Lifelong Learning (L^2)

Shirin Dora
Lecturer, Loughborough University

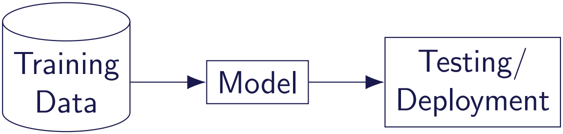
August 28, 2024

Outline

Loughborough University

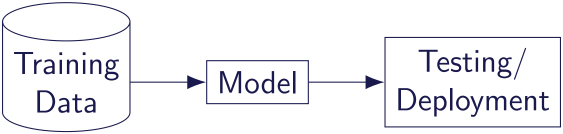
- Motivation: Why lifelong learning?
  - Classical machine learning
  - Catastrophic forgetting
- Definitions
  - Lifelong Learning
  - Three L^2 settings
- Current L^2 approaches
  - Replay Methods
  - Regularization Methods
  - Architectural Methods
- Evaluation of L^2 methods
- My Research interest

Classical Machine Learning



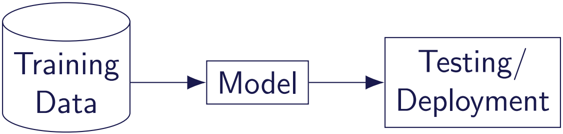
- **Classical machine learning**: Isolated single-task learning.
- What is a task?

Classical Machine Learning



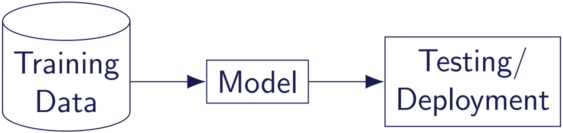
- **Classical machine learning**: Isolated single-task learning.
- What is a task?
  - Classification: cars vs truck, MNIST.
  - Regression: stock prices, rainfall.

Classical Machine Learning



- **Classical machine learning**: Isolated single-task learning.
- What is a task?
  - Classification: cars vs truck, MNIST.
  - Regression: stock prices, rainfall.
- Each model performs a single task.
  - Support vector machines, Deep neural networks.

Classical Machine Learning



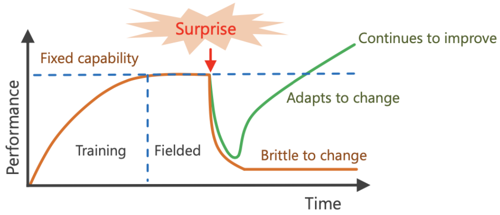
- **Classical machine learning**: Isolated single-task learning.
- What is a task?
  - Classification: cars vs truck, MNIST.
  - Regression: stock prices, rainfall.
- Each model performs a single task.
  - Support vector machines, Deep neural networks.
  - Very effective but there are limitations!

Classical Machine Learning

Key issues in classical machine learning

- No lifelong/continual learning
  - Learning occurs in isolation.
  - Knowledge accumulation is not possible.
  - Knowledge transfer is not possible.

Classical Machine Learning

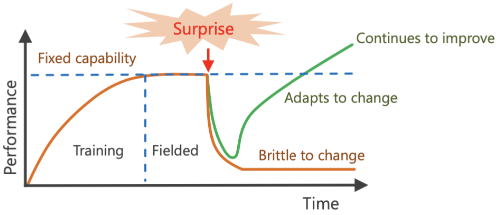


Key issues in classical machine learning

- No lifelong/continual learning
  - Learning occurs in isolation.
  - Knowledge accumulation is not possible.
  - Knowledge transfer is not possible.
- Assumes stationarity: Can't handle environment changes.

Classical Machine Learning

Loughborough University



Key issues in classical machine learning

- No lifelong/continual learning
  - Learning occurs in isolation.
  - Knowledge accumulation is not possible.
  - Knowledge transfer is not possible.
- Assumes stationarity: Can't handle environment changes.
- No more learning is possible after deployment.

Catastrophic Forgetting

Easy Solution

- Train a single model on multiple tasks one after another (in a sequence).

Catastrophic Forgetting

Easy Solution

- Train a single model on multiple tasks one after another (in a sequence).
- Learning requires updating parameters of the **shared** model for each task (like weights in a neural network).

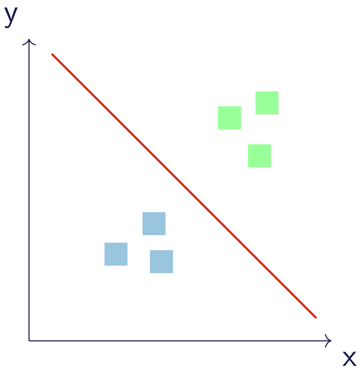
Catastrophic Forgetting

Easy Solution

- Train a single model on multiple tasks one after another (in a sequence).
- Learning requires updating parameters of the **shared** model for each task (like weights in a neural network).
  - Knowledge acquired from previous tasks will be overwritten.
  - Performance on previous tasks deteriorate as we learn new tasks - *Catastrophic forgetting*.

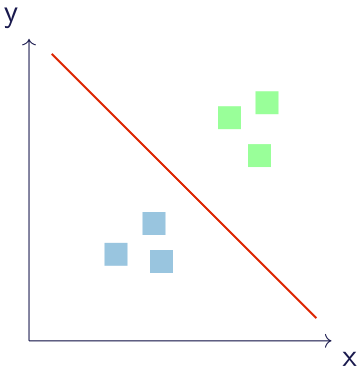
Catastrophic Forgetting

- **Task 1**
  - 2 classes: blue and green squares.



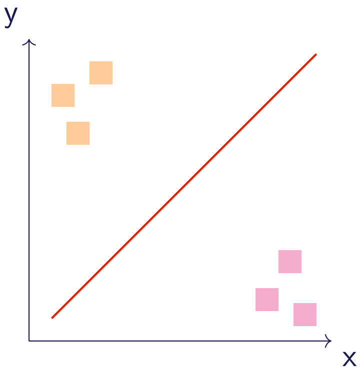
Catastrophic Forgetting

- **Task 1**
  - 2 classes: blue and green squares.
- Find a line that separates two classes.
  - $y = mx + c$
  - Model = Finding m and c.



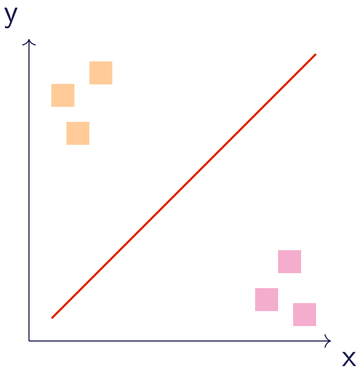
Catastrophic Forgetting

- Apply the easy solution
  - Continue training the model on Task 2.
- **Task 2**
  - 2 classes: orange and magenta squares.
  - Find a line that separates two classes.



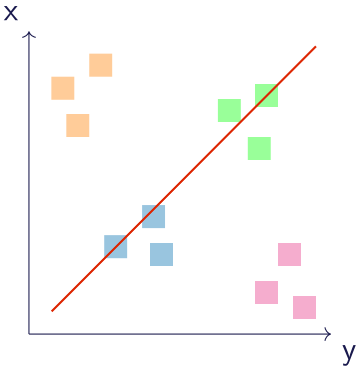
Catastrophic Forgetting

- Apply the easy solution
  - Continue training the model on Task 2.
- **Task 2**
  - 2 classes: orange and magenta squares.
  - Find a line that separates two classes.
- Red line is our model now.



Catastrophic Forgetting

- We wanted a **single shared model** for all tasks.
- New model (line) doesn't work for Task 1.
- Learning Task 2 made model forget Task 1.
  - Catastrophic forgetting.
- Catastrophic forgetting is the most fundamental problem in L^2.



Definitions

Lifelong Learning (L^2)

- Learn a sequence of tasks, $T_1, T_2, \ldots, T_N, \ldots$ sequentially.
- Each task t has a dataset for training, $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$

Lifelong Learning (L^2)

- Learn a sequence of tasks, $T_1, T_2, \ldots, T_N, \ldots$ sequentially.
- Each task t has a dataset for training, $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$
- **Goal**: Learn each new task T_{N+1} sequentially:
  - *Without catastrophic forgetting*: Learning of new task T_{N+1} should not result in degradation of accuracy for the previous N tasks.

Lifelong Learning (L^2)

Loughborough University

- Learn a sequence of tasks, $T_1, T_2, \ldots, T_N, \ldots$ sequentially.
- Each task t has a dataset for training, $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$
- **Goal**: Learn each new task T_{N+1} sequentially:
  - *Without catastrophic forgetting*: Learning of new task T_{N+1} should not result in degradation of accuracy for the previous N tasks.
  - *With knowledge transfer*: Utilize knowledge acquired from previous tasks to learn the new task T_{N+1} more effectively.

Lifelong Learning (L^2)

- Learn a sequence of tasks, $T_1, T_2, \ldots, T_N, \ldots$ sequentially.
- Each task t has a dataset for training, $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$
- **Goal**: Learn each new task T_{N+1} sequentially:
  - *Without catastrophic forgetting*: Learning of new task T_{N+1} should not result in degradation of accuracy for the previous N tasks.
  - *With knowledge transfer*: Utilize knowledge acquired from previous tasks to learn the new task T_{N+1} more effectively.
- **Assumption**: Once a task is learned, its data is no longer accessible (at least majority of it).

Lifelong Learning (L^2)

Three L^2 settings [Van de Ven, 2022]

1. Task incremental learning
   - Models are *partially* shared across tasks.
   - Task identity is required during testing.

Lifelong Learning (L^2)

Three L^2 settings [Van de Ven, 2022]

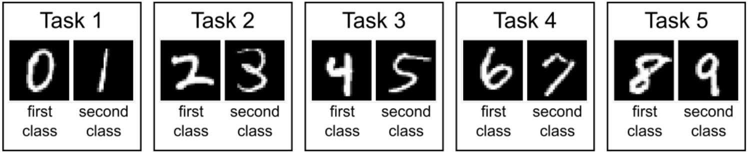
1. Task incremental learning
   - Models are *partially* shared across tasks.
   - Task identity is required during testing.
2. Domain incremental learning
   - All tasks have the same set of classes.
   - Task identity is not required during testing.
   - E.g. Classifying objects under different lightning conditions.

Lifelong Learning (L^2)

Three L^2 settings [Van de Ven, 2022]

1. Task incremental learning
   - Models are *partially* shared across tasks.
   - Task identity is required during testing.
2. Domain incremental learning
   - All tasks have the same set of classes.
   - Task identity is not required during testing.
   - E.g. Classifying objects under different lightning conditions.
3. Class incremental learning
   - Produce a single model from all tasks.
   - All classes in all tasks are handled by one model.

Three L^2 Settings



Task-IL	With task given, is it the 1st or 2nd class? (e.g., 0 or 1)
Domain-IL	With task unknown, is it a 1st or 2nd class? (e.g., in [0, 2, 4, 6, 8] or in [1, 3, 5, 7, 9])
Class-IL	With task unknown, which digit is it? (i.e., choice from 0 to 9)

Current Approaches

Current Approaches

- **Replay/Rehearsal-based methods**
- **Regularization-based methods**
- **Architectural methods**

Current Approaches

- **Replay/Rehearsal-based methods**
  - Replay samples from previous tasks while learning a new task.
- **Regularization-based methods**
- **Architectural methods**

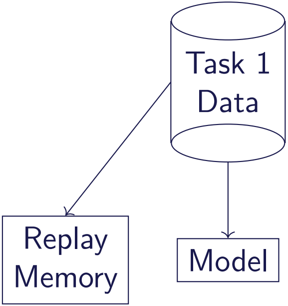
Current Approaches

- **Replay/Rehearsal-based methods**
  - Replay samples from previous tasks while learning a new task.
- **Regularization-based methods**
  - Update to model parameters is regularized according to a parameter's importance for previous tasks.
- **Architectural methods**

Current Approaches

- **Replay/Rehearsal-based methods**
  - Replay samples from previous tasks while learning a new task.
- **Regularization-based methods**
  - Update to model parameters is regularized according to a parameter's importance for previous tasks.
- **Architectural methods**
  - Expand network architecture to learn new tasks.

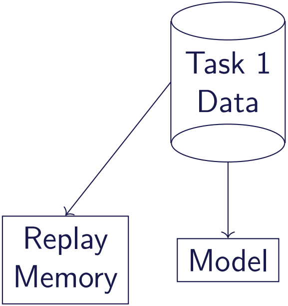
Replay Methods



Task 1 Training

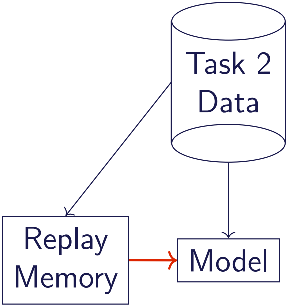
- Save subset of samples in replay memory.

Replay Methods



Task 1 Training

- Save subset of samples in replay memory.



Task 2 Training

- Replay stored samples from previous tasks.
- Save subset of samples in replay memory.

Replay Methods

Why replay works?

- For training, mix data from new task and replay samples.
  - **Replay samples**: Won't allow changes to weights important for previous tasks.
  - **New task samples**: Learn the new task.
- The quality of stored samples is very important!

Replay Methods

Why replay works?

- For training, mix data from new task and replay samples.
  - **Replay samples**: Won't allow changes to weights important for previous tasks.
  - **New task samples**: Learn the new task.
- The quality of stored samples is very important!

Is there another way to obtain replay samples?

Replay Methods

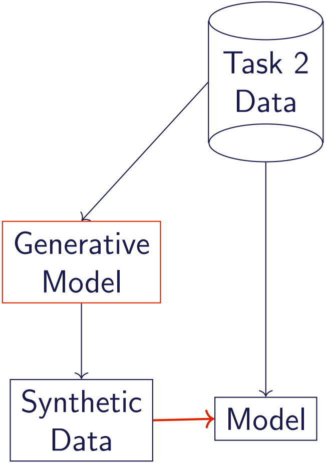
Why replay works?

- For training, mix data from new task and replay samples.
  - **Replay samples**: Won't allow changes to weights important for previous tasks.
  - **New task samples**: Learn the new task.
- The quality of stored samples is very important!

Is there another way to obtain replay samples?

- Train a generative model in-parallel to your classifier.
- Get replay samples from the generative model [Shin, 2017].

Replay Methods



- Exact replay
  - Actual samples from the original dataset are used for replay.
- Generative replay
  - Samples for replay are obtained from generative models.
  - L2 problem is shifted to generative models.

Replay Methods

- Learning in deep neural networks and brains with similarity-weighted interleaved learning, 2022 - Only previous memories with high similarity to the new data.

Coresets

- Gradient based sample selection for online continual learning, 2019 - Select samples that exhibit maximum variance in gradient.
- Online Continual Learning with Maximally Interfered Retrieval, 2019. - Samples most affected by a given gradient update.

Regularization Methods

Regularization Methods

- Use an additional penalty term during learning.
- Without regularization

$$\mathcal{L}_N = \sum_{x \in \mathcal{D}_N} \mathcal{E}(y_{output}, y_{desired}) \tag{1}$$

  - $\mathcal{E}$ could be MSE, cross entropy, etc.

Regularization Methods

- Use an additional penalty term during learning.
- Without regularization

$$\mathcal{L}_N = \sum_{x \in \mathcal{D}_N} \mathcal{E}(y_{output}, y_{desired}) \tag{1}$$

  - $\mathcal{E}$ could be MSE, cross entropy, etc.
- With regularization,

$$\mathcal{L}_N = \sum_{x \in \mathcal{D}_N} \mathcal{E} + \text{parameter-wise penalty} \tag{2}$$

Regularization Methods

Regularization: Old wine in new bottles!

$$\mathcal{L}_N = \sum_{x \in \mathcal{D}_N} \mathcal{E} + \lambda \sum_i \theta_i^2 \tag{3}$$

- **L2 Regularization**
- θ_i denotes parameters of your network.
- λ determines how strongly regularization is applied.
- *Almost* a standard technique to prevent overfitting.

Regularization Methods

Regularization: Old wine in new bottles!

$$\mathcal{L}_N = \sum_{x \in \mathcal{D}_N} \mathcal{E} + \lambda \sum_i \theta_i^2 \tag{3}$$

- **L2 Regularization**
- θ_i denotes parameters of your network.
- λ determines how strongly regularization is applied.
- *Almost* a standard technique to prevent overfitting.
- Keep θ_i close to zero.

$$\mathcal{L}_N = \sum_{x \in \mathcal{D}_N} \mathcal{E} + \lambda \sum_i (\theta_i - 0)^2 \tag{4}$$

Regularization Methods

Loughborough University

$$\mathcal{L}_B = \sum_{x \in B} \mathcal{E} + \lambda \sum_i \theta_i^2 \tag{5}$$

- λ is same for all θ_i.
- All parameters might not be equally important!

Regularization Methods

Loughborough University

$$\mathcal{L}_B = \sum_{x \in B} \mathcal{E} + \lambda \sum_i \theta_i^2 \tag{5}$$

- λ is same for all θ_i.
- All parameters might not be equally important!

$$\mathcal{L}_B = \sum_{x \in B} \mathcal{E} + \lambda_i \sum_i \theta_i^2 \tag{6}$$

- High λ_i for less important θ_i.
- Low λ_i for highly important θ_i.

Regularization Methods

Loughborough University

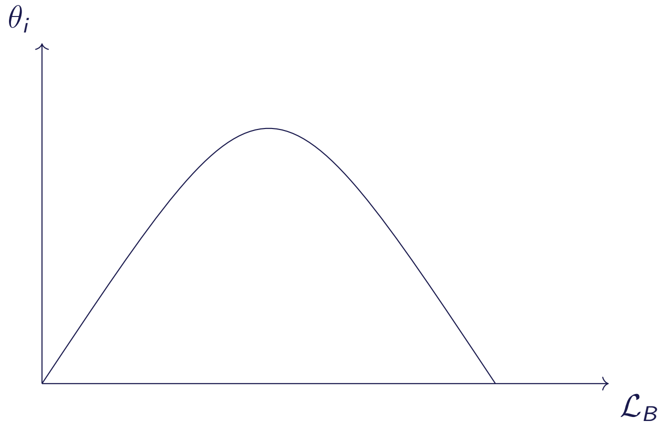
$$\mathcal{L}_B = \sum_{x \in B} \mathcal{E} + \lambda \sum_i \theta_i^2 \tag{5}$$

- λ is same for all θ_i.
- All parameters might not be equally important!

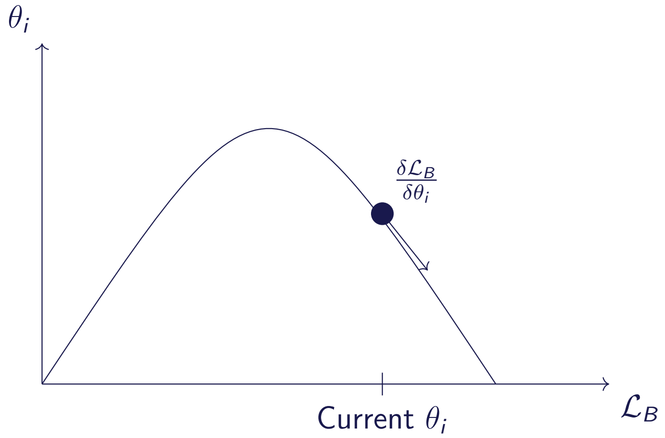
$$\mathcal{L}_B = \sum_{x \in B} \mathcal{E} + \lambda_i \sum_i \theta_i^2 \tag{6}$$

- High λ_i for less important θ_i.
- Low λ_i for highly important θ_i.
- **How do we set λ_i?**

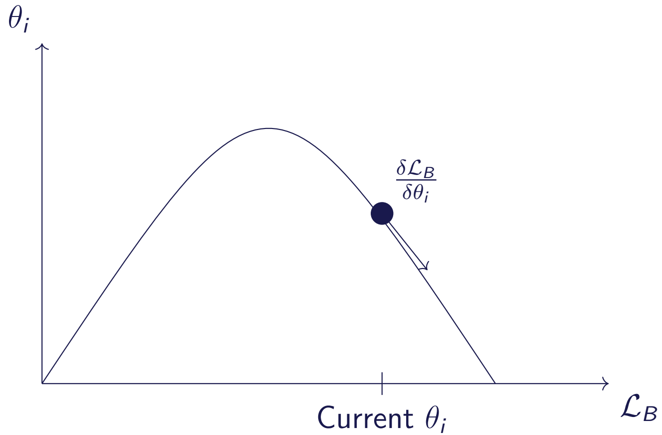
Regularization Methods



Regularization Methods

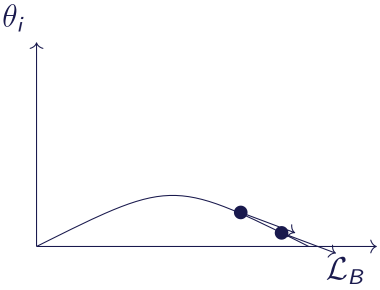


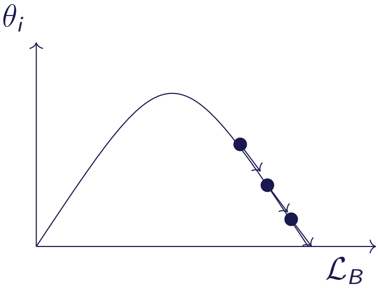
Regularization Methods



The usual gradient descent!

Regularization Methods





Low gradient

- Little change in weight.
- Less importance.

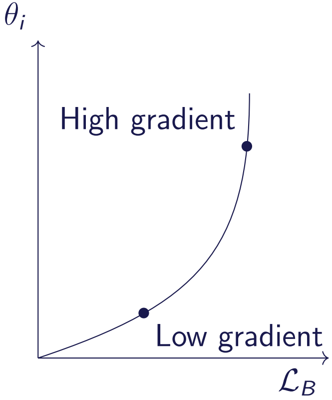
High gradient

- Large change in weight.
- High importance.

Regularization Methods

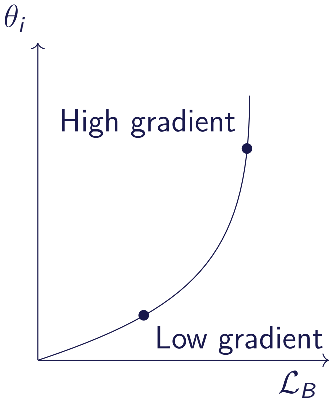
Continual learning

- Use parameters less important for previous tasks to learn new task.
  - Low gradient for previous tasks.
  - High gradient for current task.



Continual learning

- Use parameters less important for previous tasks to learn new task.
  - Low gradient for previous tasks.
  - High gradient for current task.
- Double derivative (gradient of the gradient) represents speed of change in the gradient.



Regularization Methods

$$\mathcal{L}_{new} = \sum_{x \in new} \mathcal{E} + c\lambda_i \sum_i (\theta_i - \theta_{prev})^2 \tag{7}$$

- θ_{prev} represents network parameters after learning previous tasks.
- λ_i would be estimated using double derivative [Kirkpatrick, 2016].

$$\lambda_i \propto \frac{\delta^2 \mathcal{L}}{\delta \theta_i^2}$$

- $(\theta_i - \theta_{prev})$ has the effect of keeping θ_i close to θ_{prev} when λ_i is high.
- c controls strength of regularization.

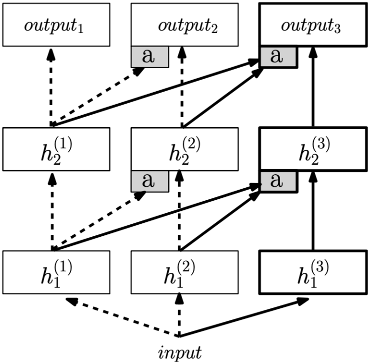
Regularization Methods

- Continual learning through synaptic intelligence - Computationally less intensive method for estimating λ.
- Memory Aware Synapses: Learning what (not) to forget - Effect of a parameter change on the network output.
- Optimization and Generalization of Regularization-Based Continual Learning: a Loss Approximation Viewpoint - Unifies various parameter-level regularization methods.
- Continual learning with node-importance based adaptive group sparse regularization, 2020 - Regularization at the level of neurons.

Architectural Methods

Architectural Methods

- Expand the network to learn new tasks.
- Brute force approach to L^2.
- Leads to parameter explosion.



Progressive Neural Networks[Rusu, 2016]

Architectural Methods

- Progress & compress: A scalable framework for continual learning. - Introduced a compression step for progressive neural networks.

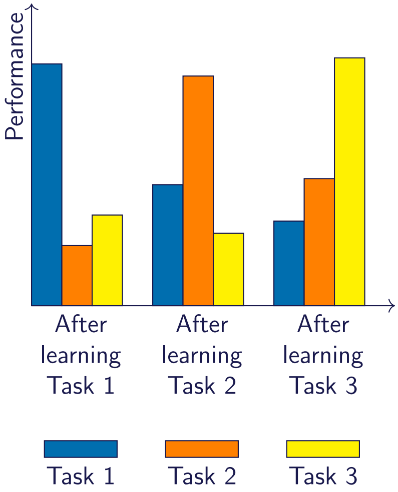
Prune/Add weights to the network

- Lifelong learning with dynamically expandable networks.
- Compacting, picking and growing for unforgetting continual learning.
- Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting.

Evaluating and Debugging L^2 Methods

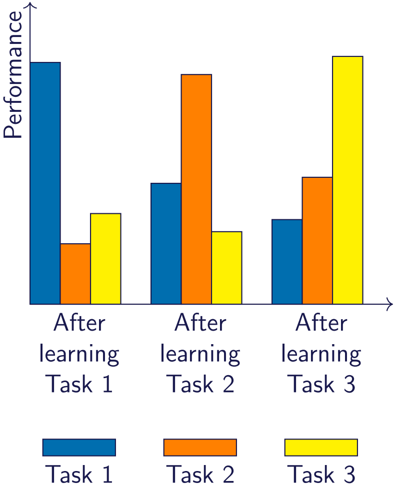
Interpreting L^2 Results

- What do you notice?



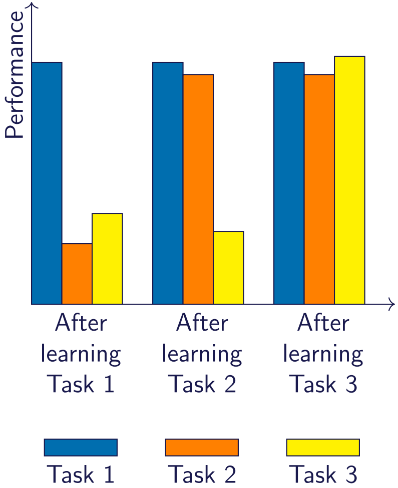
Interpreting L^2 Results

- What do you notice?
- Learned task performance is getting worse after each task is trained.
- This is a sign of catastrophic forgetting!



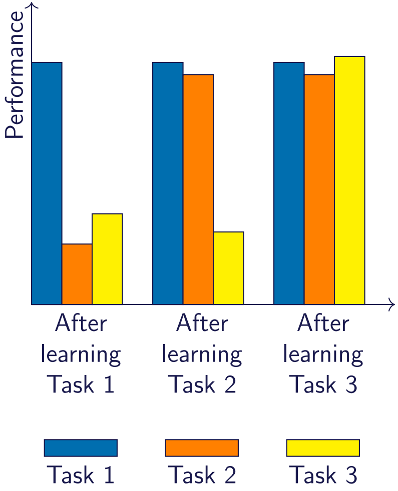
Interpreting L^2 Results

- What do you notice?



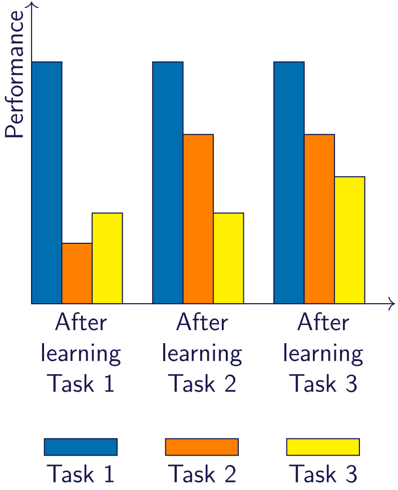
Interpreting L^2 Results

- What do you notice?
- Learned task performance is maintained after learning a new task.
- This indicates no forgetting!



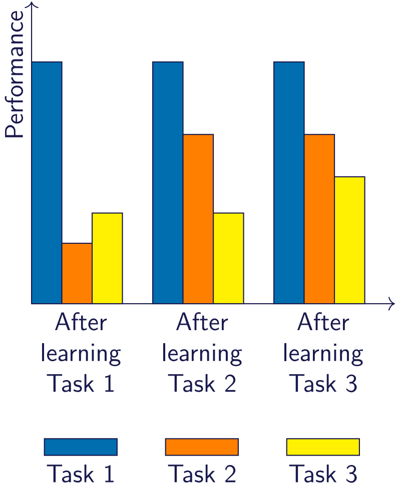
Interpreting L^2 Results

- What do you notice?



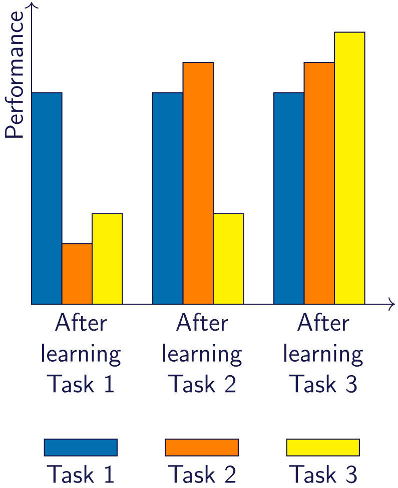
Interpreting L^2 Results

- What do you notice?
- Learned task performance is maintained but new tasks are not learned well enough.
- Problematic learning
  - Over regularization.
  - Limited network capacity.



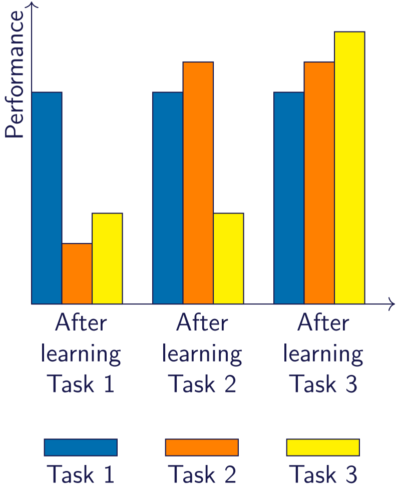
Interpreting L^2 Results

- What do you notice?



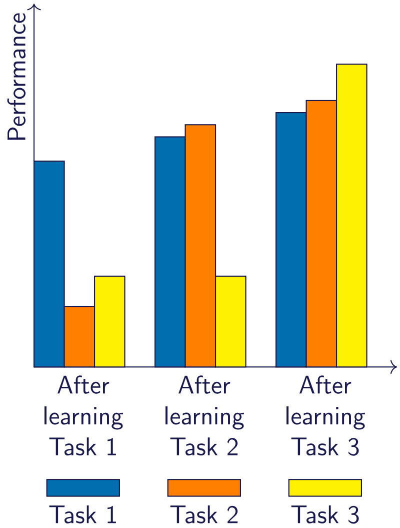
Interpreting L^2 Results

- What do you notice?
- Learned task performance is maintained
- New task performance is higher than previous task.
- **Forward transfer**
  - Future tasks utilize knowledge from previous tasks.



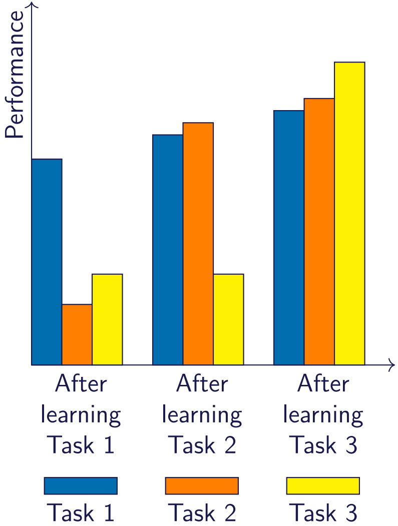
Interpreting L^2 Results

- What do you notice?



Interpreting L^2 Results

- What do you notice?
- Learned task performance is growing.
- New task performance is higher than previous task.
- Forward and **backward transfer**
  - Future tasks utilize knowledge from previous tasks.
  - Previous tasks gain from future tasks.



Metrics

- Metrics allow us to determine which scenario our model results represent.
- All L^2 metrics are computed from the accuracy matrix Lopez-paz, 2017.

Accuracy Matrix

Loughborough University

Testing task

Tasks trained so far

	T_1	T_2	T_3	T_4	$\cdots$
T_1	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$	$R_{1,4}$	
T_2	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$	$R_{2,4}$	
T_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$	$R_{3,4}$	
T_4	$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$	
$\vdots$					

$R_{m,n}$: The performance of the model on task T_n, after continually training till task T_m

Accuracy Matrix

Loughborough University

- Increasing → Positive Forward transfer

Testing task

Tasks trained so far

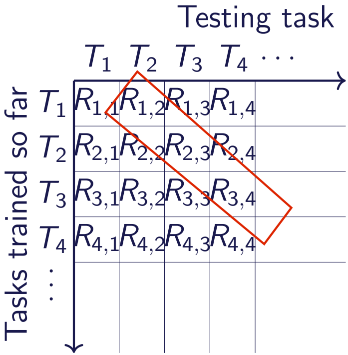
$$
\begin{array}{c|cccccc}
 & T_1 & T_2 & T_3 & T_4 & \cdots \\
\hline
T_1 & R_{1,1} & R_{1,2} & R_{1,3} & R_{1,4} & \\
T_2 & R_{2,1} & R_{2,2} & R_{2,3} & R_{2,4} & \\
T_3 & R_{3,1} & R_{3,2} & R_{3,3} & R_{3,4} & \\
T_4 & R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} & \\
\vdots & & & & &
\end{array}
$$

Accuracy Matrix

- Increasing $\rightarrow$ Positive Forward transfer
- Measuring forward transfer (FWT)

$$\frac{1}{T-1}\sum_{i=2}^{T}(R_{i-1,i} - b_i) \tag{8}$$

- b_i is accuracy before training.



Accuracy Matrix

- Measuring backward transfer (BWT)

$$\frac{1}{T-1}\sum_{i=1}^{T-1}(R_{T,i} - R_{i,i}) \tag{9}$$

Testing task

Tasks trained so far

$$\begin{array}{c|cccccc}
 & T_1 & T_2 & T_3 & T_4 & \cdots \\
\hline
T_1 & R_{1,1} & R_{1,2} & R_{1,3} & R_{1,4} & \\
T_2 & R_{2,1} & R_{2,2} & R_{2,3} & R_{2,4} & \\
T_3 & R_{3,1} & R_{3,2} & R_{3,3} & R_{3,4} & \\
T_4 & R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} & \\
\vdots & & & & &
\end{array}$$

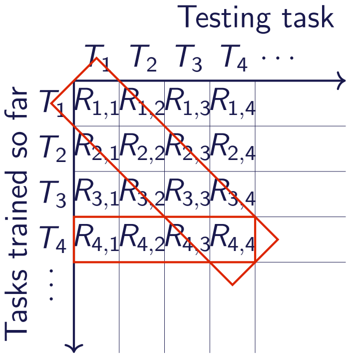
Accuracy Matrix

Loughborough University

- Measuring backward transfer (BWT)

$$\frac{1}{T-1}\sum_{i=1}^{T-1}(R_{T,i} - R_{i,i}) \tag{9}$$

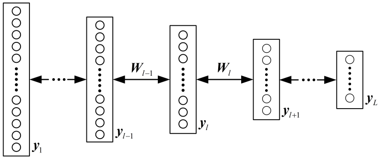
- $R_{i,i}$: accuracy after training a task.
- $R_{T,i}$: accuracy after training last task.



My Research Interests

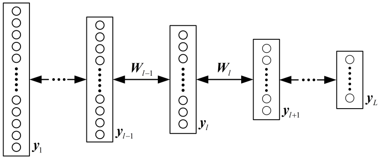
Efficient Generative L^2

- Deep Bidirectional Predictive Coding (DBPC) [Senhui 2023].
- Propagation can occur in both directions in DBPC.



Efficient Generative L^2

- Deep Bidirectional Predictive Coding (DBPC) [Senhui 2023].
- Propagation can occur in both directions in DBPC.
- Single network can do classification and generation.
  - Replay without additional memory!



Efficient Generative L^2

Task 1: 0 and 1

Task 2: 2 and 3

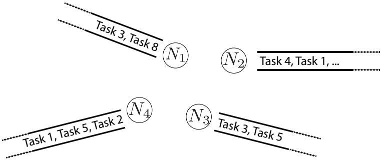
Task 3: 4 and 5

Task 4: 6 and 7

Task 5: 8 and 9

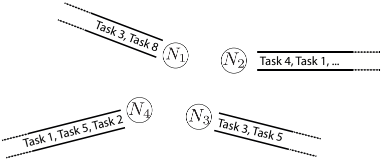
Distributed L^2

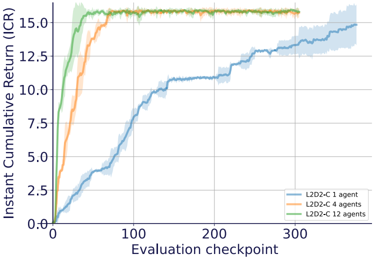
- Group of nodes perform L^2 collaboratively.
- Nodes encounter tasks in different sequence.



Distributed L^2

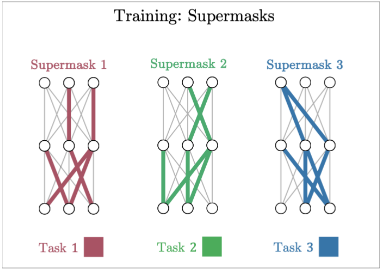
- Group of nodes perform L^2 collaboratively.
- Nodes encounter tasks in different sequence.
- Sharing knowledge improves speed of learning [Nath 2023].
  - 12 agents learned 16 tasks faster than a smaller group.
  - Agents exchanged knowledge through task-specific masks.





Distributed L^2

- Mask represents a subnetwork in a randomly initialized network [Wortsman 2020].
- Training involves estimating a mask for a given task.
- Testing involves processing a sample using the estimated mask.



Thank you!

s.dora@lboro.ac.uk

References

- van de Ven, G. M., Tuytelaars, T., & Tolias, A. S. (2022). Three types of incremental learning. Nature Machine Intelligence, 4(12), 1185-1197.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13), 3521-3526.
- Shin, H., Lee, J. K., Kim, J., & Kim, J. (2017). Continual learning with deep generative replay. Advances in neural information processing systems, 30.
- Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., ... & Hadsell, R. (2016). Progressive neural networks. arXiv preprint arXiv:1606.04671.
- Lopez-Paz, D., & Ranzato, M. A. (2017). Gradient episodic memory for continual learning. Advances in neural information processing systems, 30.

References

- Qiu, S., Bhattacharyya, S., Coyle, D., & Dora, S. (2023). Deep Predictive Coding with Bi-directional Propagation for Classification and Reconstruction. arXiv preprint arXiv:2305.18472.
- Nath, S., Peridis, C., Ben-Iwhiwhu, E., Liu, X., Dora, S., Liu, C., ... & Soltoggio, A. (2023). Sharing Lifelong Reinforcement Learning Knowledge via Modulating Masks. arXiv preprint arXiv:2305.10997.
- Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., & Farhadi, A. (2020). Supermasks in superposition. Advances in Neural Information Processing Systems, 33, 15173-15184.