Continual Learning in the Era of (Very) Large Models

Marc’Aurelio Ranzato
Google DeepMind
ranzato@google.com

Stanford MS&E 338 Reinforcement Learning, 8 May 2023
1 What is Continual Learning?

“The real world is characterized by a seemingly unlimited degree of detail and regularity... Humans, during the course of their lives, continually grasp ever more complicated concepts and exhibit ever more intricate behaviors. The world supports this continual learning process by providing a never-ending multitude of complexities and regularities ...”

Continual Learning to better understand human learning.
Why Continual Learning?

3 Motivating Examples

3.1 Recommendation systems

We can frame a prototypical recommendation system in terms of the above formulation. In particular, let $X_t$ encode a combination of features pertaining to a product and a user, and let $Y_t$ be a binary label indicating whether or not the user will later express that they “like” the product. Suppose the recommendation system serves users of a Web service. In such a context, data arrives continuously and accumulates. And the statistics of this data changes over time, perhaps driven by trends in fashion or emerging news or consumer sentiment. A typical workflow involves applying the above steps periodically, starting from scratch each time. For example, a production system could train a new model at the end of each month on the most recent two years of data. In this way, continual learning is reduced to the standard machine learning paradigm.

3.2 Large language models

Today’s large language models assume an autoregressive structure, typically based on transformer architectures. These models work with tokenized text, which is essentially text gathered from the Web and turned into time series of symbols called tokens. Each token might represent a word or punctuation. A typical LLM takes as its input $X_t$, a sequence of, say, one thousand consecutive tokens, and this sequence is labeled with...
Reality Check: Applications Using Continual Learning
Intelligence of humans and machines is different because they have:

- Different sensory inputs
- Different constraints
- Different goals

Hypotheses:

i) Maybe machines need not to be continual learners after all!

ii) Maybe large-scale models are already continual. Is (lots of) data enough to be continual?

iii) Maybe machines would work better if they were continual learners, but it just so happens that our current continual learning methods are not good enough.
(Machine) Continual Learning Today

- A lot of different settings. E.g.:
  - memory restrictions
  - task boundaries
  - type of supervision
  - choice of metrics
  - type of non-stationarity
  - methodology (what does it mean to have one life only?)
- Unclear what is the goal, what matters and what actually works
  - Experience replay (from RL) is amongst the most robust and effective method.
Currently there is not much CL in large-scale modeling.
  ○ At best, one step adaptation.
• Unclear alignment between CL research and large-scale model research.

This lecture is about discussing opportunities for CL (and RL) in large-scale learning moving forward.

Example of recent papers on CL in a large-scale setting:
Ramasesh et al. Effects of scale on catastrophic forgetting in neural networks ICLR 2022
Scialom et al. Finetuned language-models are continual learners EMNLP 2022
Liska et al. Streaming QA ICML 2022
Agenda

- Prologue [10min]
- **Continual Learning for Large-Scale Learning: Why, What & How [15min]**
- Sandboxes for Supervised CL [20min]
- Toy approach to CL [15min]
- Discussion [20min]

References

- Bornschein et al. **NEVIS'22 Benchmark** (arXiv 2022, in submission)
- Veniat et al. **Efficient Continual Learning with Modular Networks and Task Driven Priors** (ICLR 2021)
Why Continual Learning?
The typical life cycle of a ML practitioner:

- Look at the data
- Borrow ideas, code, model parameters, etc.
- Try out a few things
- and a few more..
The typical life cycle of a ML practitioner:

There is a hierarchy of continual learning problems.

Continual learning is already happening in ML, but it is not efficient, it is not automated.
The typical life cycle of the swarm model:

(Semi-Automated) Continual Learner is social too!
Continual Learning for Machine Learning

Assumptions
- We (both humans and machines) live in time.
- Changes (e.g., data, constraints) happen over time.
  - Hypothesis: Nobody will have ever learned enough! ← to be verified.
- Efficiency matters as much as efficacy.

What is continual learning:
- Continual learning is about learning over time and leveraging past learnings to improve future learnings in terms of both efficacy and efficiency.

Current Machine Learning and Large-Scale Learning are already continual, but inefficiently so.
Continual Learning for Large-Scale Learning

One (some) of these models are very large-scale.
The current state of affairs

Current large-scale systems are accurate but costly and inefficient. Could this be an opportunity for CL?
The continual learning dream...
Leverage what we learnt before → improve efficiency

A single model, shared by everybody, that evolves over time to become more efficient at learning the next thing.
Hard Questions

- What abstraction to use for continual learning?
  - What does cross-validation mean in this context?
  - What data can be useful to study this problem in a controlled setting?
- How to characterize a swarm model?
- How to measure performance?

It is often a good idea to start from a concrete application or problem, and derive from there abstractions. Judgement is required to figure out the good level of coarseness of the abstraction. In our case, we want to build a large-scale system that is effective but also efficient at both training and testing time.
What Continual Learning?
Lecture 2: Non-stationary linear regression

\[ \max_{\pi, \nu} \lim_{T \to \infty} \inf \frac{1}{T} \mathbb{E} \left[ \sum_{t=0}^{T-1} R_{t+1} \right] \]

s.t. flops \leq C
Catastrophic forgetting is a symptom of poor learning, not an objective per se.
Lecture 2: Non-stationary linear regression

\[
\max_{\pi, \nu} \liminf_{T \to \infty} \frac{1}{T} \mathbb{E} \left[ \sum_{t=0}^{T-1} R_{t+1} \right]
\]

s.t. flops \leq C

Constraints are critical!
On Intelligence

Intelligence must arise when there are suitable constraints.

What are the constraints?
• number of examples?
• compute?
• memory?
• time?
• ?

Continual learning is an instance of multi-objective learning.
Don’t tell me which method is most accurate.. but which one strikes the best trade-off between efficiency and accuracy.

Legg et al. Universal Intelligence arXiv 2007
How Continual Learning?
Desiderata

- No human in the loop
- Scalable
- Distributed
- Efficient to use
- Efficient to update, e.g. enables incremental learning
- Self-Improving

} Never-ending learning
Ingredients

- **Transfer learning** to cope with constraints:
  - time
  - compute
  - number of examples needed

- **Modular**
  - Enables efficient incremental learning
  - Enables efficient inference
  - Enables compositional generalization

- **Distributed**
  - Enables learning of swarms
  - Enables scaling
  - Robustness

- **Meta-Learning**
  - Operates at coarser time scale
  - Searches over optimizers, architectures, etc.
Ingredients

- **Transfer learning** to cope with constraints:
  - time
  - compute
  - number of examples needed

- **Modular**
  - Enables efficient incremental learning
  - Enables efficient inference
  - Enables compositional generalization

- **Distributed**
  - Enables learning of swarms
  - Enables scaling
  - Robustness

- **Meta-Learning**
  - Operates at coarser time scale
  - Searches over optimizers, architectures, etc.

Constraints

- Time
- Communication
- Latency
- Privacy
- ...

What is not a constraint:
- memory (disk)

Not so much of a constraint:
- compute
- data
Colab Demo on Modularity
Hard Questions

- Efficiency hinges on knowledge transfer: what is knowledge?
- Do “universal representations” exist? Can data be enough at some point?
- What are the modules?
- How to train a modular system s.t. modules interface well and yet communication amongst them is scarce?
- How to learn efficient meta-learners?
- How to enable self-improvement?

The answer to these questions require a rather interdisciplinary approach. We may draw inspiration from neuroscience (knowledge characterization), empirical analysis (universal representation), algorithmic development (modularity), systems (distributed computing), RL (self-improvement), etc.
Figure 4: Mental picture for supervised learning (SL), continual supervised learning (CSL), reinforcement learning (RL), and continual reinforcement learning (CRL).
Figure 1: The above depicts how one may slot CL into existing frameworks.

Figure 2: The above depicts how one may view CL as being a core to all existing ML frameworks.
Relation of CL to Other Fields: This Lecture

Most of ML is continual! CL needs input from sub-fields like meta-learning and auto-ml. Vice-versa CL can lift these subfields and make them more practical.
How about RL?

- Natural setting for CL akin to human learning
- Not so natural for large-scale learning
- Perhaps controversial: Many (all?) fundamental questions might be answered without making the next observation depend on the agent’s action…

Even in the supervised setting, there are lots of opportunities for RL research:

- How to use resources when horizon is infinite?
- How to optimize with non-differentiable constraints?
- How to efficiently search and self-improve?

Overall, there are plenty of opportunities for RL researchers to impact large-scale continual learning.
Agenda

- Prologue [10min]
- Continual Learning for Large-Scale Learning: Why, What & How [15min]
- **Sandbox**es for Supervised CL [20min]
- Toy approach to CL [15min]
- Discussion [20min]

References

- Bornschein et al. [NEVIS'22 Benchmark](arXiv 2022, in submission)
- Veniat et al. [Efficient Continual Learning with Modular Networks and Task Driven Priors](ICLR 2021)
Designing Benchmarks

Benchmark = \{\text{data, metrics, methodology}\} + \text{codebase, baselines}...

Progress in the field is driven by the choice of benchmarks.
If we know what problem we want to solve (efficient large-scale learning), then we can abstract it into a suitable benchmark.

Properties useful for large-scale learning:

- rules out methods that do not transfer
- rules out methods that do not scale
- useful to assess how large-scale models can operate over time
- construction is method agnostic
Designing Benchmarks

Benchmark = \{\text{data, metrics, methodology}\} + \text{codebase, baselines}...

Progress in the field is driven by the choice of benchmarks. If we know what problem we want to solve (efficient large-scale learning), then we can abstract it into a suitable benchmark.

Properties useful for large-scale learning:
- rules out methods that do not transfer
- rules out methods that do not scale
- useful to assess how large-scale models can operate over time
- construction is method agnostic

References
- NEVIS’22 Benchmark (arXiv 2022, in submission)
- Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
Continual Supervised Learning

- Each dataset is a task, with its own input/output distribution.
- Tasks may relate to each other, but in unknown ways.
- Task ids are given to the learner both at training and test time.
- At test time, learner can be asked to perform any previous task.

CTrL is a suite of streams probing (supervised) continual learners across same basic axes.

Veniat et al. [Efficient Continual Learning with Modular Networks and Task Driven Priors](ICLR 2021)
Is the learner capable of figuring out $T_4 = T_1$ and that it should directly transfer knowledge?

Measure accuracy of $T_4$ when:

a) Model is trained on entire stream
b) Model is trained just on $T_4$

Ideally: $A(T_4 | T_1, T_2, T_3, T_4) - A(T_4) >> 0$

Veniat et al. *Efficient Continual Learning with Modular Networks and Task Driven Priors* (ICLR 2021)
Continual Transfer Learning: Knowledge Update

Is the learner capable of figuring out it is best to update knowledge of first task even though T4 == T1?

Measure accuracy of T4 when:

a) Model is trained on entire stream

b) Model is trained just on T4

 ideally:

\[ A(T4 \mid T1, T2, T3, T4) - A(T4) = 0 \]
If T4 is related to T1, can the learner transfer knowledge?

Simple measure accuracy of T4 when:

a) Model is trained on entire stream
b) Model is trained just on T4

Ideally:

\[ A(T4 | T1, T2, T3, T4) - A(T4) >> 0 \]
Are previous unrelated tasks interfering with learning of the current task?

Simple measure accuracy of T4 when:

a) Model is trained on entire stream
b) Model is trained just on T4

\[ A(T4 | T1, T2, T3, T4) - A(T4) = 0 \]

ideally:

Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
CTrL: Holistic Performance Assessment

Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
CTrL: Pros & Cons

Pros
● Intuitive
● Useful to rule out bad methods
● Not only a benchmark, it’s an approach to benchmark construction

Cons
● Toy. Datasets are CIFAR, MNIST, SVHN, etc.
● In practice there might be more interesting dimensions of transfer to consider
● ... Too many metrics to track?

Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
The Never Ending VIIsual classification Stream (NEVIS’22)

Bornschein et al. NEVIS’22 Benchmark (arXiv 2022, in submission)
Objective

Want to simulate life of a large-scale never-ending learning system which is exposed to new tasks over time.

Benchmark should have:

- More realistic stream
- Simpler metric
- Explicit cross-validation methodology
- A large-scale benchmark for large-scale learning, still enabling quick prototyping

Bornschein et al. NEVIS'22 Benchmark (arXiv 2022, in submission)
NEVIS Construction
3 decades: 1992 – 2021

Step 1: Sort proceedings chronologically.
Step 2: Sample 90 papers per year.
Step 3: Extract image classification tasks, if any, from each paper.
Step 4: Filter out tasks. Criteria: data availability, license type, duplicates, etc.
NEVIS Benchmark

106 tasks in total
~8 million images
Many domains

Number of datasets

Data types per year

- xray
- counting
- quality
- medical
- scene
- ocr
- shape
- object
- texture
- face
- satellite
NEVIS Benchmark

106 task in total
~8 million images
Many domains

- Simple
- Reproducible
- Unbiased to task selection
- Largish-scale...
Example of tasks in NEVIS

1. Aberdeen face database
2. Magellan Venus Volcanoes
3. Brodatz
4. LandSat UCI
5. Olivetti Face Dataset
6. COIL 20
7. MPEG-7
8. CVC-MUSCIMA
9. KTH-TIPS
10. UIUC texture
11. NORB
12. KTH-TIPS2-a
13. IAPRTC-12
14. sketch
15. ...
16. ...
17. 100. Pneumonia Chest X-ray
18. 101. Oxford Flowers 102
19. 102. Synthetic COVID-19 Chest X-ray
20. 103. ImageNet
21. 104. NIH Chest X-ray
22. 105. covid-19 x-ray
23. 106. Tuberculosis

Bornschein et al. NEVIS'22 Benchmark (arXiv 2022, in submission)
NEVIS dataset | Metrics

- We evaluate efficacy as the **average error**:

\[ \mathcal{E} = \frac{1}{N} \sum_{i=1}^{N} e_i \]

- We evaluate efficiency via cumulative FLOPS. This includes the cost of hyper-parameter search.

\[ c\text{FLOP} = \sum_{i=1}^{N} \text{FLOP}_i \]

Bornschein et al. [NEVIS'22 Benchmark](arXiv 2022, in submission)
Bornschein et al. [NEVIS'22 Benchmark](https://arxiv.org/2022, in submission)
Evaluation Protocol

Goal: Assess ability to efficiently adapt to future tasks.

Use meta-train to develop learner/meta-learner. This can be replayed as many times as desired.

Meta-test is used only for final evaluation. It is not possible to access future tasks from meta-test. Agent/Learner has only 1 life in meta-test.
Results
Baselines: Independent training per task

Task Stream

Initialize

Train/Finetune

No knowledge transfer

Task 1

Task 2

Task 3
Baselines: **Finetune** from the most relevant

Initialize from the most related task selected automatically

Train/Finetune

Task 1

Task 2

Task 3

Bornschein et al. [NEVIS’22 Benchmark](arXiv 2022, in submission)
Baselines: Finetune from best encountered model

Construct features using previous models

Task 3 data

Pick the one with the highest kNN accuracy

Task 3

Veniat et al. “Efficient continual learning with modular networks and task-driven priors” ICLR 2021
Baselines: **Multi-task training**

**Initialize**

```
<table>
<thead>
<tr>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>head 1</td>
<td>head 2</td>
<td>head 3</td>
</tr>
</tbody>
</table>
```

**Train/Finetune**

```
<table>
<thead>
<tr>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>head 1</td>
<td>head 2</td>
<td>head 3</td>
</tr>
</tbody>
</table>
```
Pareto Fronts

Curves generated by varying the number of h.p. configurations and the number of weight updates per task.

Each point represents about 16 (# h.p. configs per task) * 107 (# tasks) ~ 1700 experiments!

Alayrac et al. “Flamingo: a visual language model for few-shot learning” NeurIPS 2022
Wall clock on a single GPU

This is the region we’ll be focussing on.

4 days

112 days
Pareto Fronts

Finetuning from the most relevant provides the best Pareto front

Bornschein et al. NEVIS’22 Benchmark (arXiv 2022, in submission)
Pareto Fronts

Significant improvement brought by finetuning

Bornschein et al. NEVIS'22 Benchmark (arXiv 2022, in submission)
Pareto Fronts

Best results with pre-trained models, but these cost 10x more

Bornschein et al. *NEVIS’22 Benchmark* (arXiv 2022, in submission)
Finetuning pretrained models improves performance
Finetuning pretrained models improves performance

Hypothesis: Nobody will have ever learned enough...
This experiment validated this hypothesis with the current choice of large models and datasets.
Finetuning from the most relevant task

$A \rightarrow B = \text{"B fine-tuned from A"}$

Colors correspond to domains.
Several hubs are formed
ImageNet: the biggest hub
Long chains are formed
Finetuning chain of PT+FT

- Shorter chains
- Lots of nodes at depth > 1
- A few hubs
- Nice clustering by domain
NEVIS’22 is a benchmark for measuring the performance of algorithms in the field of continual learning. Please see the accompanying paper for more details.

Within this Python package, we provide three components,

1. Library code to download and post-process datasets that are not available within tfds, so that the stream used in the paper can be replicated.
2. A package to combine the NEVIS’22 datasets into a stream, and robustly evaluate learners using the evaluation protocol proposed in the NEVIS’22 paper.
3. Baseline learners implemented in JAX and PyTorch. The JAX learners are identical to the learners used for the figures in the paper, the PyTorch learners are provided for example purposes.

https://github.com/deepmind/dm_nevis

https://arxiv.org/abs/2211.11747
Conclusions

- NEVIS’22 complements CTrL.
- Performance measured in terms of efficiency/efficacy trade-off.
- Large-scale models exhibit better performance when adapted over time.
- Eventually the finetuning approach creates a pool of models.

Observation: Finetuning generates a (very naïve) modular system.

Q.: How to improve efficiency by sharing not just initialization but also parameters across these modules?

Even when aiming at large-scale learning, we need to be able to prototype quickly, and run sanity checks. CTrL is great for this purpose. NEVIS’22 offers a good middle ground between toy scale and large-scale.
Agenda

- Prologue [10min]
- Continual Learning for Large-Scale Learning: Why, What & How [15min]
- Sandboxes for Supervised CL [20min]
- Toy approach to CL [15min]
- Discussion [20min]

References
- Bornschein et al. NEVIS’22 Benchmark (arXiv 2022, in submission)
- Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
Designing a Modular Approach

- Grow capacity over time to retain plasticity.
- Decouple overall model size with amount of parameters and compute needed to perform any single task.
- Scalable
Modular network at time $t$

Existing pool of modules

Mixture of experts with gating performed at the task level.

Veniat et al. **Efficient Continual Learning with Modular Networks and Task Driven Priors** (ICLR 2021)
Step 1: Receive new task

Existing pool of modules

- Layer 3 modules
- Layer 2 modules
- Layer 1 modules

Data of new task

Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
Step 2: Module retrieval

Retrieve most relevant modules at each layer. E.g.: select modules of networks trained on most related past tasks. The retrieval set is a (data-driven) prior.

Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
Step 3: Perturb & Search

Existing pool of modules

Layer 3 modules

Layer 2 modules

Layer 1 modules

Data of new task

Module retrieval

Train in parallel k variants and select the best

One could also use REINFORCE to train the k variants all at once.

Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
Step 4: Pool expansion

Existing pool of modules

Module retrieval

Data of new task

Arch. Search

New pool of modules

Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
If tasks relate to each other, an “older” network can re-use more modules than a “younger” one. With age, the network grows less.

Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
Results on CTRL
Results on $S_{long}$

MNTDP achieves highest average accuracy while growing sub-linearly in memory.
Conclusions

- MNTDP is a robust and simple way to grow a modular network.
- Size of search space defines efficiency/efficacy trade-off.
- General idea:
  - Retrieve most similar modules
  - Perturb & learn
  - Add to the existing pool the newly trained modules
- Because of growth, model is not going to lose plasticity over time.

Open questions
- Scaling up
- Efficient architecture search
- How to learn a good initial set of modules

Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
Agenda

● Prologue [10min]
● Continual Learning for Large-Scale Learning: Why, What & How [15min]
● Sandboxes for Supervised CL [20min]
● Toy approach to CL [15min]
● Discussion [20min]

References

● Bornschein et al. NEVIS'22 Benchmark (arXiv 2022, in submission)
● Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)
Conclusions

● (Most) ML is continual in a naïve and poorly automated way.
● As models get bigger, it is more important than ever to make them more efficient.
● CL is about making learning more efficient by leveraging knowledge acquired in the past.
● CL research needs good playgrounds to validate hypotheses. These playgrounds need to target continual large-scale learning.
● Conjecture: A big part to the solution of large-scale learning and CL is modularization.
Could this be the future?
Some Open Research Questions

- How to contribute to the development of large-scale learning without access to huge computational resources?
- Learning is about striking trade-offs: How to formalize and derive practical algorithms or architectures?
- Constraints depend on the setting and application. Is compute the right constraint?
- How to retain efficiency as we scale up?
- How to modularize in a distributed way?
- How to grow from small to big?
- How to do efficient meta-learning?
- How do we cross-validate in a never-ending learning setting?
- How to add/update/remove knowledge?
- What’s the role of memory?
Questions?