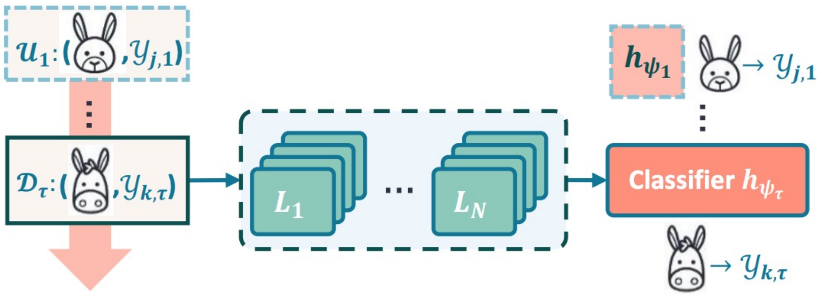
SUPERVISED CONTINUAL LEARNING (SCL)

UNSUPERVISED CONTINUAL LEARNING (UCL)

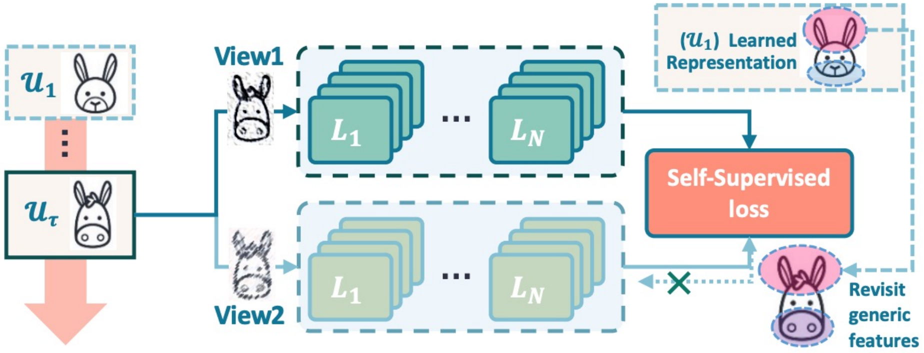
Madaan et al. Representational Continuity for Unsupervised Continual Learning. ICLR 2022.

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)

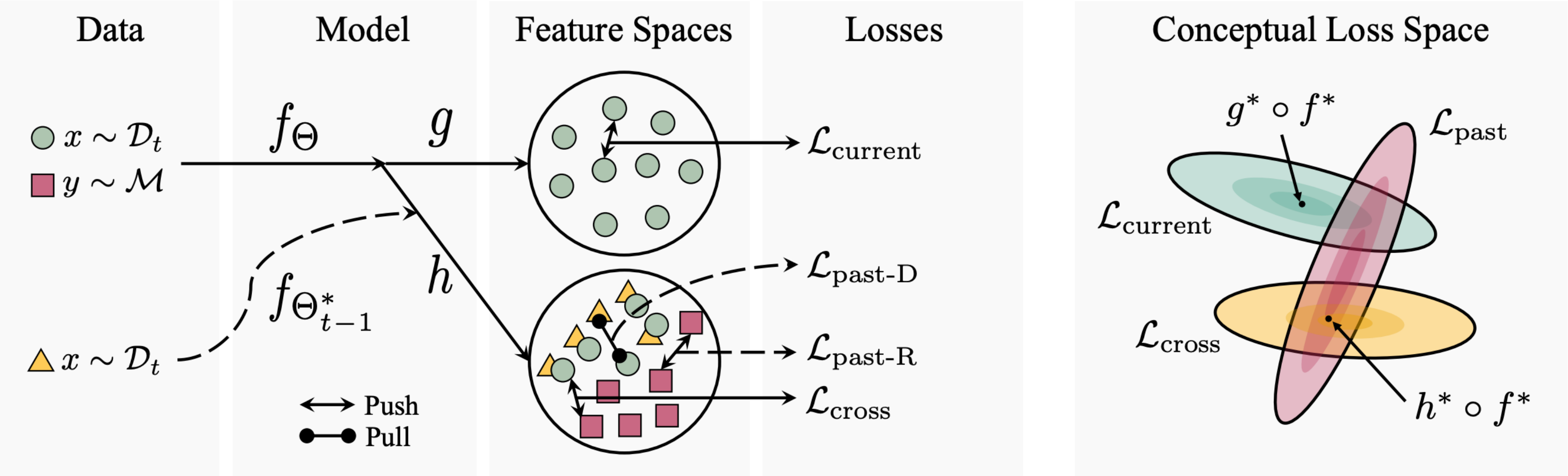
Madaan et al. Representational Continuity for Unsupervised Continual Learning. ICLR 2022.

NYU

Continual Self-Supervised Learning

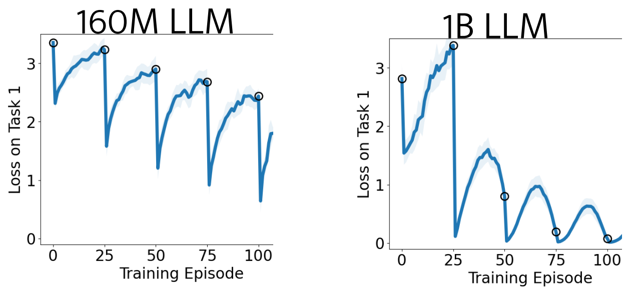
- Integrating learning objectives of both past and present.

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



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

Outlooks

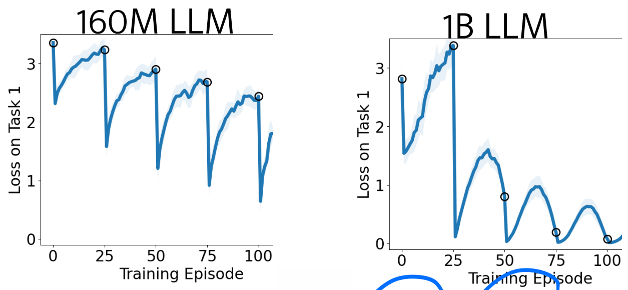


- Understand continual learning at scale

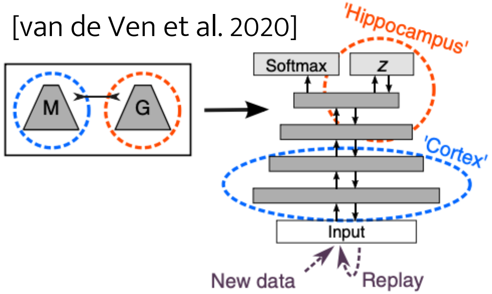
[Mayo et al. 2023]

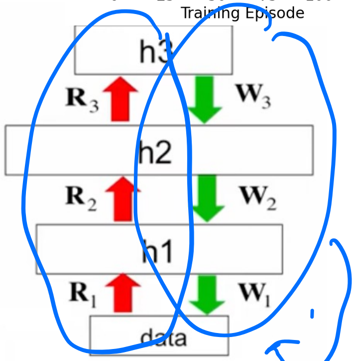
Outlooks

- Understand continual learning at scale
- Unified learning architecture, objective and replay, role of sleep



[van de Ven et al. 2020]





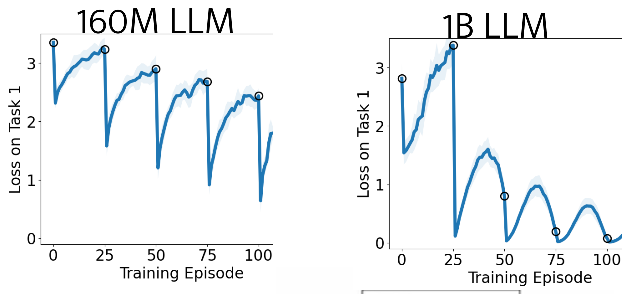
[Hinton et al. 1995]

[Mayo et al. 2023]

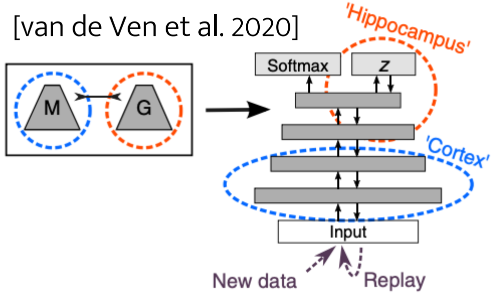
NYU

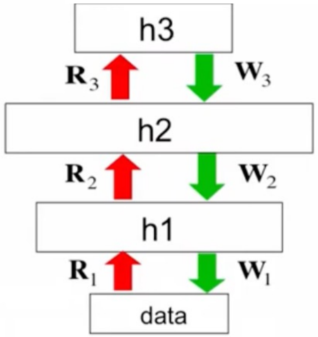
Outlooks

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

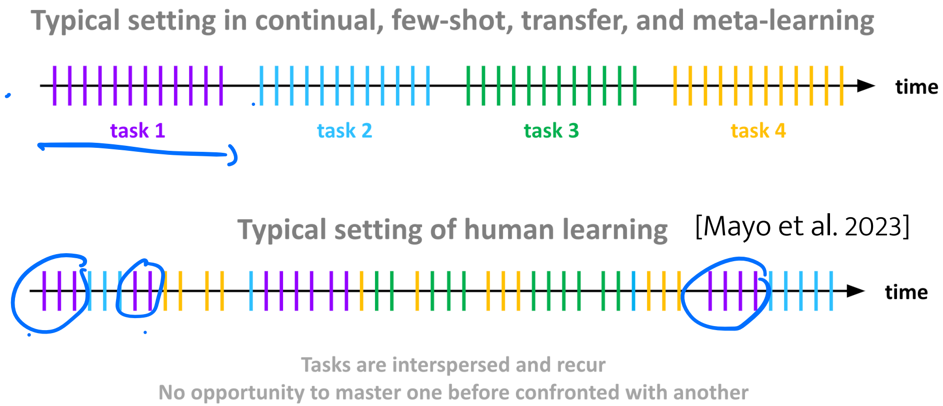


[van de Ven et al. 2020]





[Hinton et al. 1995]



[Mayo et al. 2023]

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

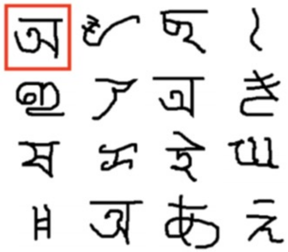
Summary: Continual Learning

- 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

Few-Shot Learning and Meta-Learning

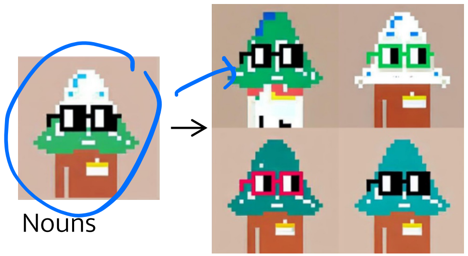
Few-Shot Learning (FSL)

- Humans can quickly learn new concepts with a few examples.
- Learning in embodied agents also needs to be adaptive and swift.
- Examples: Recognize new objects, perform new skills, map new areas, etc.



[Lake et al. 2011]

[Ren et al. 2018]



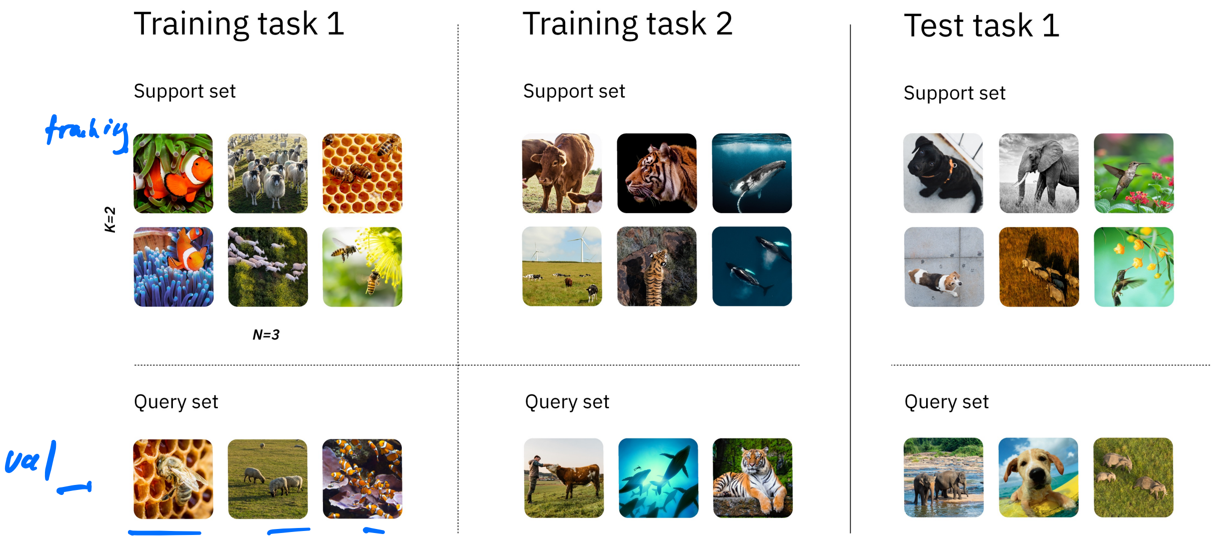
[Lu et al. 2024]

FSL: General Setup

- Quickly learn a task (learning episode) with very few number of training examples and get evaluated on a set of test examples.

FSL: General Setup

- Quickly learn a task (learning episode) with very few number of training examples and get evaluated on a set of test examples.
- During training, going through many episodes of the same structure.

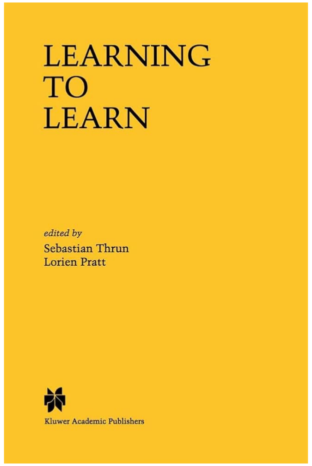


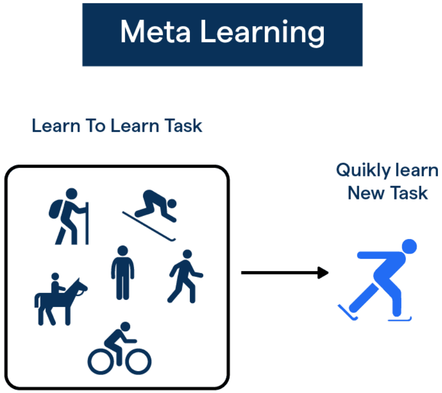
https://www.ibm.com/think/topics/few-shot-learning

NYU

Learning to Learn

- Conceptually, we'd like to generalize new learning experiences.

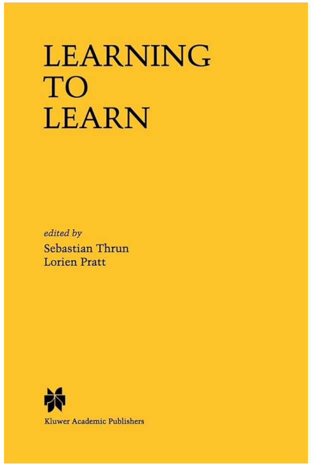


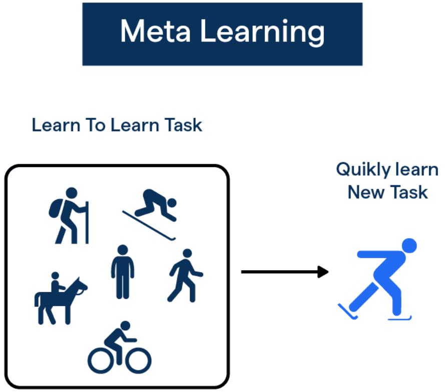


NYU

Learning to Learn

- Conceptually, we'd like to generalize new learning experiences.
- 1 learning experience = 1 training example

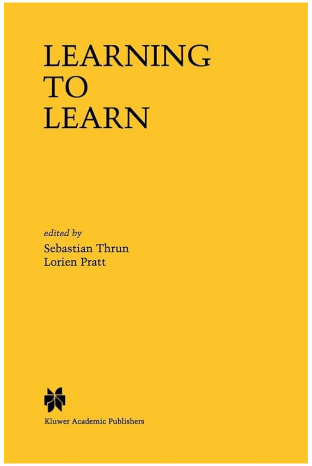


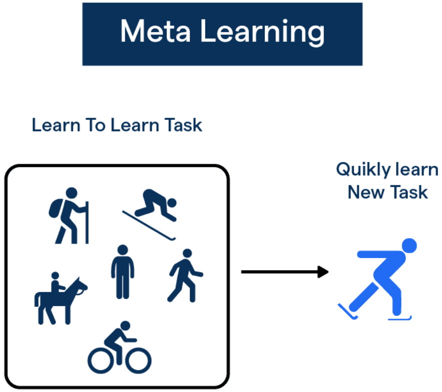


T_1 T_2 . . . T_N

Learning to Learn

- Conceptually, we'd like to generalize new learning experiences.
- 1 learning experience = 1 training example
- Related to multi-task learning

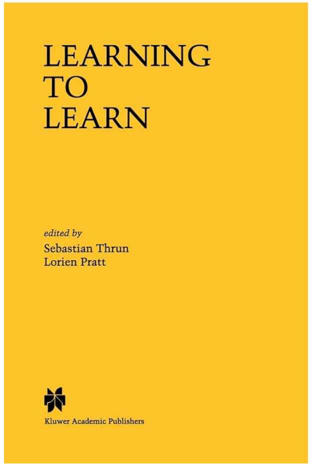


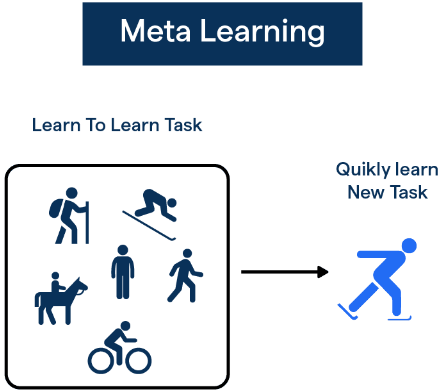


NYU

Learning to Learn

- Conceptually, we'd like to generalize new learning experiences.
- 1 learning experience = 1 training example
- Related to multi-task learning
- What can be meta-learned?
    - Optimizer
    - Initialization
    - Architecture
    - Representations
    - Abstraction of tasks

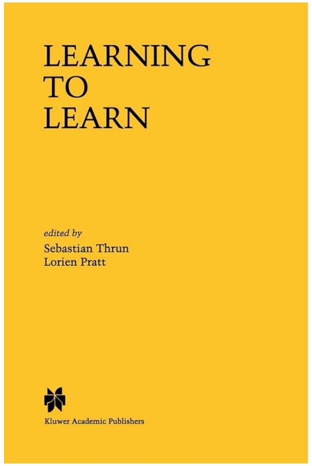


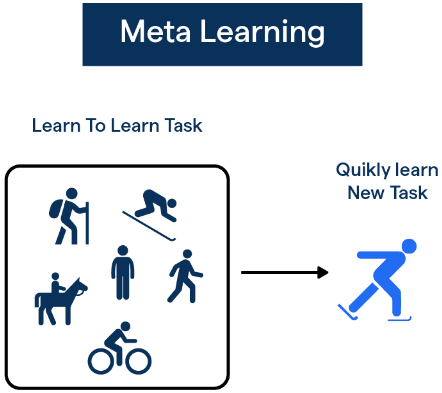


NYU

Learning to Learn

- Conceptually, we'd like to generalize new learning experiences.
- 1 learning experience = 1 training example
- Related to multi-task learning
- What can be meta-learned?
    - Optimizer
    - Initialization
    - Architecture
    - Representations
    - Abstraction of tasks
- Discover learning algorithms that support FSL.





NYU

Meta-Optimization (Bi-level Optimization)

- One can formulate meta-learning as a meta-optimization problem.

$$\min_{\lambda} \mathop{\mathbb{E}}_{D \sim \mathcal{D}} \boxed{\min_{\theta} \mathop{\mathbb{E}}_{x \sim D} \mathcal{L}(x; \theta, \lambda).}$$

hyperparameter

Meta-Optimization (Bi-level Optimization)

- One can formulate meta-learning as a meta-optimization problem.

$$\min_{\lambda} \mathop{\mathbb{E}}_{D \sim \mathcal{D}} \min_{\theta} \mathop{\mathbb{E}}_{x \sim D} \mathcal{L}(x; \theta, \lambda).$$

- Need to optimize through the inner optimization.
  - BPTT
  - Fixed point (implicit differentiation)
  - Zeroth order optimization

Meta-Optimization (Bi-level Optimization)

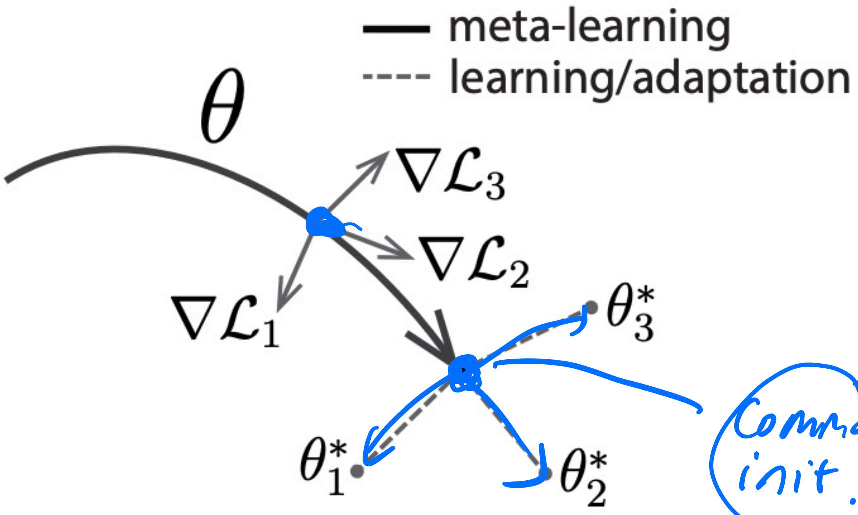
- One can formulate meta-learning as a meta-optimization problem.

$$\min_{\lambda} \mathop{\mathbb{E}}_{D\sim\mathcal{D}} \min_{\theta} \mathop{\mathbb{E}}_{x\sim D} \mathcal{L}(x;\theta,\lambda).$$

- Need to optimize through the inner optimization.
  - BPTT
  - Fixed point (implicit differentiation)
  - Zeroth order optimization
- Short-horizon bias: Optimal actions in the next few steps may not be optimal in the long run.
  - Example: Lowering learning rate will always result in short-term gains.

MAML (Truncated Optimization)

- For few-shot learning, short horizon is actually needed.
- Unroll the gradient graph for a few iterations.
- MAML (Model-agnostic meta-learning)



Common init.

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

Finn et al. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017.

Hypernetworks, Best Response Functions

- Another way to "amortize" the optimization is to use a network to predict the optimal inner parameters.

MacKay et al. Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions. ICLR 2019.

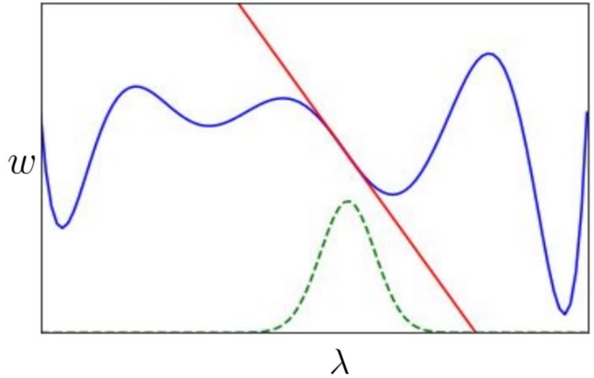
Hypernetworks, Best Response Functions

- Another way to "amortize" the optimization is to use a network to predict the optimal inner parameters.
- Best response function: $\boxed{w^*} = \operatorname{argmin}_w \mathcal{L}_T(\boxed{\lambda}, w)$. $\lambda^* = \operatorname{argmin}_\lambda \mathcal{L}(\lambda, w^*)$.

NYU

Hypernetworks, Best Response Functions

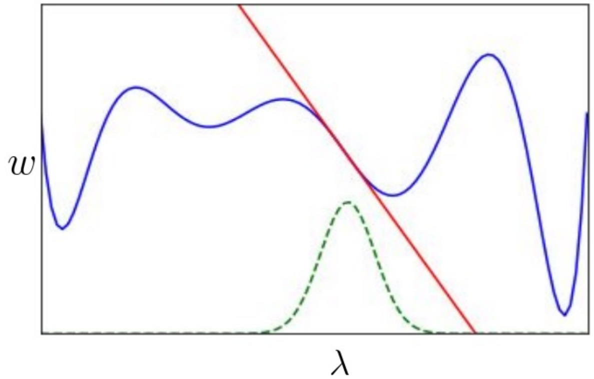
- Another way to "amortize" the optimization is to use a network to predict the optimal inner parameters.
- Best response function: $w^* = \text{argmin}_w \mathcal{L}_T(\lambda, w)$. $\lambda^* = \text{argmin}_\lambda \mathcal{L}(\lambda, w^*)$.
- Approximation: $\lambda^* \approx \text{argmin}_\lambda \mathcal{L}(\lambda, \hat{w}_\phi(\lambda))$. $\min_\phi \mathbb{E}_\epsilon[f(\lambda + \epsilon, \hat{w}(\lambda + \epsilon))]$.



MacKay et al. Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions. ICLR 2019.

Hypernetworks, Best Response Functions

- Another way to "amortize" the optimization is to use a network to predict the optimal inner parameters.
- Best response function: $w^* = \operatorname{argmin}_w \mathcal{L}_T(\lambda, w)$. $\lambda^* = \operatorname{argmin}_\lambda \mathcal{L}(\lambda, w^*)$.
- Approximation: $\lambda^* \approx \operatorname{argmin}_\lambda \mathcal{L}(\lambda, \hat{w}_\phi(\lambda))$. $\min_\phi \mathbb{E}_\epsilon[f(\lambda+\epsilon, \hat{w}(\lambda+\epsilon))]$.
- Scalable version? Instead of predicting full parameters?



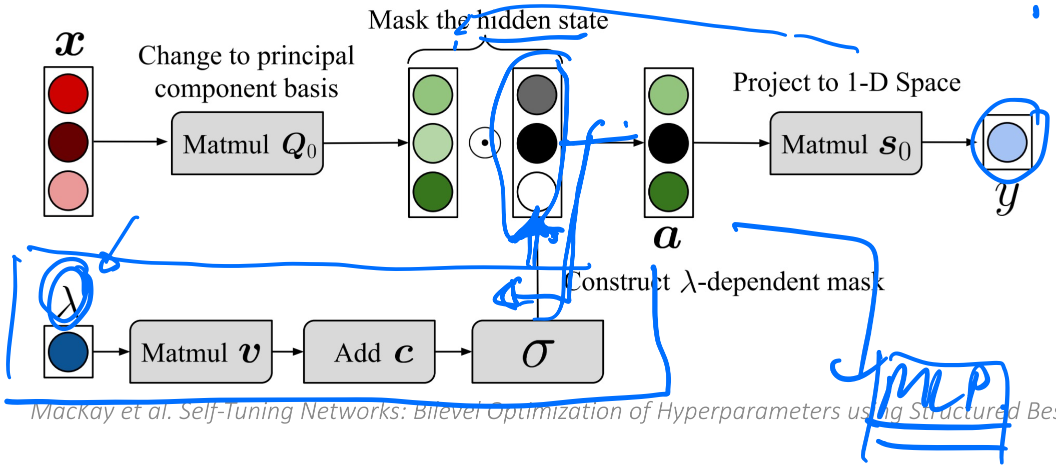
MacKay et al. Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions. ICLR 2019.

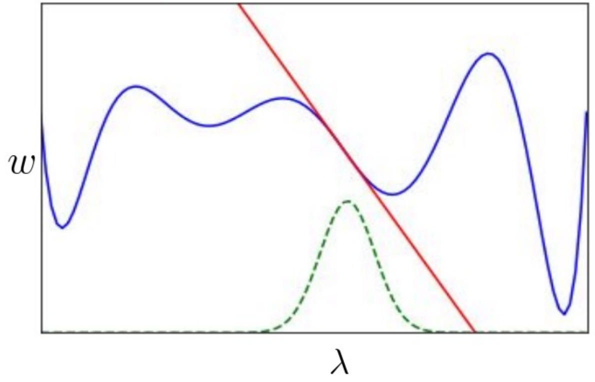
NYU

approx. inner loop.

Hypernetworks, Best Response Functions

- Another way to "amortize" the optimization is to use a network to predict the optimal inner parameters.
- Best response function: $w^* = \text{argmin}_w \mathcal{L}_T(\lambda, w)$. $\lambda^* = \text{argmin}_\lambda \mathcal{L}(\lambda, w^*)$.
- Approximation: $\lambda^* \approx \text{argmin}_\lambda \mathcal{L}(\lambda, \hat{w}_\phi(\lambda))$. $\min_\phi \mathbb{E}_\epsilon[f(\lambda + \epsilon, \hat{w}(\lambda + \epsilon))]$.
- Scalable version? Instead of predicting full parameters?





MacKay et al. Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions. ICLR 2019.

Representation and Memory

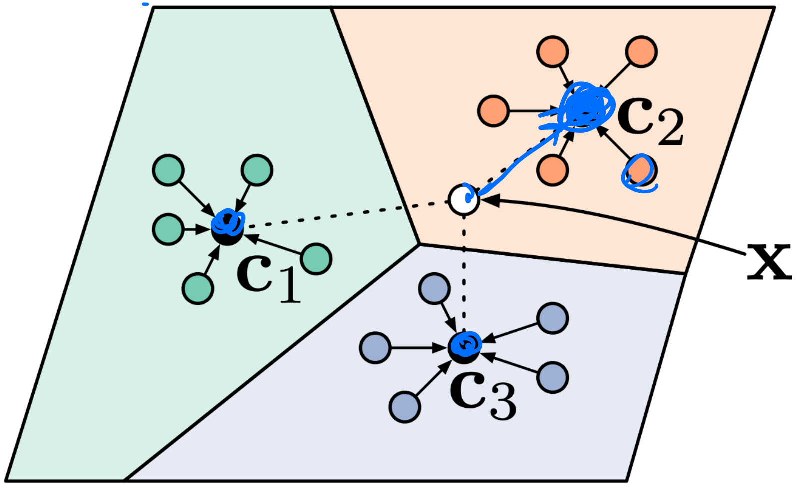
- Prototypical Network: Few-shot Classification

Snell et al. Prototypical Networks for Few-shot Learning. NIPS 2017.
Allen et al. Infinite Mixture Prototypes for Few-Shot Learning. ICML 2019.

Representation and Memory

- Prototypical Network: Few-shot Classification
- Prototype = Avg. representation of a class

$$\mathbf{p}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x},y)\in S_k} f_\phi(\mathbf{x}_i).$$



$$p(y = k \mid \mathbf{x}) = \mathrm{softmax}(-d(f_\phi(\mathbf{x}), \mathbf{p}_k)).$$

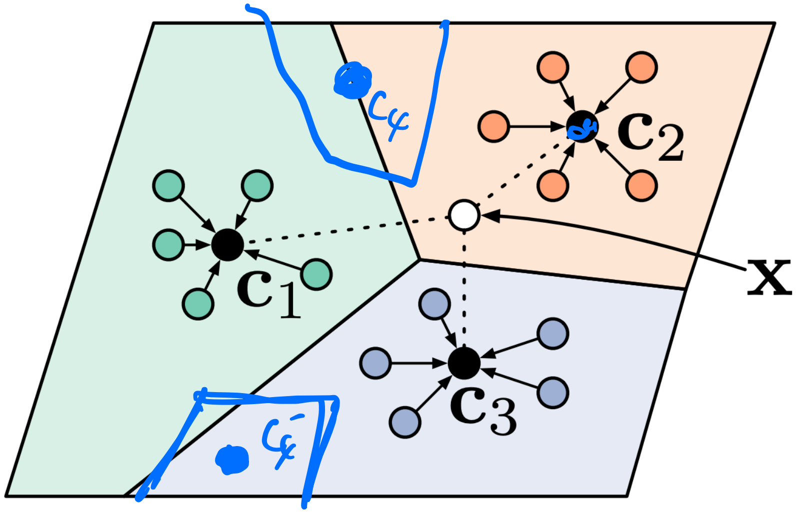
Snell et al. Prototypical Networks for Few-shot Learning. NIPS 2017.
Allen et al. Infinite Mixture Prototypes for Few-Shot Learning. ICML 2019.

NYU

Representation and Memory

- Prototypical Network: Few-shot Classification
- Prototype = Avg. representation of a class
- 1 example = exemplar-based. Can be in between.

$$\mathbf{p}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x},y)\in S_k} f_\phi(\mathbf{x}_i).$$



$$p(y = k \mid \mathbf{x}) = \mathrm{softmax}(-d(f_\phi(\mathbf{x}), \mathbf{p}_k)).$$

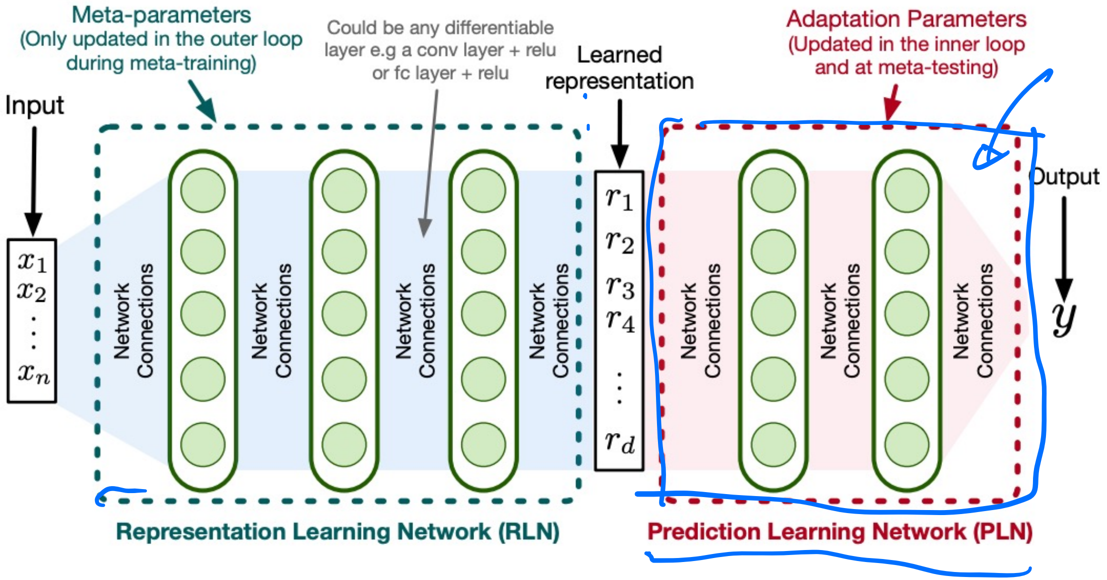
Snell et al. Prototypical Networks for Few-shot Learning. NIPS 2017.

Allen et al. Infinite Mixture Prototypes for Few-Shot Learning. ICML 2019.

NYU

Representation and Memory

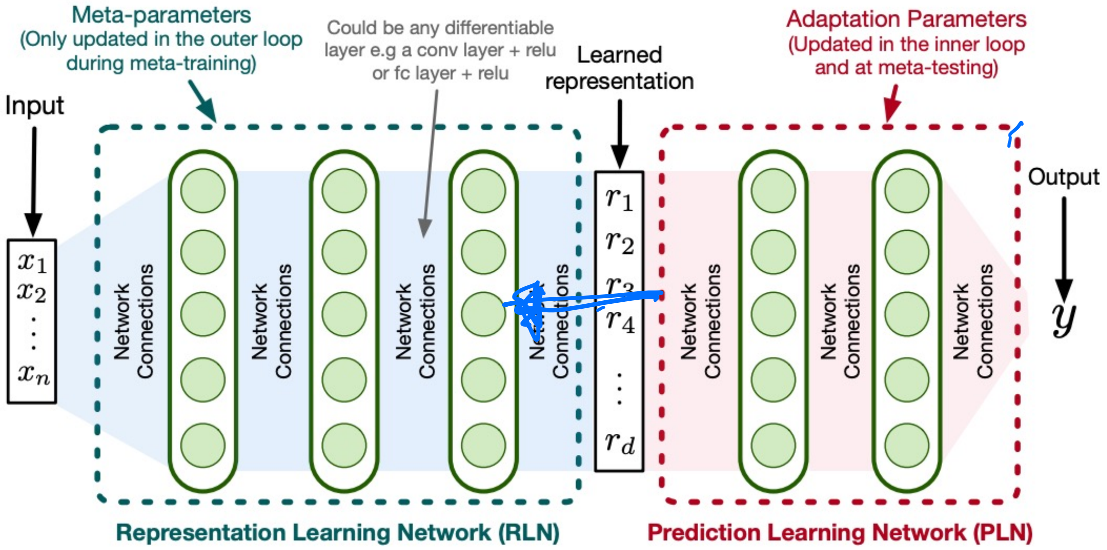
- Representation vs. Memory Layers



Javed & White. Meta-Learning Representations for Continual Learning. NeurIPS 2019.

Representation and Memory

- Representation vs. Memory Layers
- Learning to continually learn



Algorithm 2: Meta-Training : OML

Require: $p(\mathcal{T})$: distribution over CLP problems
Require: α, β: step size hyperparameters

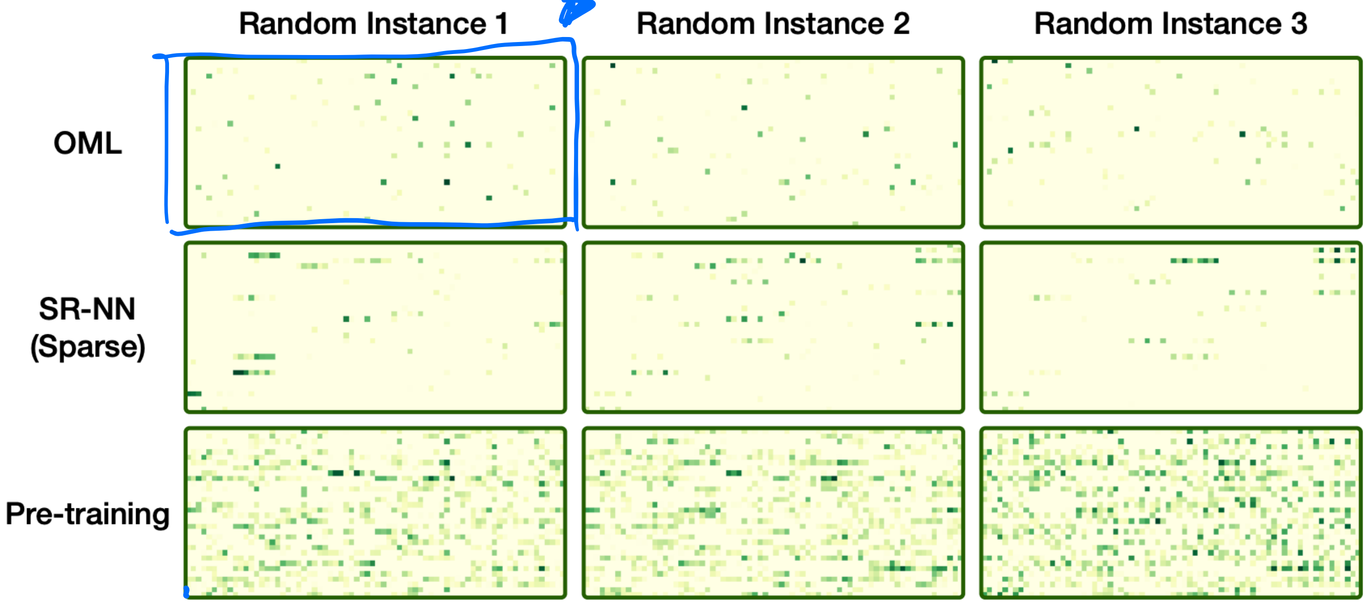
randomly initialize θ
while not done **do**
  randomly initialize W
  Sample CLP problem $\mathcal{T}_i \sim p(\mathcal{T})$
  Sample $\mathcal{S}_{train}$ from $p(\mathcal{S}_k|\mathcal{T}_i)$
  $W_0 = W$
  for $j = 1, 2, \ldots, k$ **do**
    $(X_j, Y_j) = \mathcal{S}_{train}[j]$
    $W_j = W_{j-1} - \alpha \nabla_{W_{j-1}} \ell_i(f_{\theta, W_{j-1}}(X_j), Y_j)$
  end for
  Sample S_{test} from $p(\mathcal{S}_k|\mathcal{T}_i)$
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \ell_i(f_{\theta, W_k}(S_{test}[:, 0]), S_{test}[:, 1])$
end while

Javed & White. Meta-Learning Representations for Continual Learning. NeurIPS 2019.

NYU

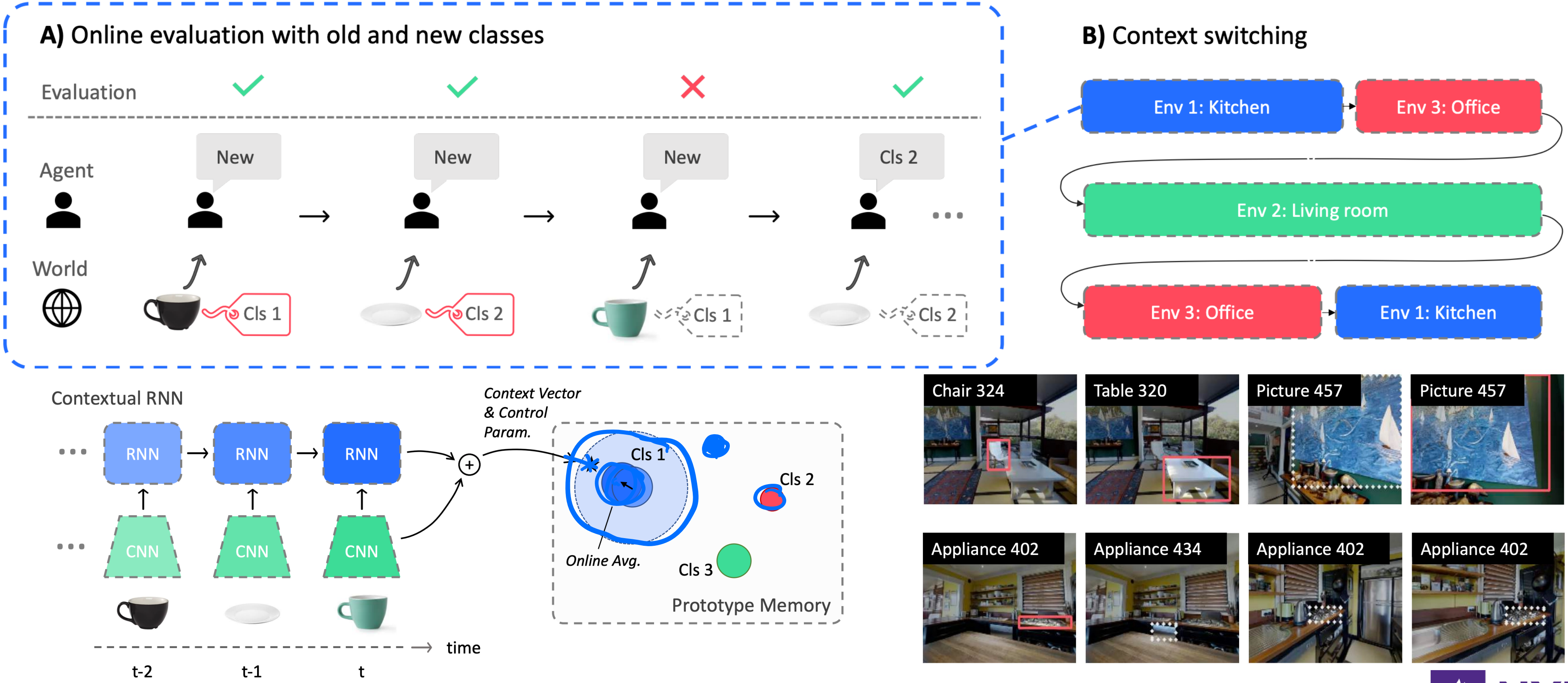
Representation and Memory

- Meta-learning leads to sparse representation suitable for continual learning.



Javed & White. Meta-Learning Representations for Continual Learning. NeurIPS 2019.

Online Continual Few-Shot Learning



Ren et al. Wandering Within a World: Online Contextualized Few-Shot Learning. ICLR 2021.

Online Continual Few-Shot Learning

- Compared to OML: Using prototype memory vs. generic MLP.

Online Continual Few-Shot Learning

- Compared to OML: Using prototype memory vs. generic MLP.
- Both learning representations through online learning episodes.

Ren et al. Wandering Within a World: Online Contextualized Few-Shot Learning. ICLR 2021.

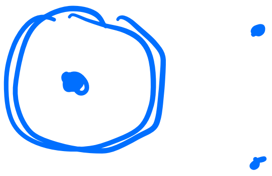
Online Continual Few-Shot Learning

- Compared to OML: Using prototype memory vs. generic MLP.
- Both learning representations through online learning episodes.
- Learning contextual representations (for context shifts).

Ren et al. Wandering Within a World: Online Contextualized Few-Shot Learning. ICLR 2021.

Online Continual Few-Shot Learning

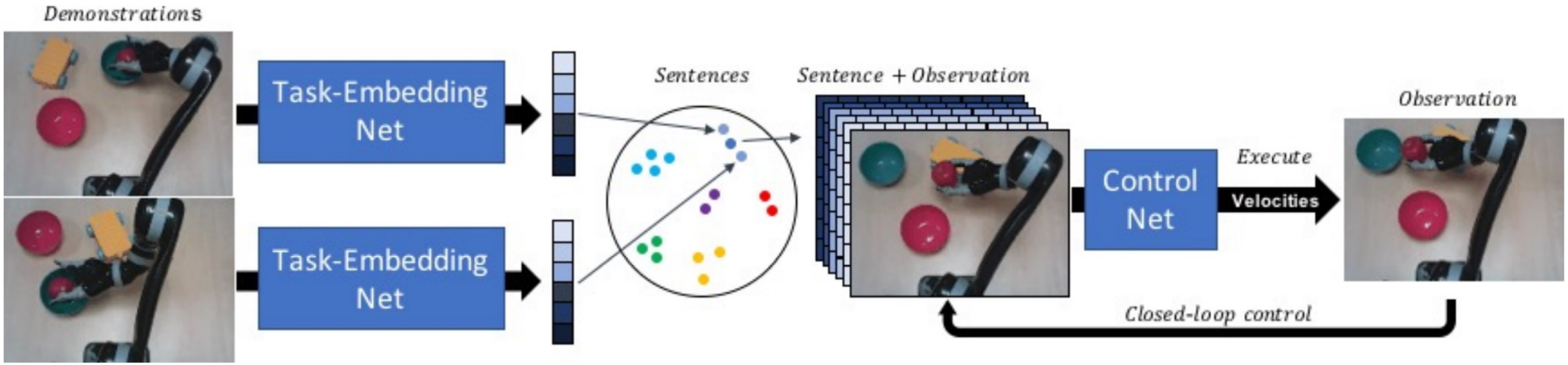
- Compared to OML: Using prototype memory vs. generic MLP.
- Both learning representations through online learning episodes.
- Learning contextual representations (for context shifts).
- Learning to output unknowns.



Ren et al. Wandering Within a World: Online Contextualized Few-Shot Learning. ICLR 2021.

Few-Shot Imitation Learning

- The idea of prototype learning can also be applied to skill learning.



Finn et al. One-Shot Visual Imitation Learning via Meta-Learning. CoRL 2017.
James et al. Task-Embedded Control Networks for Few-Shot Imitation Learning. CoRL 2018.

NYU

Few-Shot Imitation Learning

- The idea of prototype learning can also be applied to skill learning.
- Embedding learning:

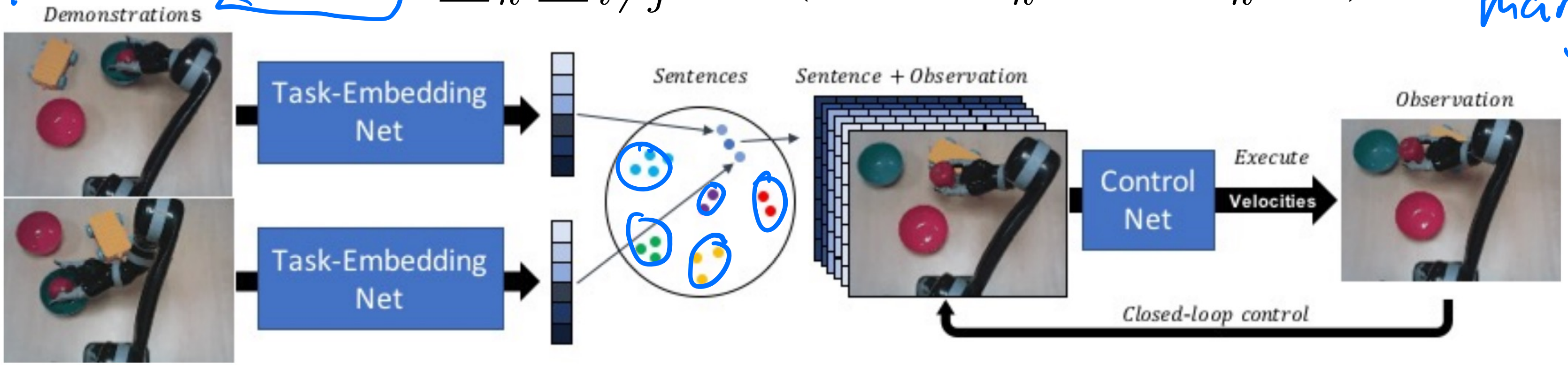
traj.

$$s^j = \frac{1}{K}\sum_{\tau_k^j \in \mathcal{T}^j} \boxed{f_\theta(\tau_k^j)}. \qquad \tilde{s}^j = s^j / \|s^j\|.$$

prototype.

$$\boxed{\mathcal{L}_{emb}} = \sum_k \sum_{i \neq j} \max(0, \Delta - \tilde{s}_k^j \cdot \tilde{s}^j + \tilde{s}_k^j \cdot \tilde{s}^i).$$

max-margin

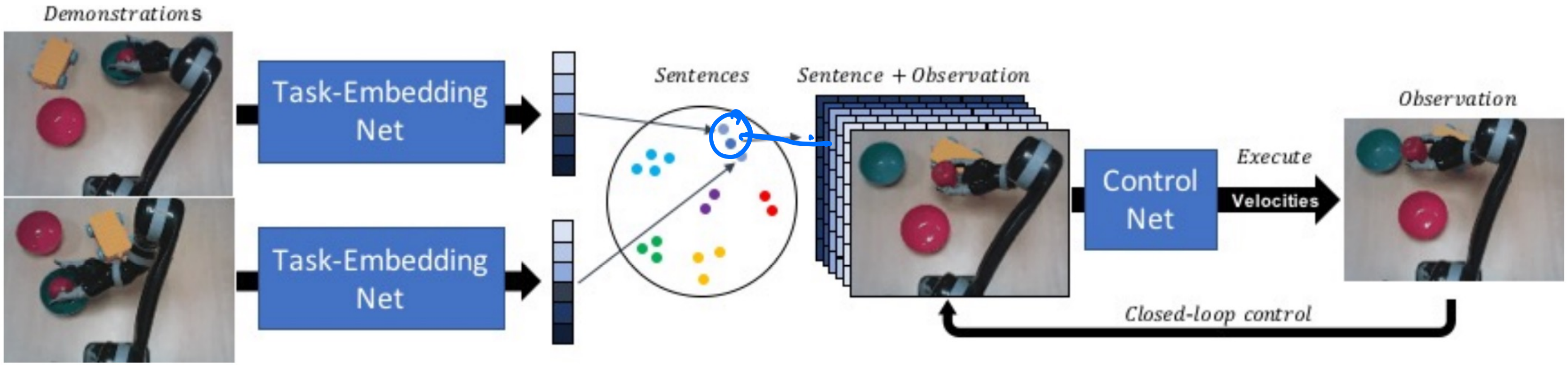


Finn et al. One-Shot Visual Imitation Learning via Meta-Learning. CoRL 2017.
James et al. Task-Embedded Control Networks for Few-Shot Imitation Learning. CoRL 2018.

Few-Shot Imitation Learning

- Control learning:

$$\mathcal{L}_{ctr} = \sum_j \sum_{(o,a)\in\tau^j} \|\underbrace{\pi(o, s^j)}_{\text{Output}} - \underbrace{a}_{\text{action.}}\|_2^2$$

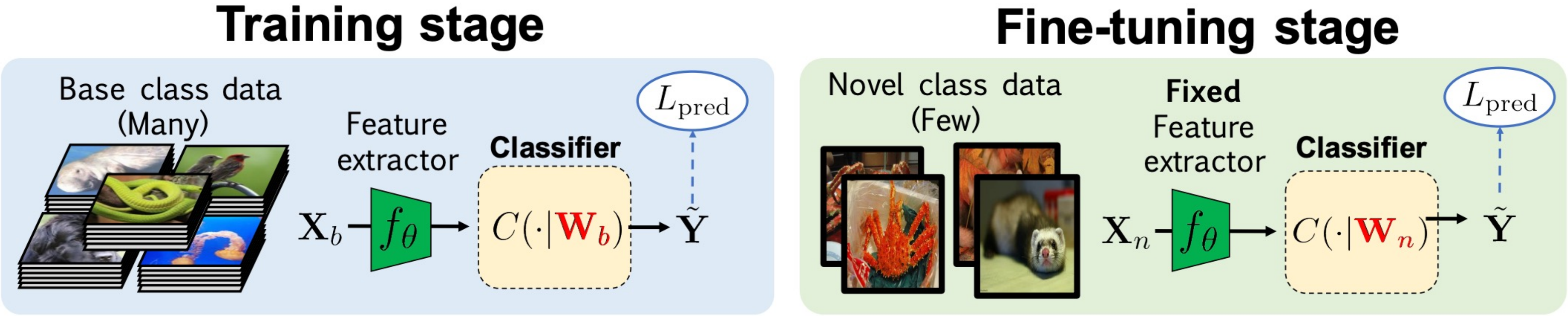


James et al. Task-Embedded Control Networks for Few-Shot Imitation Learning. CoRL 2018.

NYU

Leveraging Pretrained Representations

- Need a strong pretrained network.

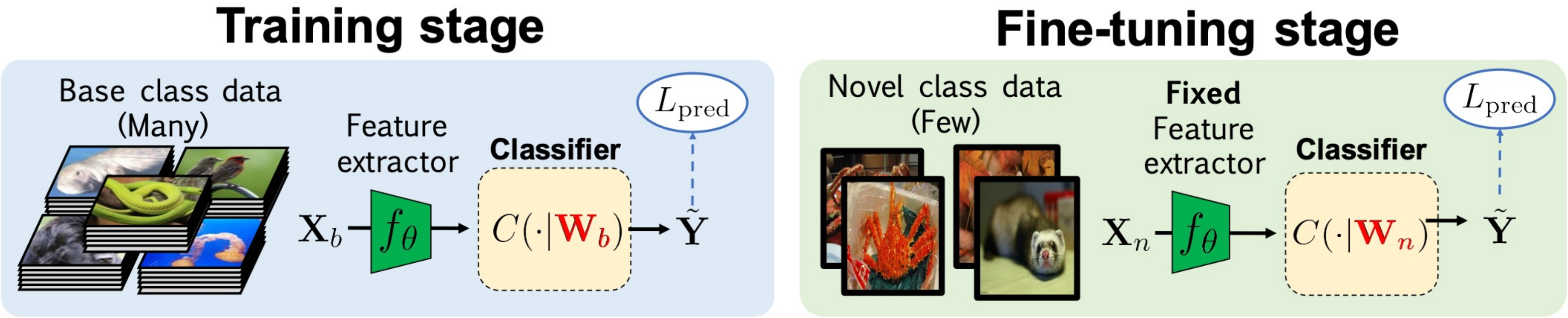


Chen et al. A Closer Look at Few-shot Classification. ICLR 2019.

NYU

Leveraging Pretrained Representations

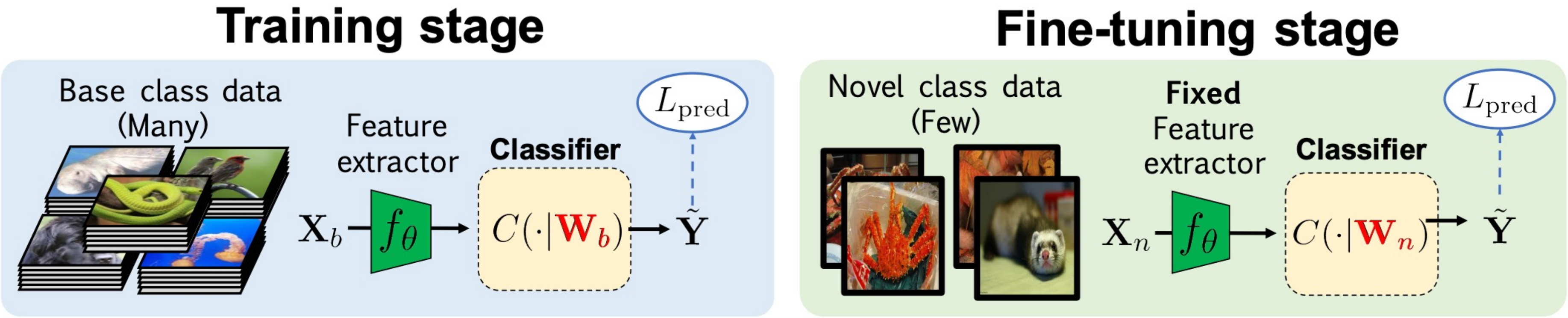
- Need a strong pretrained network.
- Works well for few-shot classification.



Chen et al. A Closer Look at Few-shot Classification. ICLR 2019.

Leveraging Pretrained Representations

- Need a strong pretrained network.
- Works well for few-shot classification.
- Once again proves that representation is crucial.



Chen et al. A Closer Look at Few-shot Classification. ICLR 2019.

In-Context Learning (ICL)

- Traditionally, learning through parameters θ.

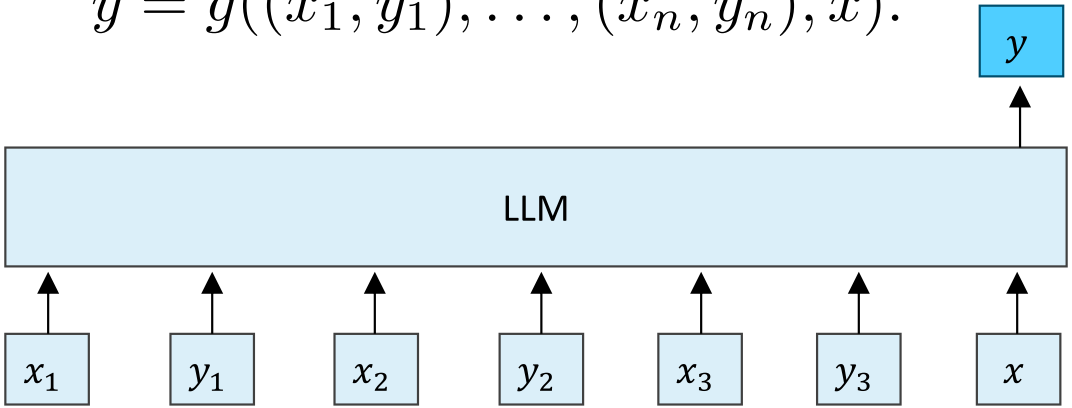
In-Context Learning (ICL)

- Traditionally, learning through parameters θ.
- ICL does not optimize any parameters, just put the training data in the context.

In-Context Learning (ICL)

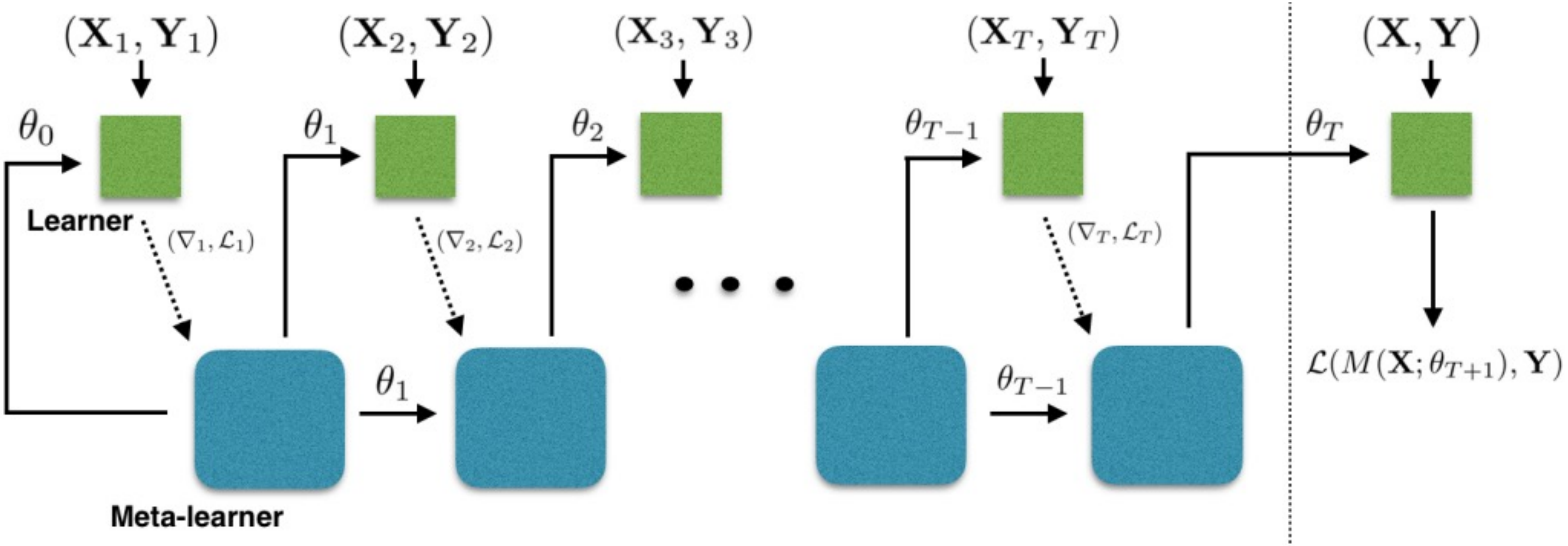
- Traditionally, learning through parameters θ.
- ICL does not optimize any parameters, just put the training data in the context.
- Inner optimization loop done in a sequence model (RNN, Transformer, etc.)

$$y = g((x_1, y_1), \ldots, (x_n, y_n), x).$$



Meta-Learning LSTM

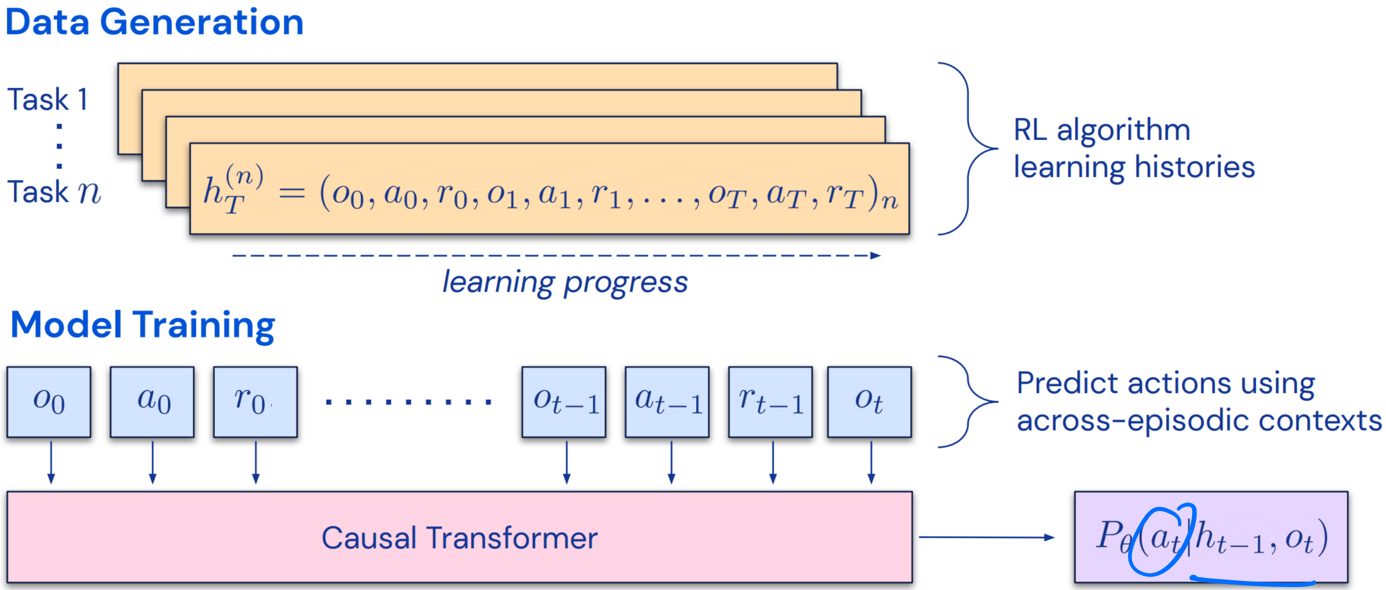
- An earlier sequential meta-learning paradigm before ICL.
- Using hidden states as "parameters"



Ravi & Larochelle. Optimization as a Model for Few-Shot Learning. ICLR 2017.

In-Context RL

- Use sequences of Obervation (o_t), Action (a_t) and Reward (r_t) generated by standard RL algorithms.



Laskin et al. In-context Reinforcement Learning with Algorithm Distillation ICLR 2023.

NYU

Test-Time Tuning/Adaptation

- Context Retrieval, e.g. nearest neighbors
- Finetuning on examples
  - Full finetuning
  - Low-rank adaptation
  - Prompt tuning
- Self-supervision objectives

Summary: Towards Embodied Learning Agents

- A modularized but differentiable end-to-end architecture
  - Perception
  - Prediction
  - Planning
  - Mapping
  - Memory

Summary: Towards Embodied Learning Agents

- A modularized but differentiable end-to-end architecture
  - Perception
  - Prediction
  - Planning
  - Mapping
  - Memory
- A combination of learning signals

Summary: Towards Embodied Learning Agents

- A modularized but differentiable end-to-end architecture
  - Perception
  - Prediction
  - Planning
  - Mapping
  - Memory
- A combination of learning signals
  - Direct supervision, energy-based (MSE, Contrastive, Max-Margin, Denoising)

NYU

Summary: Towards Embodied Learning Agents

- A modularized but differentiable end-to-end architecture
  - Perception
  - Prediction
  - Planning
  - Mapping
  - Memory
- A combination of learning signals
  - Direct supervision, energy-based (MSE, Contrastive, Max-Margin, Denoising)
  - Self-supervision (InfoNCE, MSE, Cross-entropy)

Summary: Towards Embodied Learning Agents

- A modularized but differentiable end-to-end architecture
  - Perception
  - Prediction
  - Planning
  - Mapping
  - Memory
- A combination of learning signals
  - Direct supervision, energy-based (MSE, Contrastive, Max-Margin, Denoising)
  - Self-supervision (InfoNCE, MSE, Cross-entropy)
  - Reconstruction

NYU

Summary: Towards Embodied Learning Agents

- A modularized but differentiable end-to-end architecture
  - Perception
  - Prediction
  - Planning
  - Mapping
  - Memory
- A combination of learning signals
  - Direct supervision, energy-based (MSE, Contrastive, Max-Margin, Denoising)
  - Self-supervision (InfoNCE, MSE, Cross-entropy)
  - Reconstruction
  - Future prediction

NYU

Summary: Towards Embodied Learning Agents

- A modularized but differentiable end-to-end architecture
  - Perception
  - Prediction
  - Planning
  - Mapping
  - Memory
- A combination of learning signals
  - Direct supervision, energy-based (MSE, Contrastive, Max-Margin, Denoising)
  - Self-supervision (InfoNCE, MSE, Cross-entropy)
  - Reconstruction
  - Future prediction
  - Reinforcement learning

Summary: Towards Embodied Learning Agents

- Useful inductive biases

Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding

Summary: Towards Embodied Learning Agents

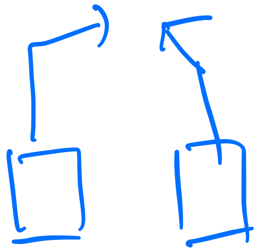
- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations

Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations
  - Permutation invariance / equivariance

Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations
  - Permutation invariance / equivariance
  - Representation invariance



Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations
  - Permutation invariance / equivariance
  - Representation invariance
  - Motion grounding

Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations
  - Permutation invariance / equivariance
  - Representation invariance
  - Motion grounding
  - Disentanglement, object-centric latents

Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations
  - Permutation invariance / equivariance
  - Representation invariance
  - Motion grounding
  - Disentanglement, object-centric latents
  - Cost volumes

Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations
  - Permutation invariance / equivariance
  - Representation invariance
  - Motion grounding
  - Disentanglement, object-centric latents
  - Cost volumes
  - Recursive iteration

NYU

Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations
  - Permutation invariance / equivariance
  - Representation invariance
  - Motion grounding
  - Disentanglement, object-centric latents
  - Cost volumes
  - Recursive iteration
  - Optimization / fixed point iteration

Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations
  - Permutation invariance / equivariance
  - Representation invariance
  - Motion grounding
  - Disentanglement, object-centric latents
  - Cost volumes
  - Recursive iteration
  - Optimization / fixed point iteration
  - Memory, replay, sparsity

NYU

Summary: Towards Embodied Learning Agents

- Useful inductive biases
  - Spatial grounding
  - Geometric projections / transformations
  - Permutation invariance / equivariance
  - Representation invariance
  - Motion grounding
  - Disentanglement, object-centric latents
  - Cost volumes
  - Recursive iteration
  - Optimization / fixed point iteration
  - Memory, replay, sparsity
  - Learning inductive biases, learning to learn

Topic Presentations

	Component 1	Component 2
Week 7 (Mar 6)	Continual Learning, Few-shot Learning (1 hr)	**Deep Learning for Structured Prediction** 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 11 (Apr 3)	**World Model 2** Pratyaksh Prabhav Rao, Sergey Sedov, Rooholla Khorrambakht	**End-to-End Planning** Raman Kumar Jha, Jovita Gandhi, Sushma Mareddy, Mrunal Sarvaiya
Week 12 (Apr 10)	**Continual Learning** Akshay Raman, Amey Joshi, Zifan Zhao	**Few-Shot Learning** Ellen Su, Xu Zhang, Swarali Borde
Week 13 (Apr 17)	Guest Lecture – Dr. Andrei Barsan (1hr)	**LLM Agents** Solim LeGris, Ravan Budda, Dan Zhao, Sunidhi Tandel

NYU

Project Presentations

Week 14 (Apr 24)	Project Presentations (7 teams)
Week 15 (May 1)	Project Presentations (7 teams)

- Apr 10: Sign up for a presentation slot. Week 14 Presenters get 2% bonus. First come first serve.
- Project topics and proposals to be shared in the class.

Today

- Tanishq Sardana: Segment Anything
- Qing Mu: DETR: End-to-End Object Detection
- Owais Saad Shuja: Latent Diffusion Models
- Discussion