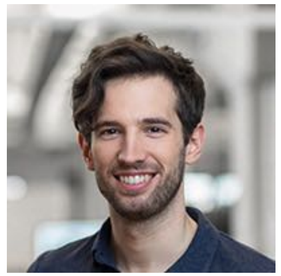
Why Efficiency Matters in CL

(How Continual Learning met Meta-Learning)

Marc'Aurelio Ranzato

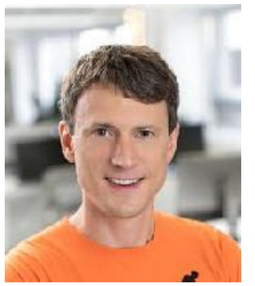
ranzato@fb.com

Arslan Chaudhry



David Lopez-Paz

Mohamed Elhoseiny



Marcus Rohrbach

A. Chaudhry et al. "Efficient Life-Long Learning with A-GEM", arXiv 2018

D. Lopez-Paz et al. "GEM for continual learning", NIPS 2017

facebook

Artificial Intelligence Research

Continual Learning Workshop @ NeurIPS, 6 December 2018

Neural Nets Are Powerful

- Quotes prior to 2010 or so:

- Quotes from the corridors of NeurIPS 2018:

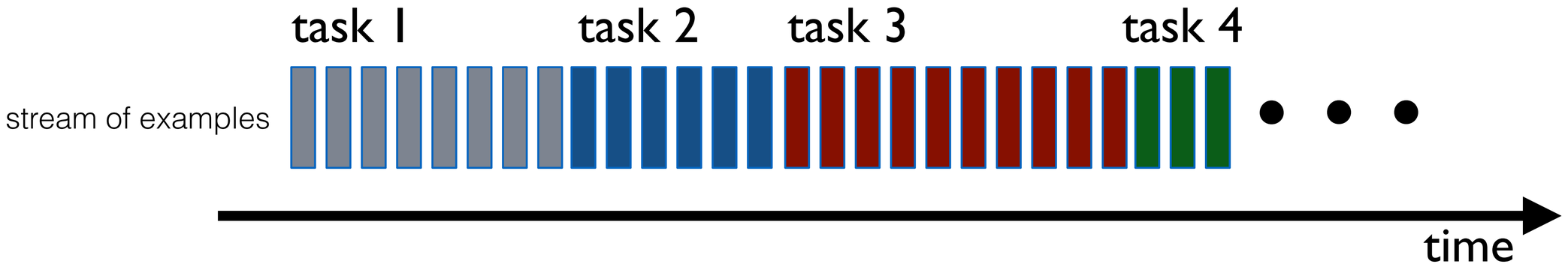
"Neural Nets can learn anything, given enough labeled data"

Grand Goal of Learning

- Arguably, neural nets can *learn almost any single task* provided enough supervision.
- We need to figure out how to *learn skills*, how to quickly adapt to new tasks.
- Key questions:
  - how to represent knowledge
  - how to transfer knowledge

Continual Learning (CL)

- CL is useful to assess the model's ability to quickly adapt to a new task.
- Catastrophic forgetting is bad because it prevents good transfer.

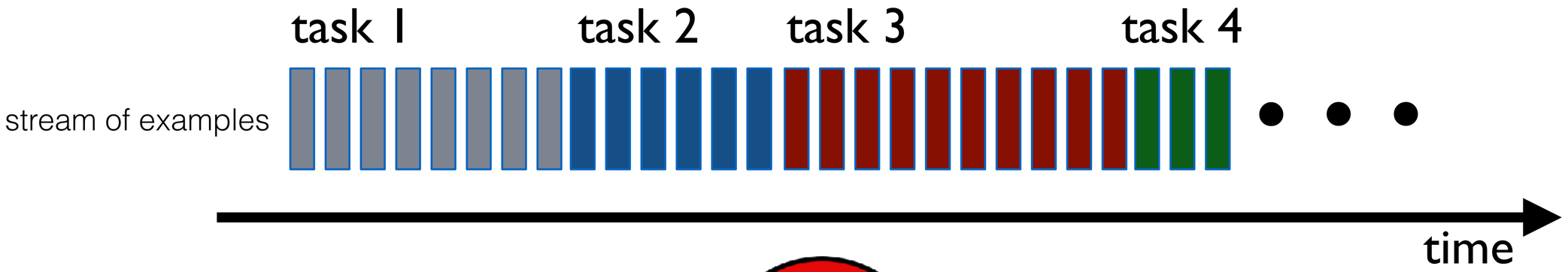


Sample Efficiency

- Past tasks are useful as long as we can transfer accrued knowledge. We shall **never learn from scratch** again!
- Under this view, sample efficiency is central to CL
  - learning protocol
  - metrics
- Other kinds of efficiency:
  - memory
  - computational

CL != SL

Current CL learning protocol is directly borrowed from supervised learning (SL).



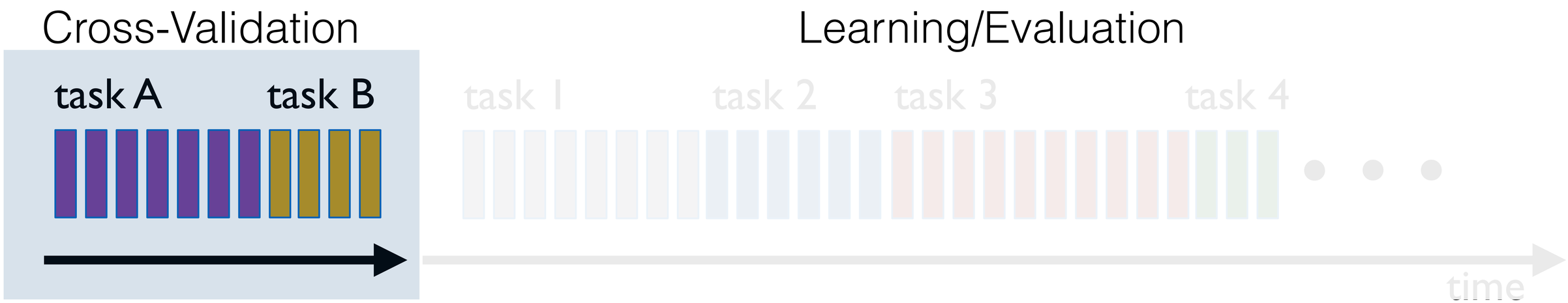
Standard Methodology

- for each hyper-paramter:
  - for each task:
    - do as many passes as needed
  - compute validation error
- report test error

Each task is learned using lots samples. Stream of data is replayed lots of times.

Usual supervised learning training/ evaluation protocol is not adequate, if we care about sample efficiency..

A New Methodology for CL



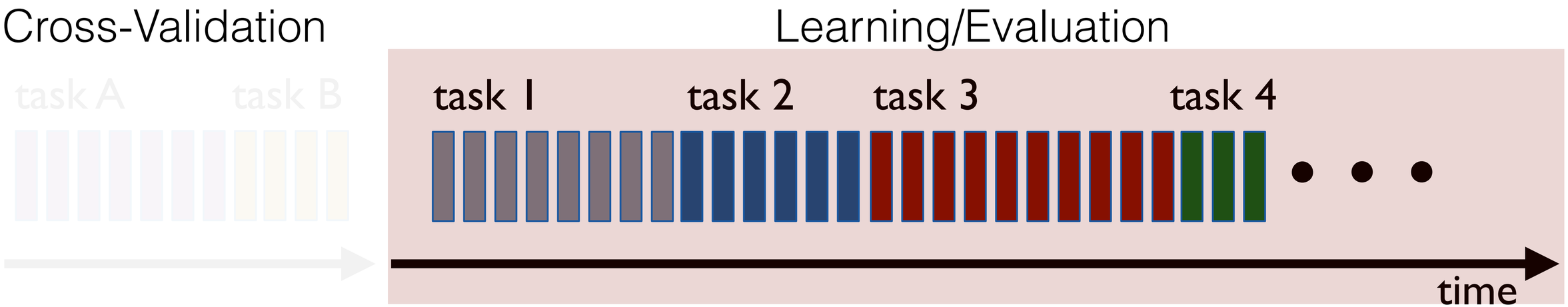
CL Methodology: **Cross-Validation**

- for each hyper-paramter:
  - for each *trial task* in {A,B}:
    - do a single pass
  - compute validation error
- select best hyper-parameter value

Phase 1 (optional): hyper-param selection on a small subset of tasks.

A. Chaudhry et al. “Efficient Life-Long Learning with A-GEM”, arXiv 2018

A New Methodology for CL



CL Methodology: **Learning & Evaluation**

- for each task in {1,2,3,4,…}:
  - do a single pass over training set
  - compute metrics on test set
- report metrics

Phase 2: actual learning and testing from a stream of examples.

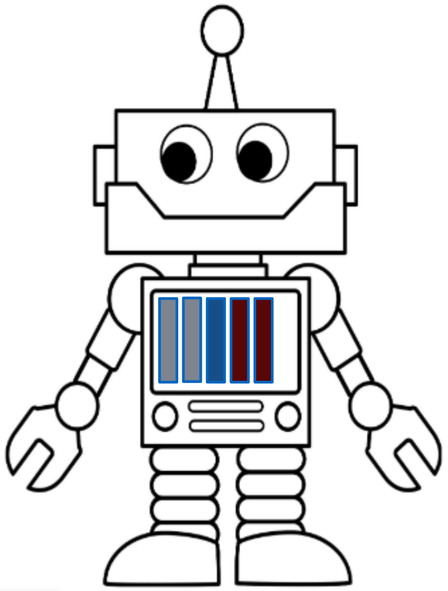
Only a single pass through the data.

A. Chaudhry et al. “Efficient Life-Long Learning with A-GEM”, arXiv 2018

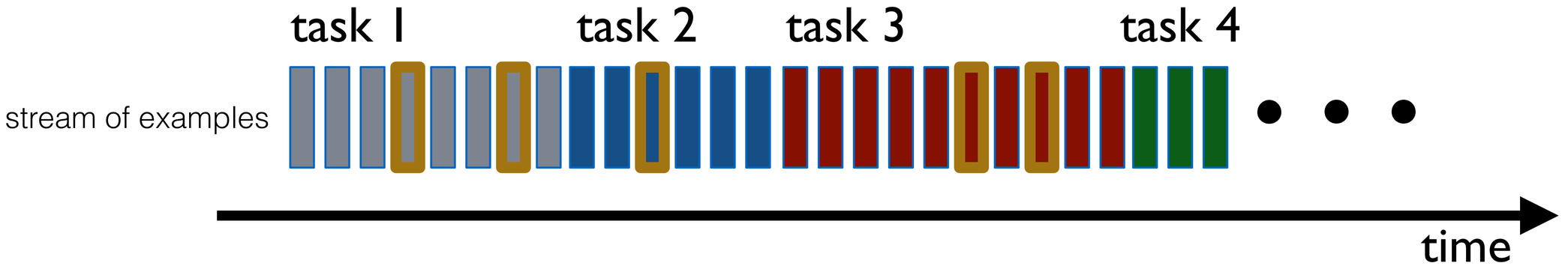
Model Efficiency

- Model has to be able to learn quickly (just a single pass over the data): sample efficiency.
- Model has to be robust to choice of hyper-parameters.
- Model has to be memory efficient: memory should not grow with the number of tasks!
- Model has to be computationally efficient: eventually CL models will operate in real-time.

Episodic Memory

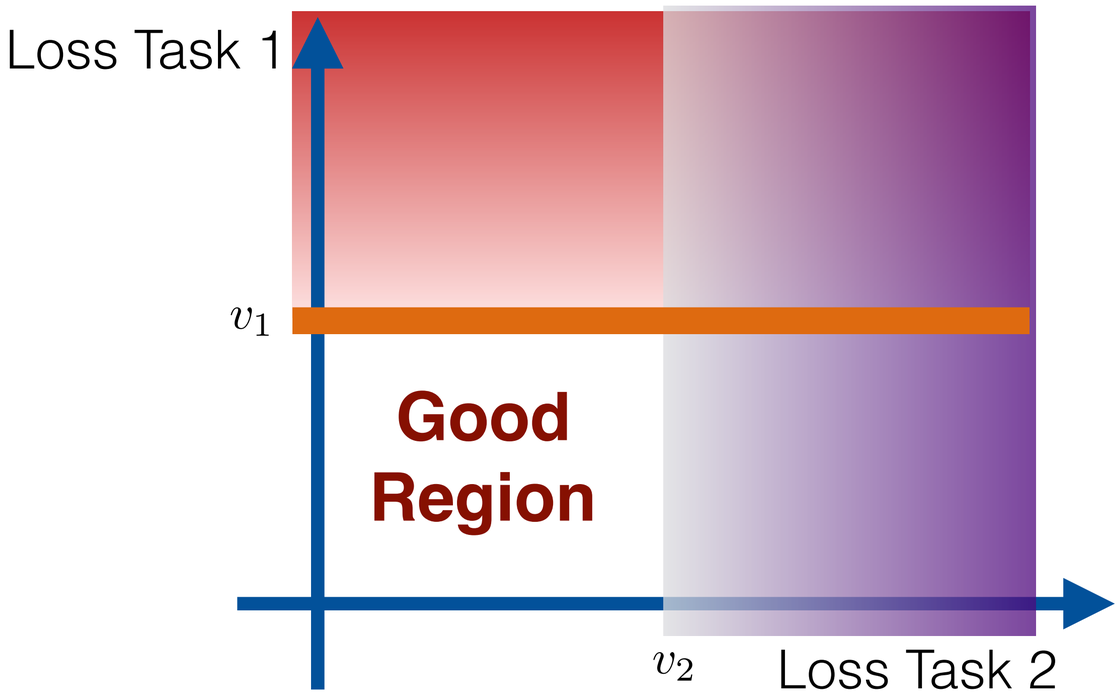


While observing data, the model may decide to store some data in its memory. Memory is very small and fixed in size.



Gradient Episodic Memory (GEM)

Assume we have already learned from examples of the first task, and we are now observing examples from task 2. We want to make sure we do not un-learn task 1.

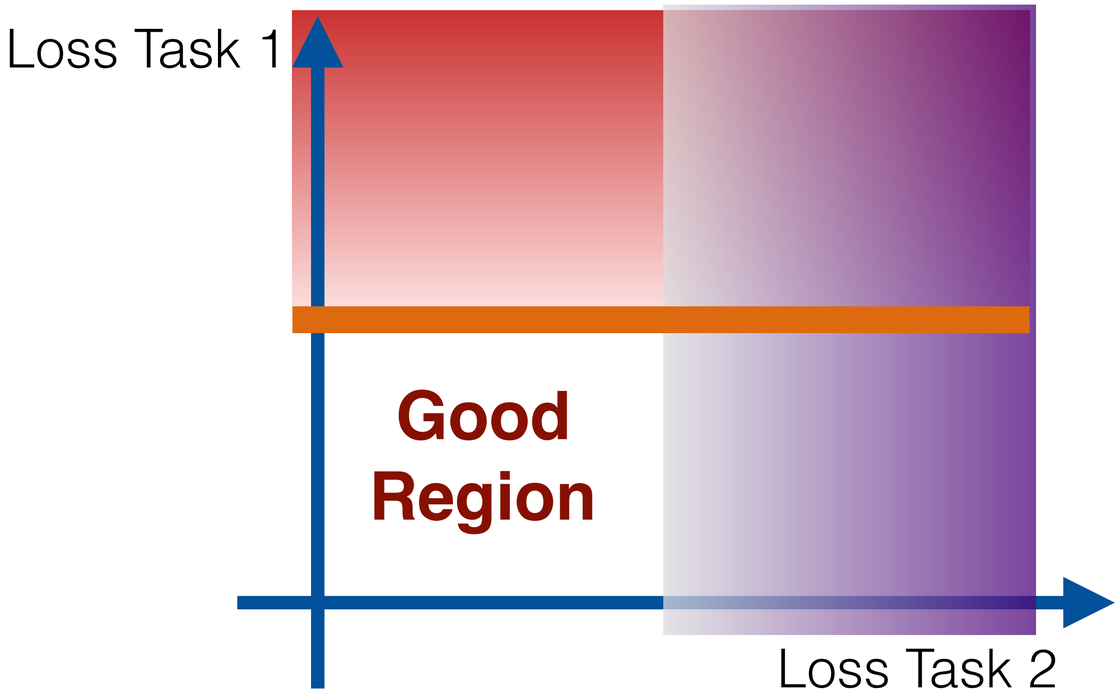


Loss on previous tasks (Task 1) is evaluated using episodic memory.

D. Lopez-Paz et al. “GEM for continual learning”, NIPS 2017

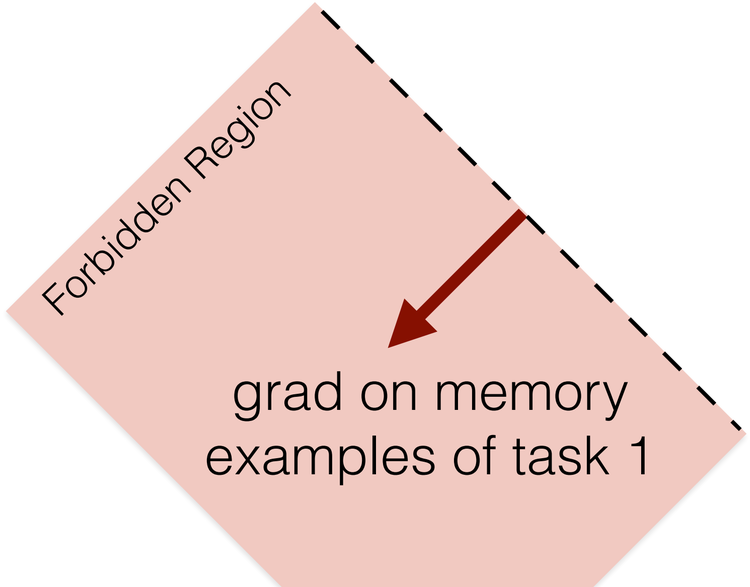
Gradient Episodic Memory (GEM)

Assume we have already learned from examples of the first task, and we are now observing examples from task 2. We want to make sure we do not un-learn task 1.



Gradient=direction of maximum increase of the loss

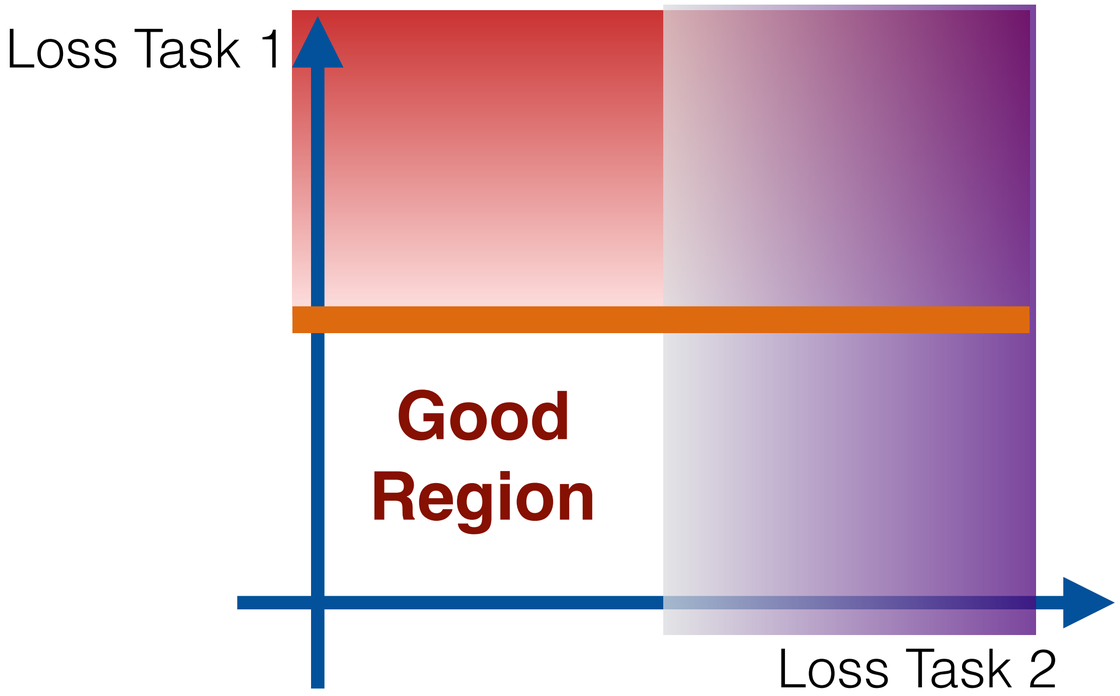
We can detect loss increase by checking the gradients:



D. Lopez-Paz et al. “GEM for continual learning”, NIPS 2017

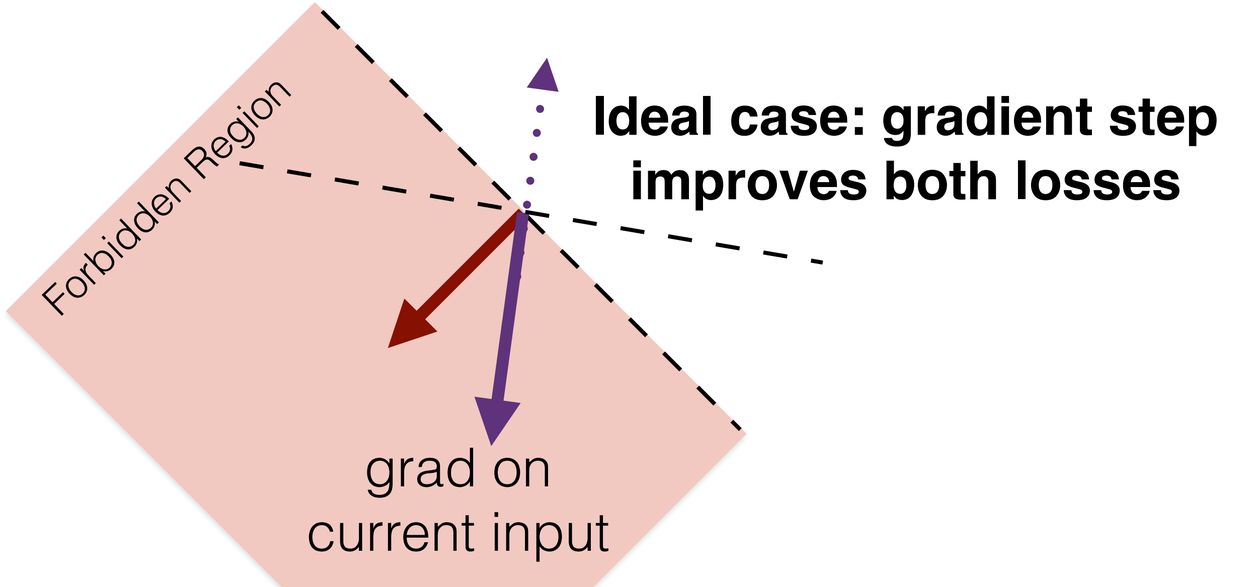
Gradient Episodic Memory (GEM)

Assume we have already learned from examples of the first task, and we are now observing examples from task 2. We want to make sure we do not un-learn task 1.



Gradient=direction of maximum increase of the loss

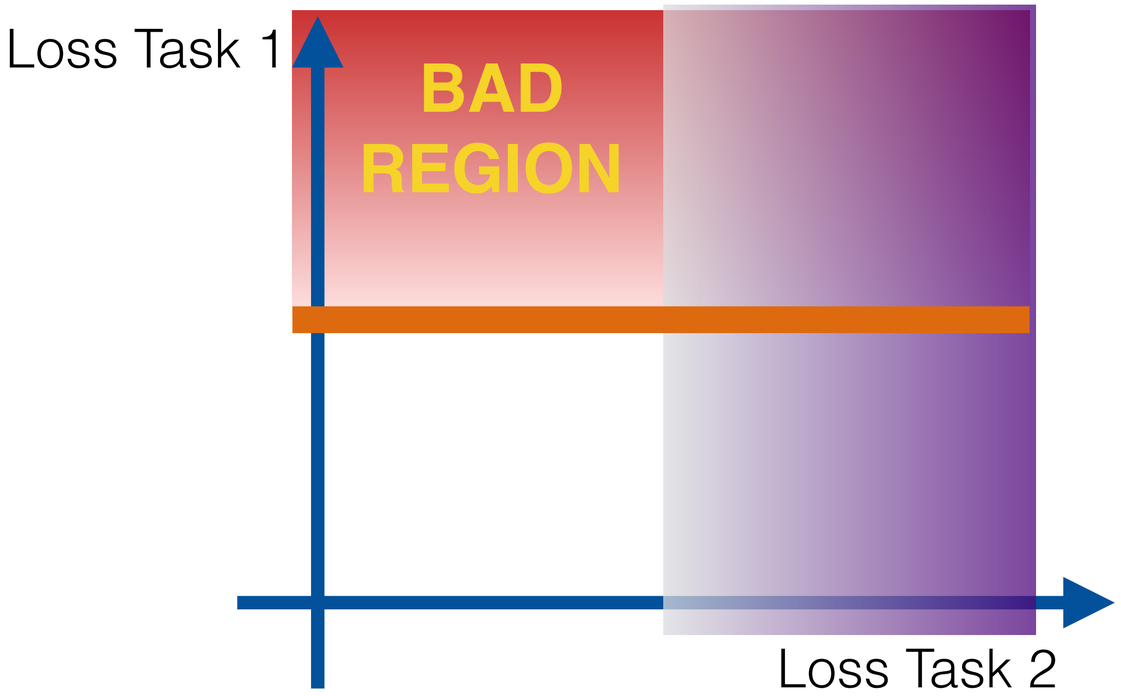
We can detect loss increase by checking the gradients:



D. Lopez-Paz et al. “GEM for continual learning”, NIPS 2017

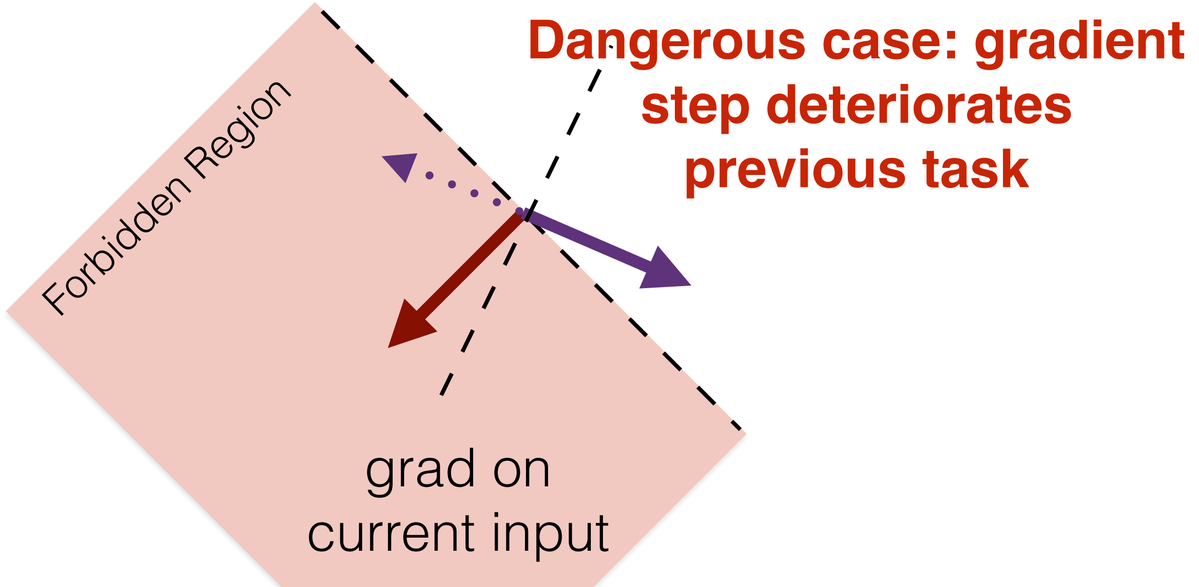
Gradient Episodic Memory (GEM)

Assume we have already learned from examples of the first task, and we are now observing examples from task 2. We want to make sure we do not un-learn task 1.



Gradient=direction of maximum increase of the loss

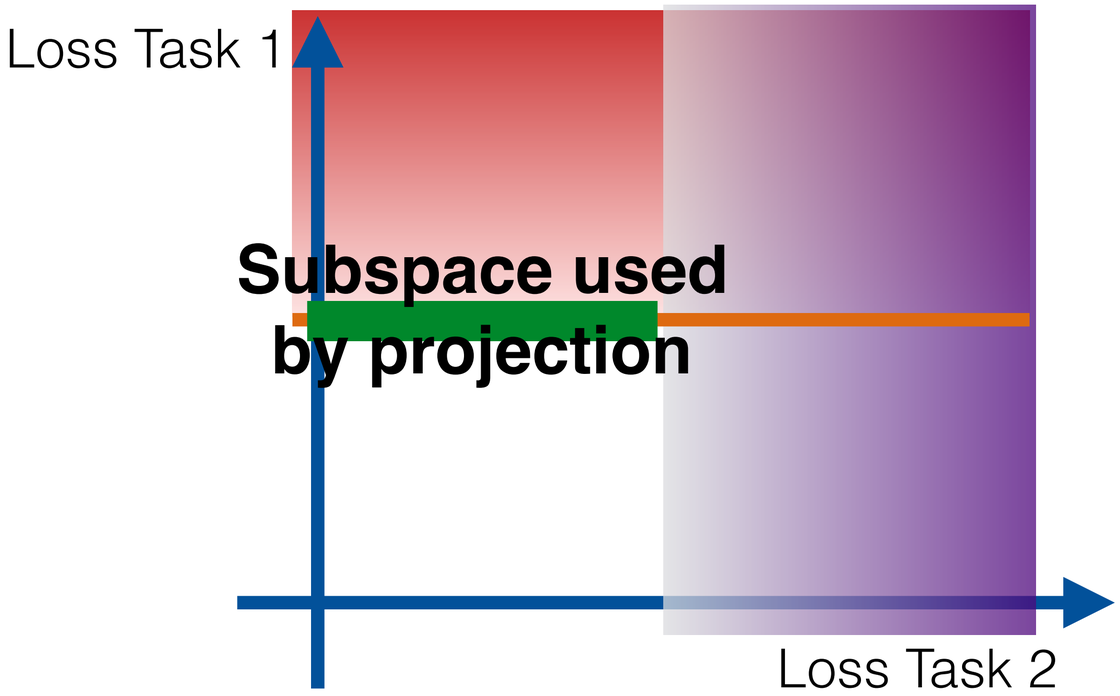
We can detect loss increase by checking the gradients:



D. Lopez-Paz et al. “GEM for continual learning”, NIPS 2017

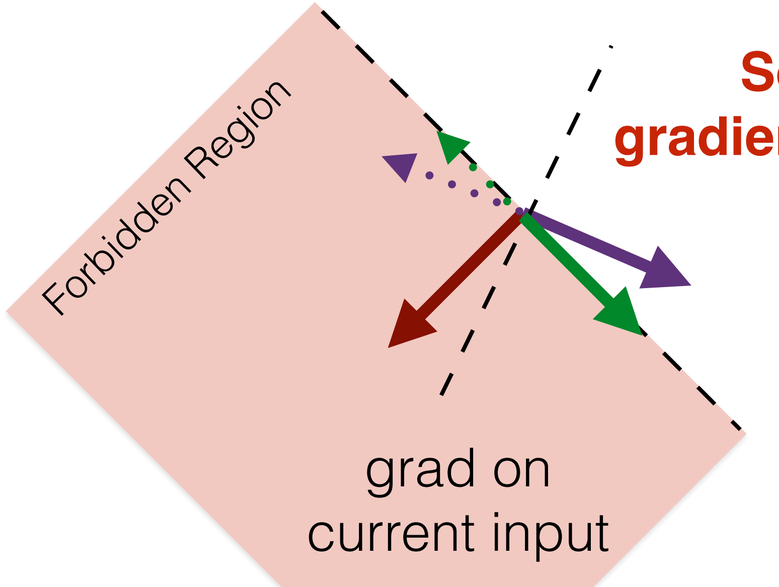
Gradient Episodic Memory (GEM)

Assume we have already learned from examples of the first task, and we are now observing examples from task 2. We want to make sure we do not un-learn task 1.



Gradient=direction of maximum increase of the loss

We can detect loss increase by checking the gradients:



D. Lopez-Paz et al. “GEM for continual learning”, NIPS 2017

GEM Algorithm

- for task k = 1…T do:
  - for every input mini-batch do: # only one pass
    - compute the gradient of the loss w.r.t. parameters at the current mini-batch: g
    - compute the gradient of the loss w.r.t. parameters using examples in the memory belonging to task i: g_i
    - if there exist some i<k s.t. g * g_i < 0, then:
      - project gradient g
    - update parameters & update memory

Learning problem: $\arg\min_{\theta} l(\theta)$

$$\text{s.t. } g \cdot g_i \geq 0, \forall i \in \{1, ..., k-1\}$$

D. Lopez-Paz et al. “GEM for continual learning”, NIPS 2017

GEM Algorithm

- for task k = 1…T do:
  - for every input mini-batch do: # only one pass
    - compute the gradient of the loss w.r.t. parameters at the current mini-batch: g
    - compute the gradient of the loss w.r.t. parameters using examples in the memory belonging to task i: g_i
    - if there exist some i<k s.t. g * g_i < 0, then:
      - project gradient g

Projection step is inefficient:

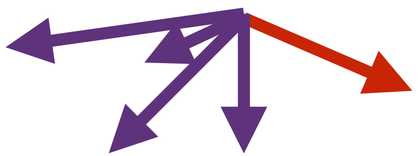
- takes memory to store g_i, i = 1..k-1
- it's computationally expensive to project on a finitely generated cone (unless k=2)

A-GEM Algorithm

A(veraged)-GEM is a very simple and efficient variant of GEM where there is only one constraint given by the average gradient over the memory examples.

Example: model has seen 4 tasks.
Memory gradients are purple arrows. Current gradient is red arrow.

GEM: would incur in 3 constraint violations.



A-GEM optimizes for the best average accuracy.
GEM minimizes worst-case forgetting.

A-GEM: incurs in 1 constraint violation.

A-GEM is less likely to incur in violations.

A. Chaudhry et al. “Efficient Life-Long Learning with A-GEM”, arXiv 2018

A-GEM Efficiency

- memory: no need to store gradient w.r.t. parameters for each previous task, just one additional gradient vector.
- computation:
  - less constraint violations
  - projection step costs just a dot product
  - gradients can be computed on a random subset of the memory

A. Chaudhry et al. “Efficient Life-Long Learning with A-GEM”, arXiv 2018

meets

GEM loss: $$\arg\min_{\theta} l(\theta)$$
$$\text{s.t. } g \cdot g_i \geq 0, \forall i \in \{1, ..., k-1\}$$

Lagrangian of GEM loss: $$\arg\min_{\theta} l(\theta) - \lambda \sum_i g \cdot g_i$$

MAML ~ REPTILE: $$\arg\min_{\theta} \mathop{\mathbb{E}}_{B_1,...,B_s} [\sum_{i=1}^{s} [l(B_i, \theta) - \lambda \sum_{j}^{i-1} g_j \cdot g_i]]$$

M. Riemer et al. “Learning to learn without forgetting by maximizing transfer and minimizing interference”, arXiv 2018

A. Nichol et al. “Reptile: a scalable metalearning algorithm”, arXiv 2018

C. Finn et al. “Model-agnostic meta-learning for fast adaptation of deep networks”, ICML 2017

meets

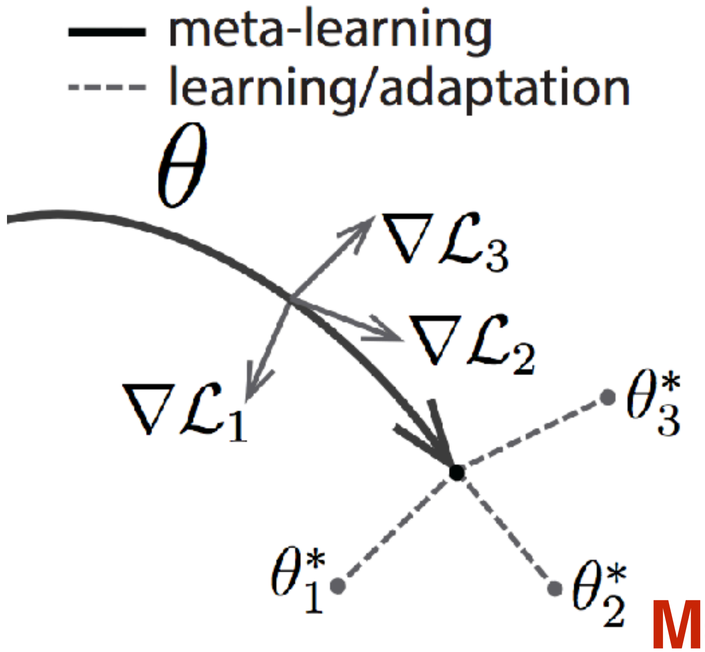
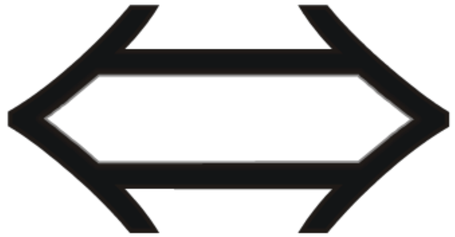


figure credit: MAML paper



Meta-learning ~ CL, once we factor in sample efficiency.

Being able to remember past tasks is key to solve quickly related tasks in the future.

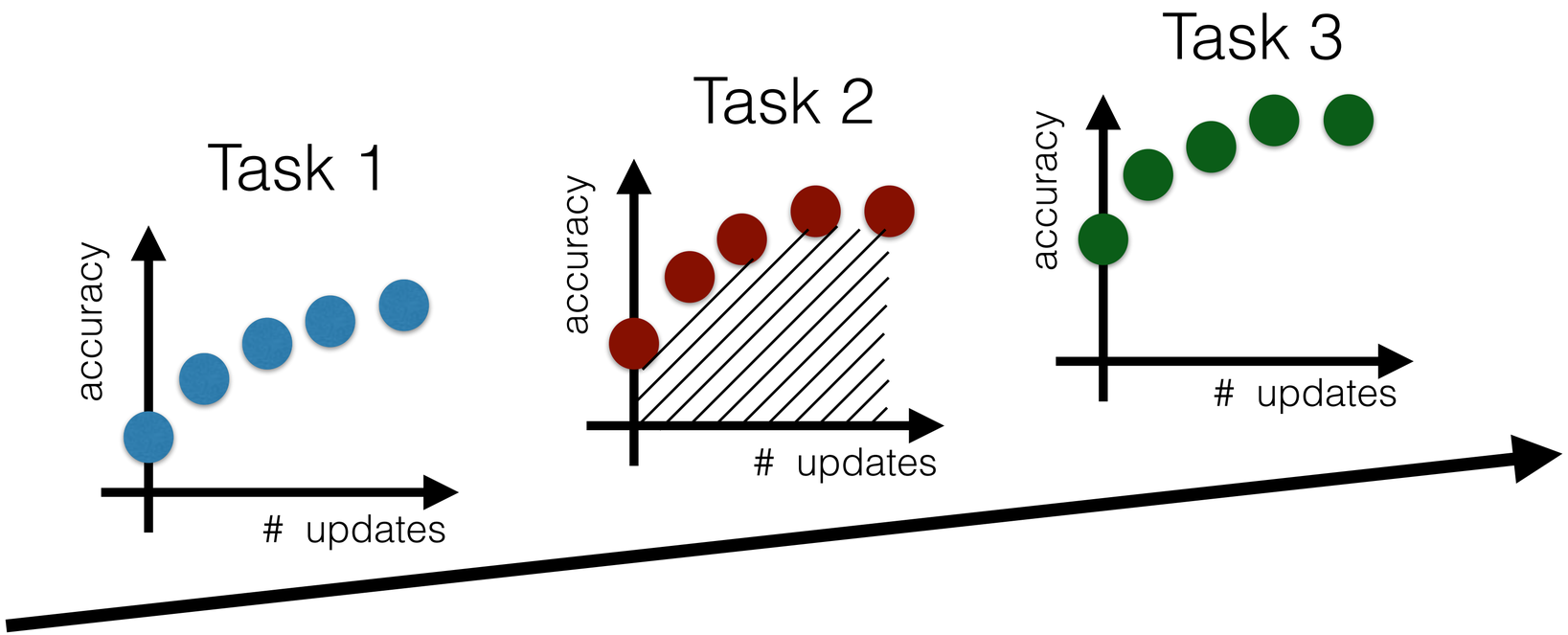
M. Riemer et al. “Learning to learn without forgetting by maximizing transfer and minimizing interference”, arXiv 2018

A. Nichol et al. “Reptile: a scalable metalearning algorithm”, arXiv 2018

C. Finn et al. “Model-agnostic meta-learning for fast adaptation of deep networks”, ICML 2017

Measuring Sample Efficiency

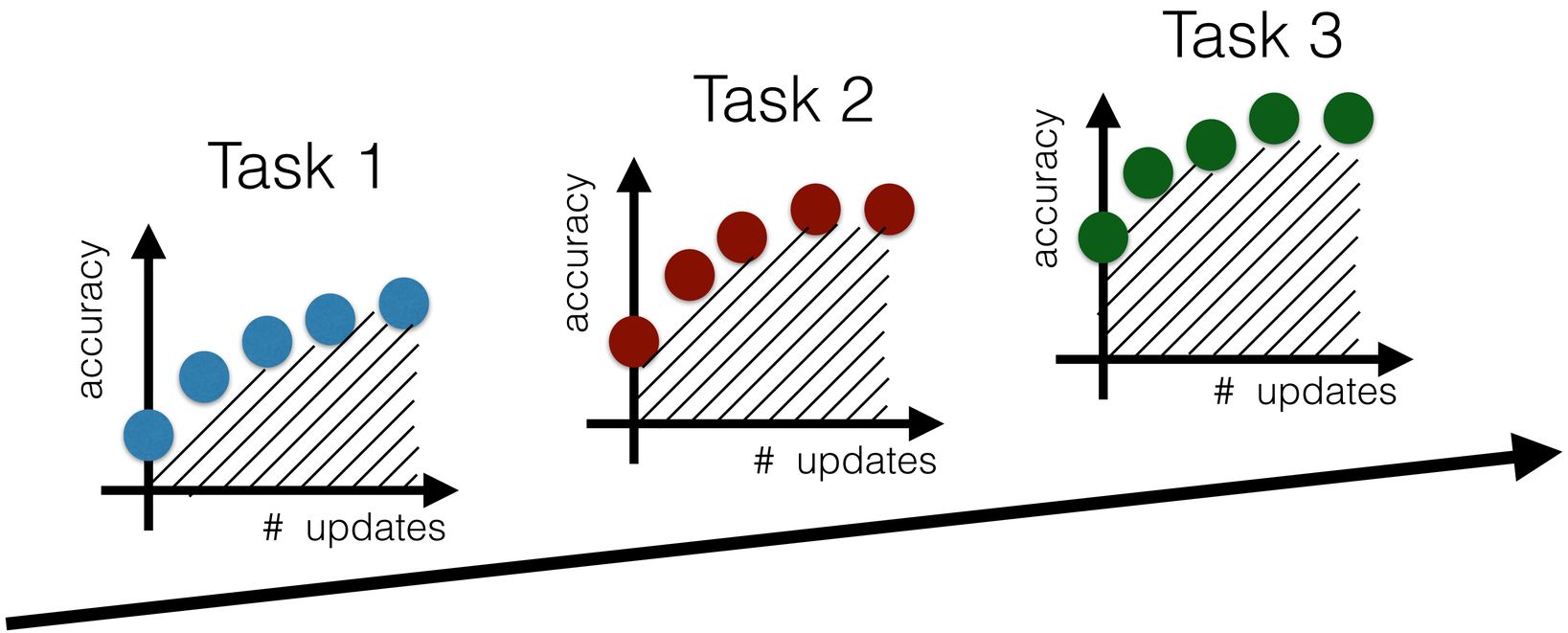
Area under the convergence curve describes how quickly model learns and the overall quality.



A. Chaudhry et al. “Efficient Life-Long Learning with A-GEM”, arXiv 2018

Measuring Sample Efficiency

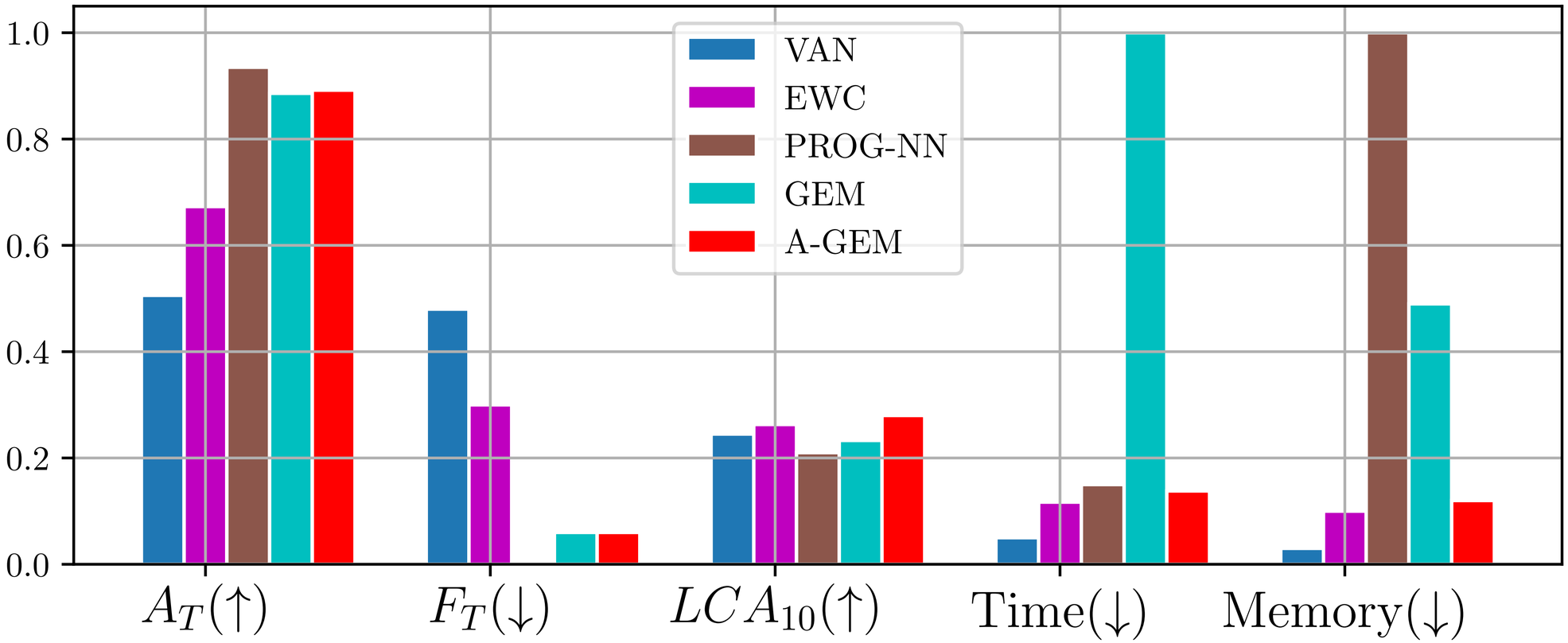
Learning Curve Area @k = average across tasks of the area under the curve up to mini-batch k measures how quickly learner adapts to first k minibatches.



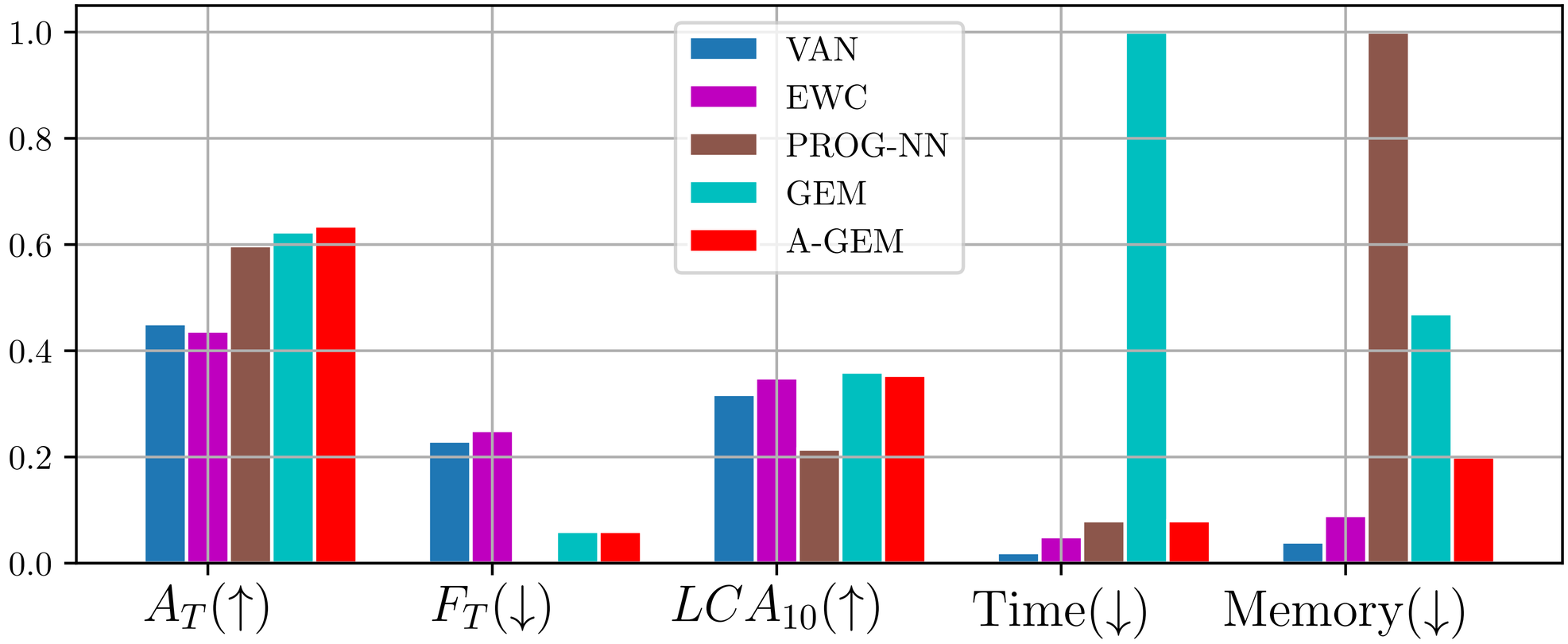
A. Chaudhry et al. “Efficient Life-Long Learning with A-GEM”, arXiv 2018

Experiments: Permuted MNIST

A-GEM has good trade-off between average accuracy VS compute/memory cost.



Experiments: Split CIFAR-100

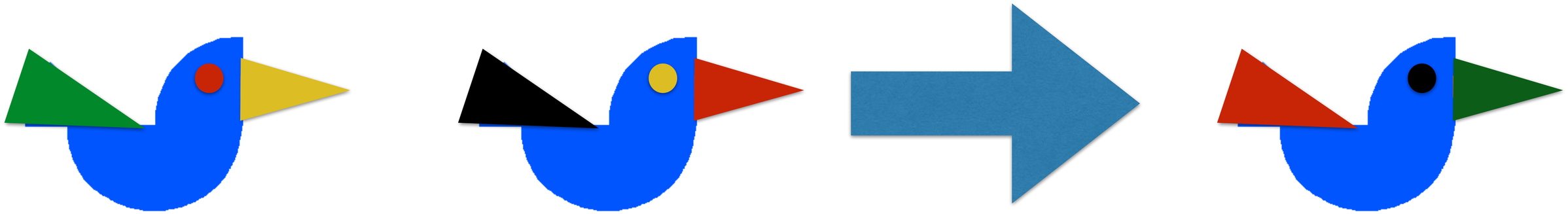


Faster Learning with Task Descriptors

- Task descriptor specifies the learning task.
- If task descriptor is compositional, it enables good 0 shot performance.

Classes of birds seen at training time

0-shot recognition at test time



input class descriptor

beak:yellow
tail:green
...

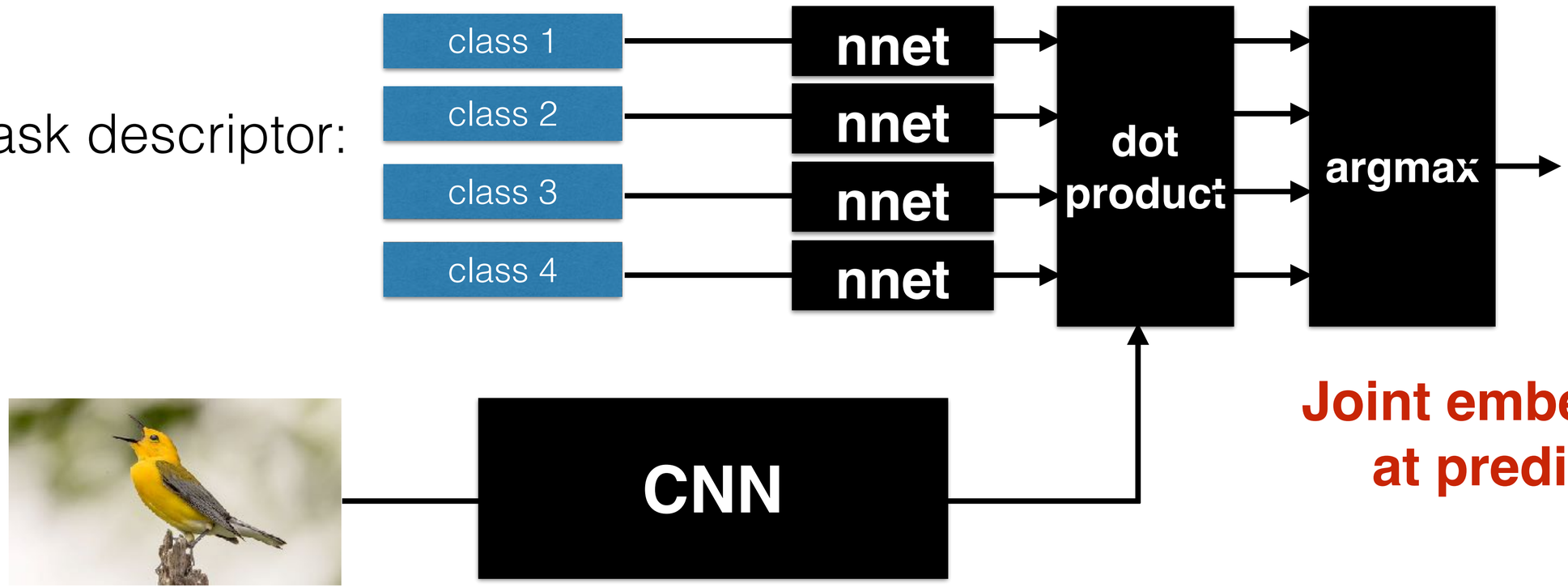
beak:red
tail:black
...

beak:green
tail:red
...

Faster Learning with Task Descriptors

Class descriptor of “swallow”: c_i = {beak: short, tail: split, size: medium/small, …}
task descriptor: [c_i, c_j, c_k]

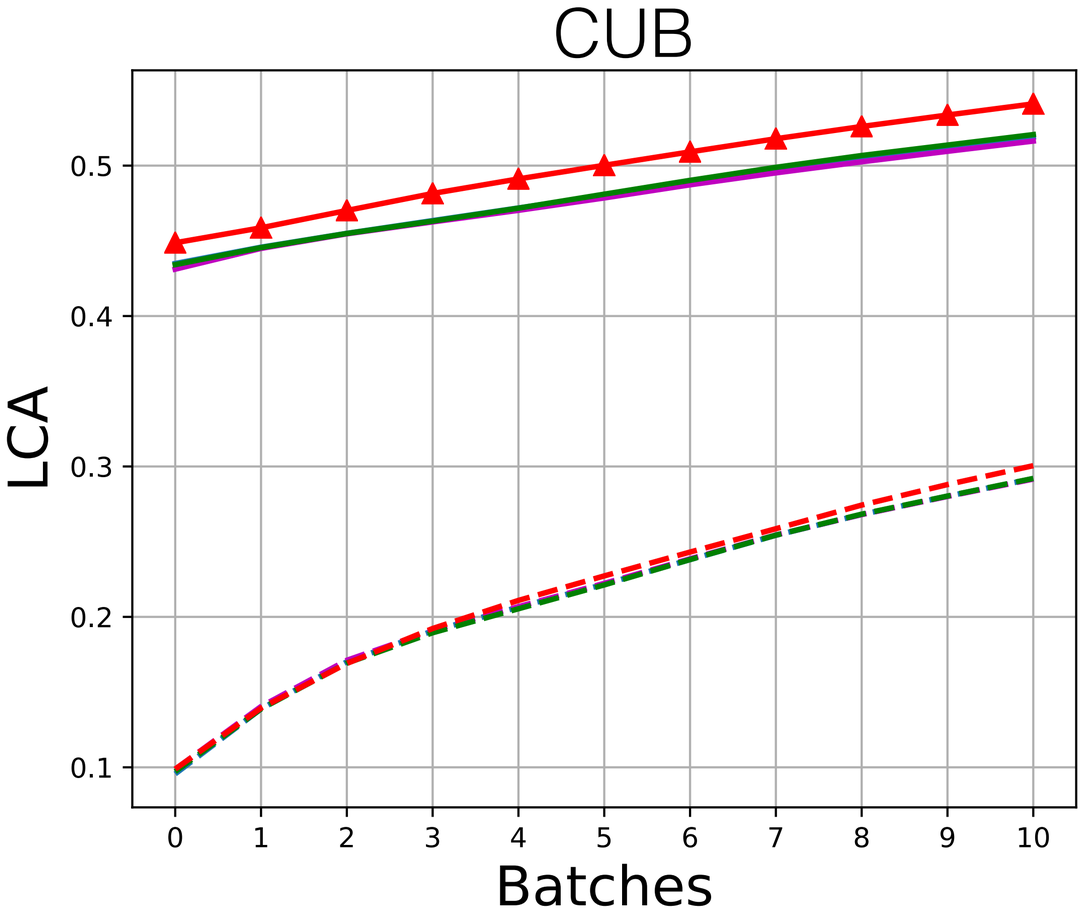
Task descriptor:

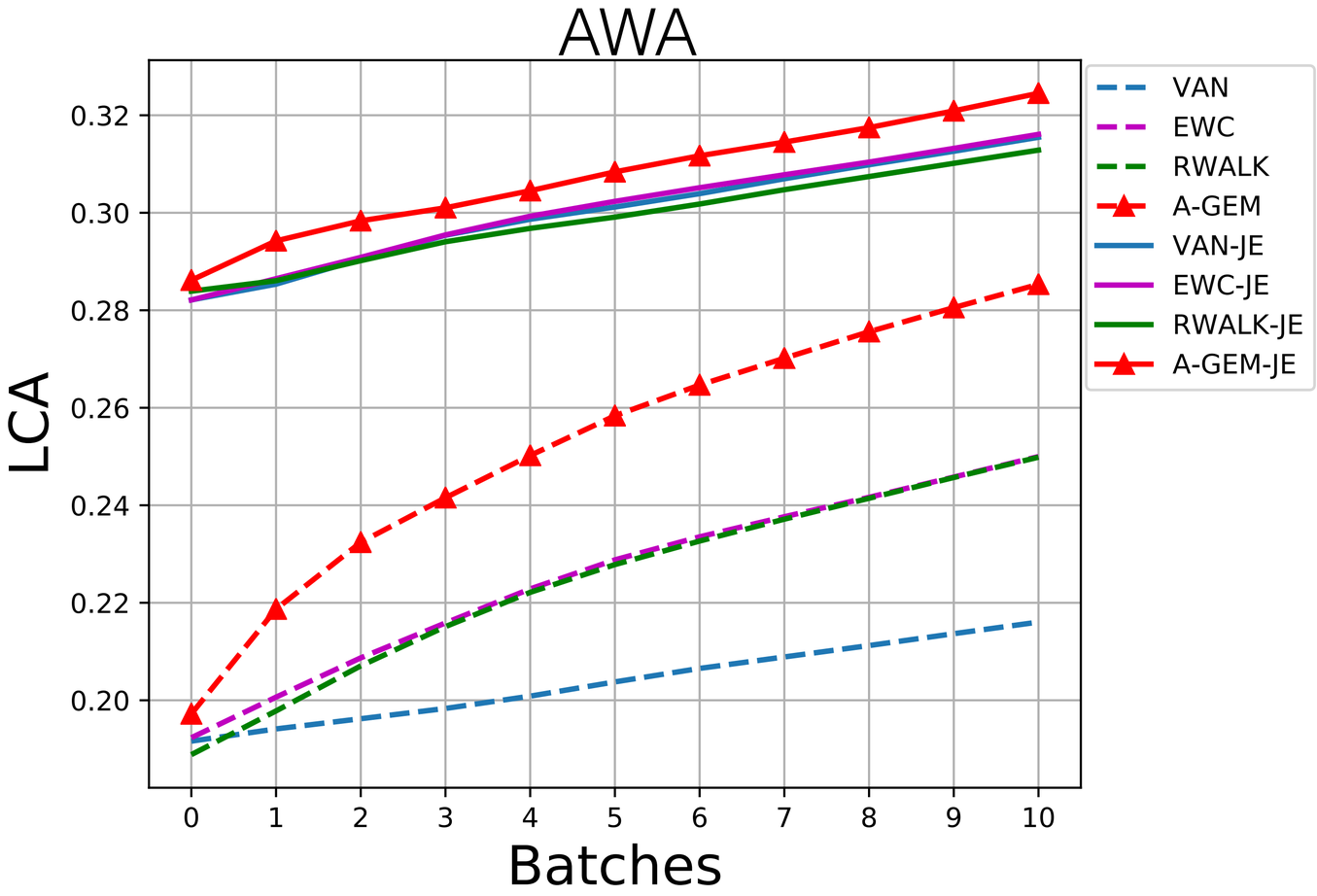


Joint embedding model at prediction time.

A. Chaudhry et al. “Efficient Life-Long Learning with A-GEM”, arXiv 2018

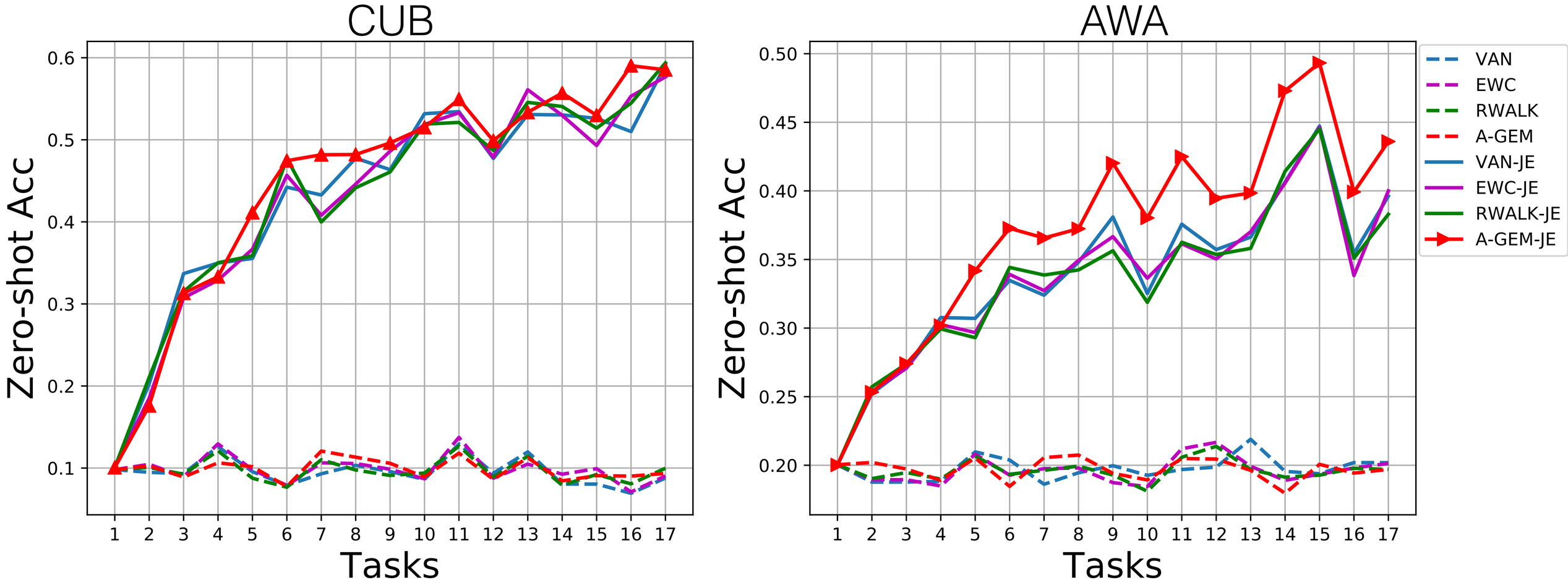
Experiments: Split CUB/AWA





Task descriptors greatly improve learning speed.

Experiments: Split CUB/AWA



0-shot performance gets better as we learn, when using task descriptors.

Conclusions

- CL: avoiding catastrophic forgetting as a mean to promote transfer.
  - how to make sure accrued knowledge is transferable?
- Efficiency as driver of methodology and algorithm design. Three axes:
  - sample,
  - memory and
  - computational efficiency.

What's new?

- New CL learning & evaluation protocol.
- New metric to measure speed of learning.
- A-GEM: a new efficient algorithm.
- Experiments showing good few-shot performance using task descriptors.
- GEM intimately related to meta-learning algorithms (Riemer et al. 2018).

Future Challenges

- Consolidating methodology.
- Reaching consensus on metrics.
- Reaching consensus on representative tasks to evaluate CL methods.
- Finding actual applications to drive progress in the field.

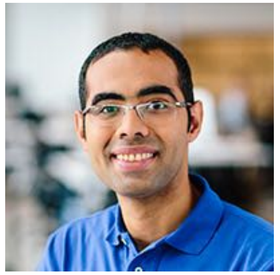
- Designing simpler and more efficient models.
- Integrating other learning components.

THANK YOU

Michelangelo at 87 — *“I am still learning”*

A. Chaudhry et al. “Efficient Life-Long Learning with A-GEM”, arXiv 2018

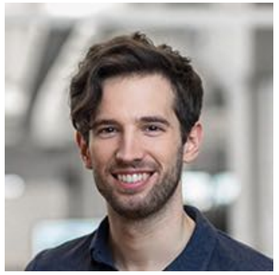
Arslan Chaudhry



Mohamed Elhoseiny

Marcus Rohrbach

D. Lopez-Paz et al. “GEM for continual learning”, NIPS 2017



David Lopez-Paz