Continual Learning: On Machines that can Learn Continually

Official Open-Access Course @ University of Pisa, ContinualAI, AIDA

Lecture 6: Methodologies [Part 2]

Vincenzo Lomonaco
University of Pisa & ContinualAI
vincenzo.lomonaco@unipi.it

TABLE OF CONTENTS

Regularization
Strategies

Early-Stopping, L1 & L2, Dropout

Early Focus

- Study the impact of **activation functions**
- Study the impact of different **optimizers**
- **L2/L1** & **Dropout** regularizations
- **Early-Stopping**, **mb-size** and **learning rate** impact

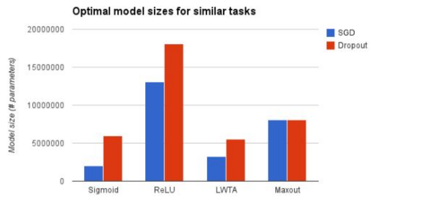


Figure 2. Optimal model size with and without dropout on the input reformatting tasks.

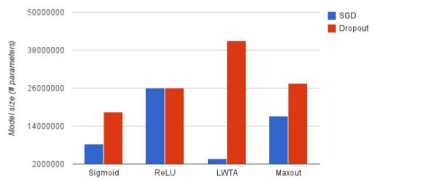


Figure 4. Optimal model size with and without dropout on the similar tasks experiment.

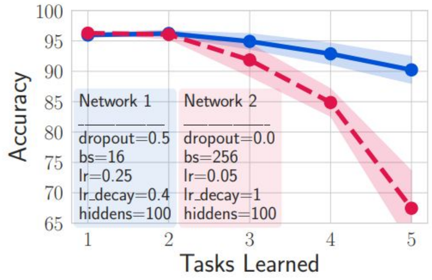


Figure 1: For the same architecture and dataset (Rotation MNIST) and only changing the training regime, the forgetting is reduced significantly at the cost of a relatively small accuracy drop on the current task. Refer to appendix C for details.

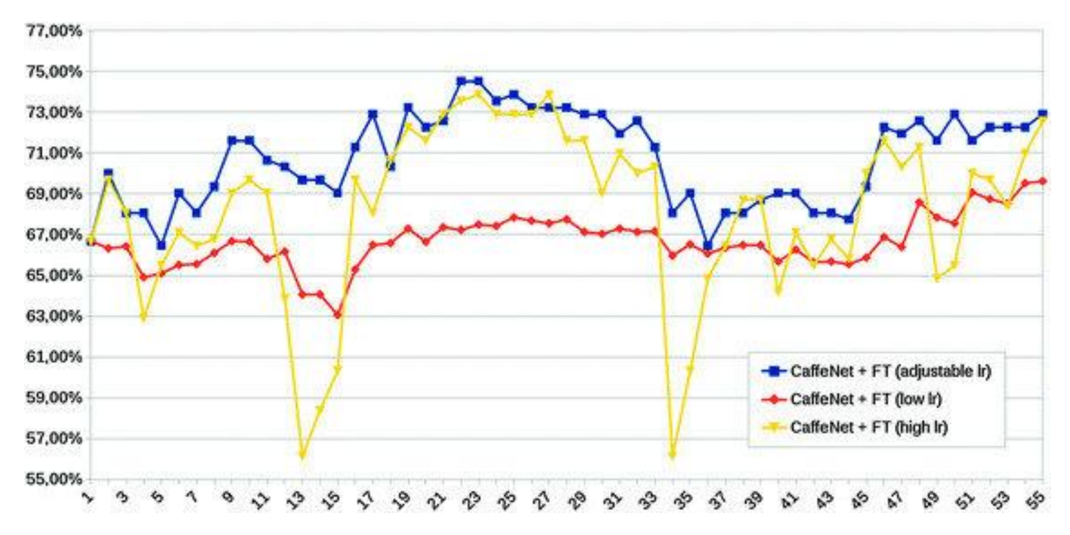
An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks, Goodfellow et al, 2015.
Understanding the Role of Training Regimes in Continual Learning, Mirzadeh et al. 2020.

Early-Stopping, L1 & L2, Dropout

Big Brother Experiment

- 7 finalists who stayed in the house for **55 days**
- **Violet-Jones** to detect faces
- **Adjustable learning rate** based on thresholds

Fig. 2.
Example images of the seven subjects contained in the SETB of the BigBrother Dataset.



Comparing Incremental Learning Strategies for Convolutional Neural Networks, Lomonaco et al, ANNPR 2016.

Learning Without Forgetting (LWF)

Key Aspects

- Straightforward application of **Knowledge Distillation**
- Originally designed for **Task-Incremental settings** can be easily extended to others
- **Efficient single-head implementations** exist
- **Easy to implement** and commonly used

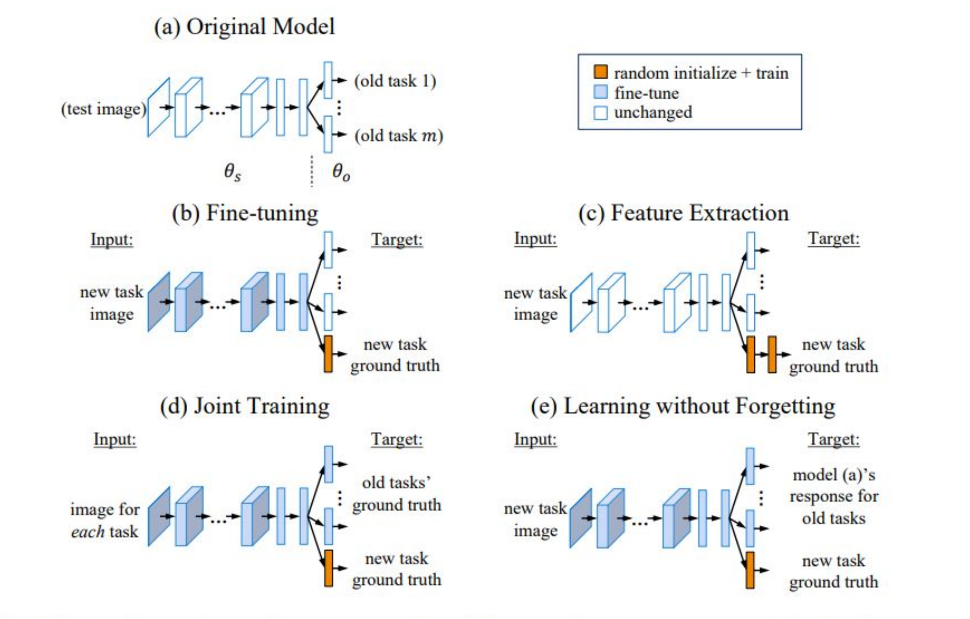


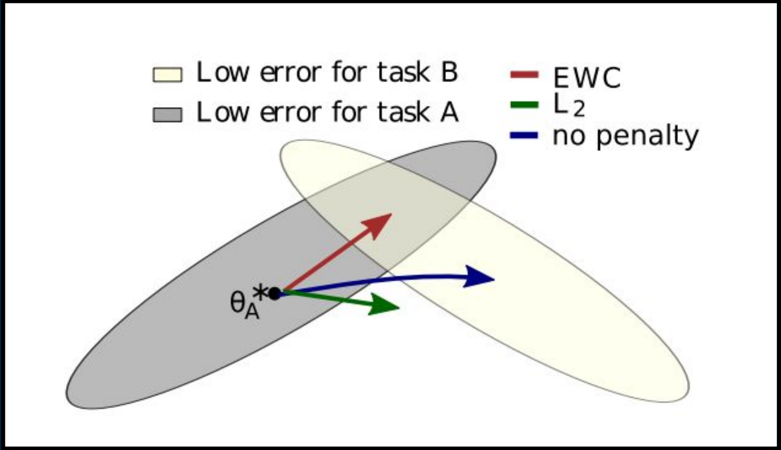
Fig. 2. Illustration for our method (e) and methods we compare to (b-d). Images and labels used in training are shown. Data for different tasks are used in alternation in joint training.

[Learning without Forgetting](), Li et al, TPAMI 2017
[Distilling the knowledge in a neural network](), Hinton et al, 2015.
[Continuous Learning in Single-Incremental Tasks](), Maltoni & Lomonaco, Neural Networks, 2019.

Elastic Weights Consolidation (EWC)

Key Aspects

- **Seminal work** that sparked new excitement and interest in *Deep Continual Learning*
- Interesting connection with more advanced computational neuroscience **memory consolidation theories**
- Many variations are possible: how to compute **parameters importance**? Do we need to maintain a **separate set of <optima weights, importance values>** for each experience?



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

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

Synaptic Intelligence (SI)

Key Aspects

- A **simple yet effective** way of computing weights importances
- Main idea: "*a parameter **importance is proportional to its contribution to the loss decrease** over time*"
- Even in this case **efficient online implementation exists**
- Hyper-parameters may be **difficult to calibrate**

$$L(\boldsymbol{\theta}(t)+\boldsymbol{\delta}(t)) - L(\boldsymbol{\theta}(t)) \approx \sum_k g_k(t)\delta_k(t)\,, \tag{1}$$

$$\int_C \boldsymbol{g}(\boldsymbol{\theta}(t))d\boldsymbol{\theta} = \int_{t_0}^{t_1} \boldsymbol{g}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\theta}'(t)dt. \tag{2}$$

$$\begin{aligned}\int_{t^{\mu-1}}^{t^\mu} \boldsymbol{g}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\theta}'(t)dt &= \sum_k \int_{t^{\mu-1}}^{t^\mu} g_k(\theta(t))\theta_k'(t)dt \\ &\equiv -\sum_k \omega_k^\mu. \end{aligned} \tag{3}$$

$$\tilde{L}_\mu = L_\mu + \underbrace{c\sum_k \Omega_k^\mu \left(\tilde{\theta}_k - \theta_k\right)^2}_{\text{surrogate loss}} \tag{4}$$

$$\Omega_k^\mu = \sum_{\nu<\mu} \frac{\omega_k^\nu}{(\Delta_k^\nu)^2 + \xi} \quad . \tag{5}$$

Continual Learning Through Synaptic Intelligence, Zenke et al, 2017.
Continuous Learning in Single-Incremental Tasks, Maltoni & Lomonaco, Neural Networks, 2019.

CL with Hypernetworks

Key Aspects

- Main idea: *let's learn how to* ***generate network weights***
- It may be seen as a form of **neurogenesis regulation**
- **Underlying hypothesis**: learning in this "compressed" space is less subject to forgetting
- **Difficult to scale** on higher-dimensional problems and without tasks labels

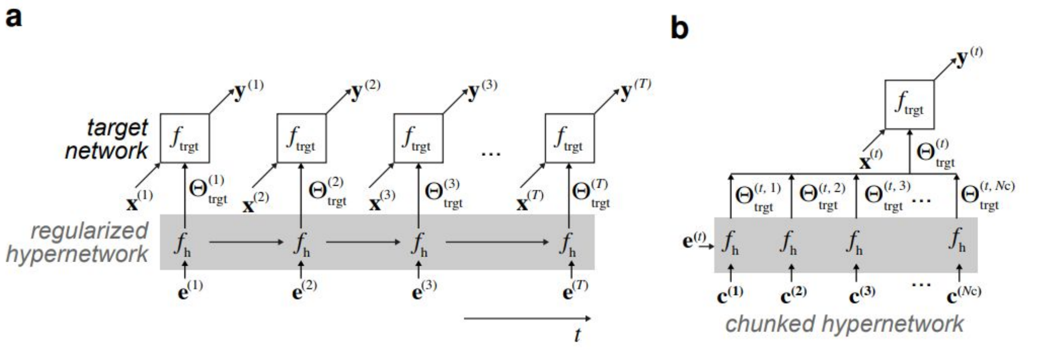


Figure 1: **Task-conditioned hypernetworks for continual learning.** **(a)** Commonly, the parameters of a neural network are directly adjusted from data to solve a task. Here, a weight generator termed *hypernetwork* is learned instead. Hypernetworks map embedding vectors to weights, which parameterize a target neural network. In a continual learning scenario, a set of task-specific embeddings is learned via backpropagation. Embedding vectors provide task-dependent context and bias the hypernetwork to particular solutions. **(b)** A smaller, chunked hypernetwork can be used iteratively, producing a chunk of target network weights at a time (e.g., one layer at a time). Chunked hypernetworks can achieve model compression: the effective number of trainable parameters can be smaller than the number of target network weights.

[Continual learning with hypernetworks](), Von Osvald et al, ICLR 2020.

Summary & Next Steps

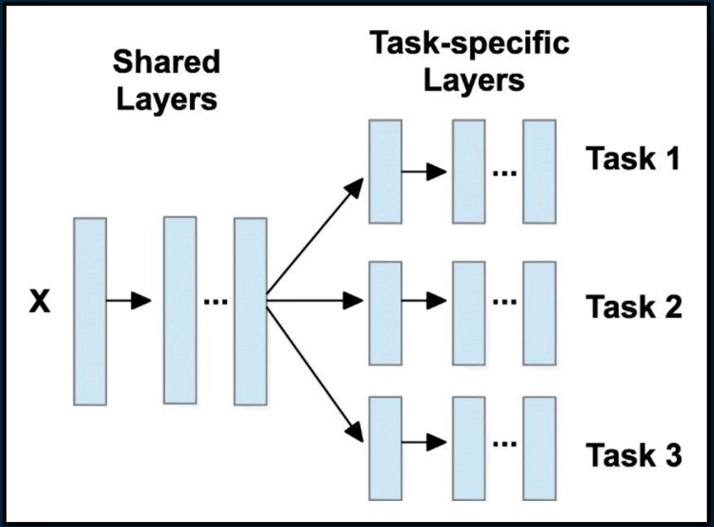
- Quite **elegant formulation** (mostly changing the loss function, adding regularization terms)
- Towards a **more principled definition of continual optimization**
- **Especially effective in Domain-Incremental** scenarios
- Better investigation in the gradient dynamics while learning may be useful
- **Plug & play orthogonal regularization terms** may be interesting to study
- We expect **significant advances** in this area in the years to come

Architectural Strategies

Multi-Head Architectures

Key Elements

- Great to **specialize behaviours if the notion of task is explicit**
- It clearly separate **shared parameters** with **private parameters**
- It may be constructed **"internally"** by the model when a significant "shift" is detected
- **A new head for each experience**: quite inefficient and possibly ineffective



[Overcoming catastrophic forgetting in neural networks](), Kirkpatrick et al, PNAS 2017.

Copy Weights with Re-Init (CWR)

Algorithm 1 CWR* pseudocode: $\bar{\Theta}$ are the class-shared parameters of the representation layers; the notation $cw[j]$ / $tw[j]$ is used to denote the groups of consolidated / temporary weights corresponding to class j. Note that this version continues to work under NC, which is seen here a special case of NIC; in fact, since in NC the classes in the current batch were never encountered before, the step at line 7 loads 0 values for classes in B_i because cw_j were initialized to 0 and in the consolidation step (line 13) $wpast_j$ values are always 0.

```
1:  procedure CWR*
2:    cw = 0
3:    past = 0
4:    init Θ̄ random or from pre-trained model (e.g. on ImageNet)
5:    for each training batch B_i:
6:      expand output layer with neurons for the new classes in B_i
        never seen before
7:      tw[j] = { cw[j],  if class j in B_i
                { 0,      otherwise
8:      train the model with SGD on the s_i classes of B_i:
9:        if B_i = B_1 learn both Θ̄ and tw
10:       else learn tw while keeping Θ̄ fixed
11:     for each class j in B_i:
12:       wpast_j = sqrt(past_j / cur_j), where cur_j is the number of patterns
          of class j in B_i
13:       cw[j] = (cw[j]·wpast_j + (tw[j] − avg(tw))) / (wpast_j + 1)
14:       past_j = past_j + cur_j
15:     test the model by using Θ̄ and cw
```

Key Aspects

- Developed for the **fully connected linear classifier** (may be extended to multiple layers)
- **Dual memory system approach**: one for better plasticity, one for memory consolidation
- Very simple and efficient, yet effective solution **agnostic to the experience content (NI, NC, NIC) and specific scenario**

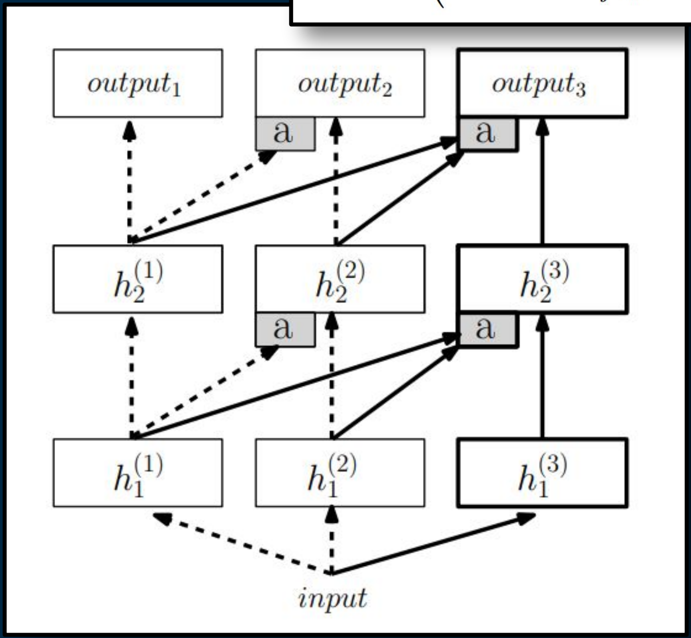
Rehearsal-Free Continual Learning over Small Non-I.I.D. Batches. Lomonaco et al, CLVision Workshop at CVPR 2020.

Progressive Neural Networks (PNNs)

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

Key Aspects

- Main **focus on forward transfer** and re-use of previously acquired representational power
- **Previous "columns" are frozen** inhibiting backward knowledge transfer
- Quite inefficient: **significant grow in the parameter space**, very difficult to scale on longer sequences of experiences
- **Adapters + pruning** can be used to tame complexity

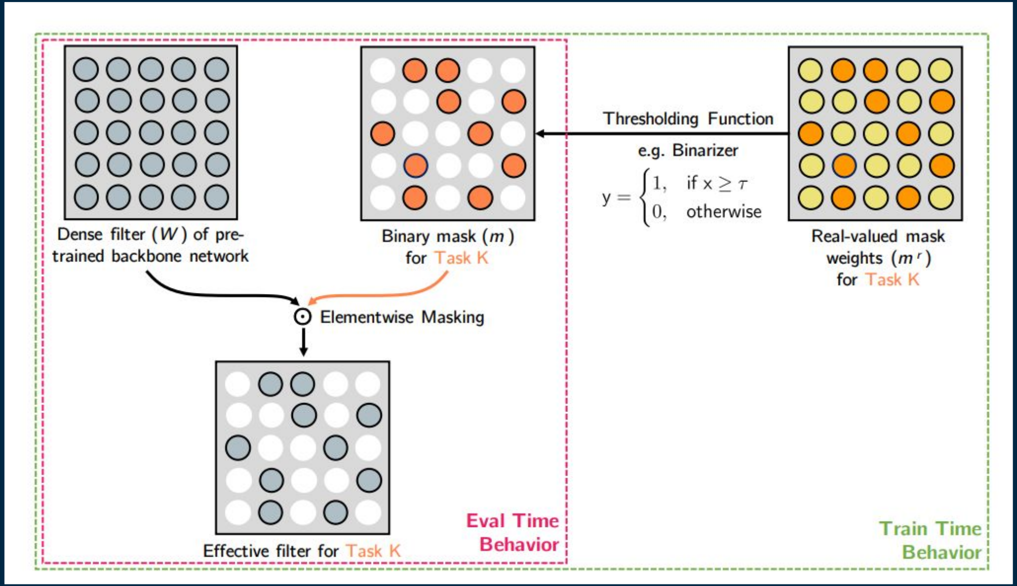


Progressive Neural Networks, Rusu et al. 2016.

Weights Mask (Piggyback)

Key Aspects

- Starting from a **pre-trained model** (backbone)
- Adding a **mask for each weight**, train float then binarize
- This achieves **zero-forgetting**, but no knowledge transfer
- It is quite efficient, and **handful of KBs per experience / task**, but it needs task labels



Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights, Mallya et al. 2018.
PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning, Mallya et al 2017.

Hard Attention to the Task (HAT)

Key Aspects

- Similar idea as Piggyback: use **hard attention masks** for each task
- The **mask is on the neurons not weights** (as "**inhibitory synapses**"), gradients masks can be created based on them
- Some form of forward transfer exist in the concept of "**cumulative attention**" and no pre-train model is necessary
- **Good accuracy-effectiveness trade-off** (binary vs real attention masks)
- Subject to the same limitations as Piggyback as for the **task labels availability**

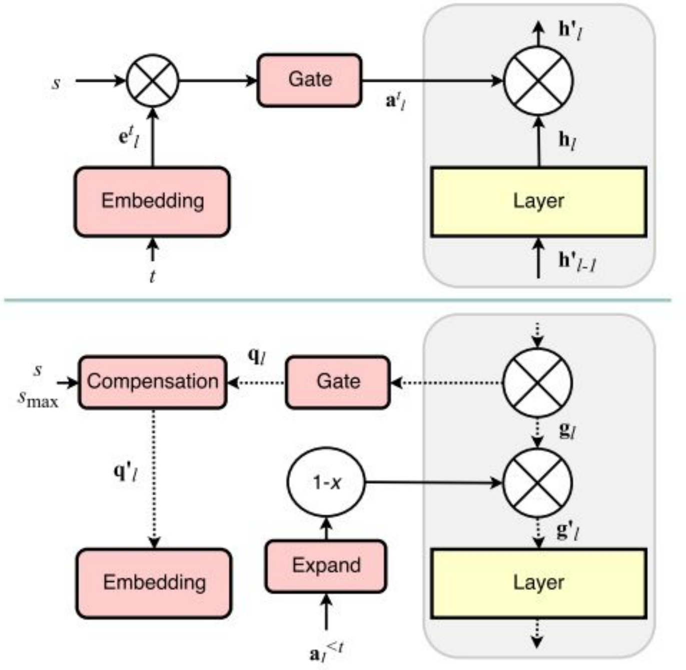


Figure 1. Schematic diagram of the proposed approach: forward (top) and backward (bottom) passes.

Overcoming catastrophic forgetting with hard attention to the task, Serra et al. 2018.

Supermasks in Superimposition

Key Aspects

- **Good binary masks** that applied to **random weights** exist
- **Random weights can be generated** on the fly based on random seed
- They can be used in **superimposition**

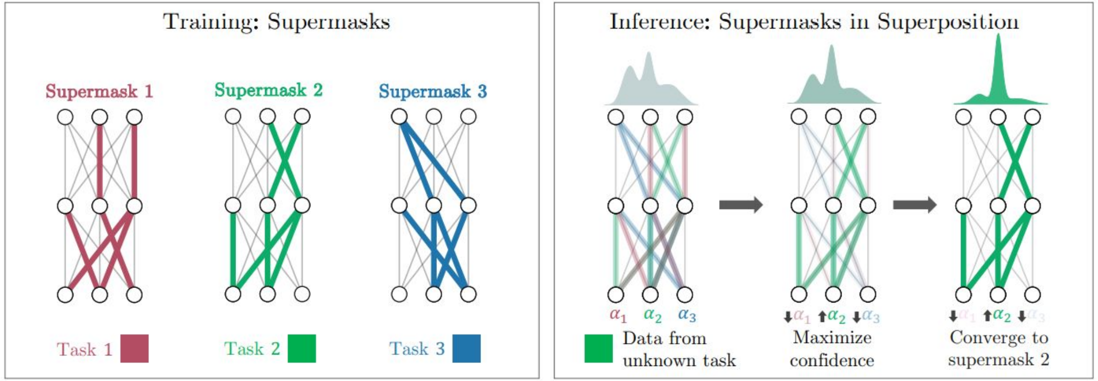


Figure 1: **(left)** During training SupSup learns a separate supermask (subnetwork) for each task. **(right)** At inference time, SupSup can infer task identity by superimposing all supermasks, each weighted by an α_i, and using gradients to maximize confidence.

Supermasks in Superposition, Wortsman et al. 2020.

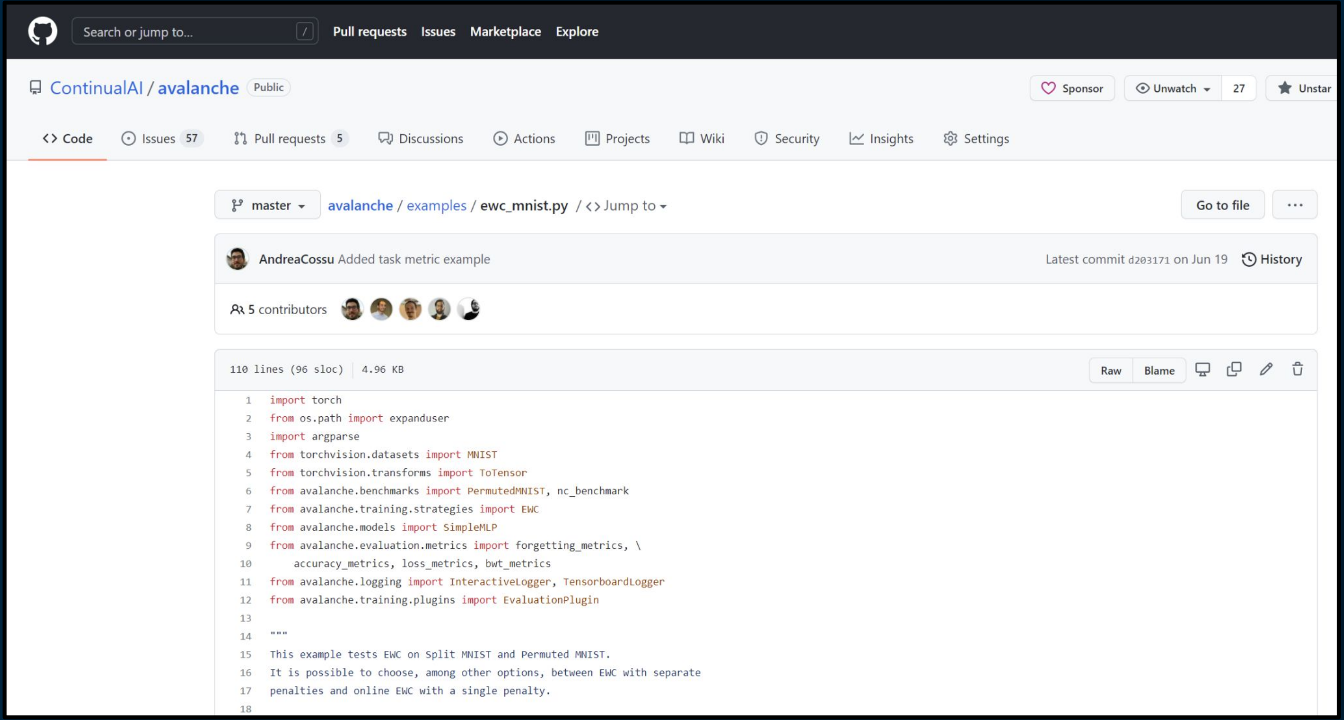
Summary & Next Steps

- Architectural methods may be **quite effective in terms of performance metrics** and reducing forgetting (knowledge preserving)
- **Difficult to perform efficient knowledge transfer** and parameter sharing
- Often involve **constant growing in the parameter space**
- The **often leverage task-specific supervised signals**
- Interesting **link** with **structural plasticity** in biological learning systems
- More **flexible and dynamic architecture re-arrangements** based on available resources may be an interesting future research direction

Avalanche
Implementation

Avalanche EWC, LWF & CWR Implementation

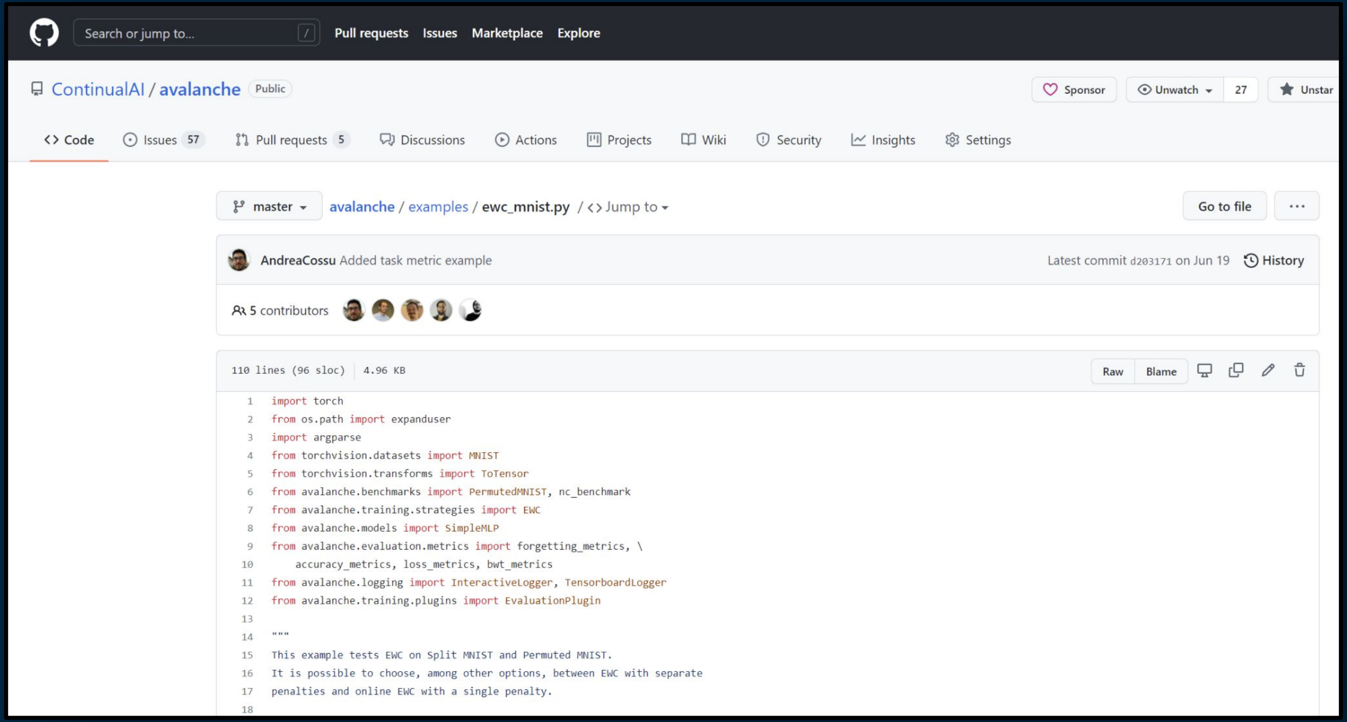
Demo Session!



https://github.com/ContinualAI/avalanche

Your Turn: Regularization Strategies in Class-Incremental Scenarios

Hands-on Session!



https://github.com/ContinualAI/avalanche

Next:

Methodologies [Part 3], Applications & Tools

Do you have any questions?

vincenzo.lomonaco@unipi.it
vincenzolomonaco.com
University of Pisa

THANKS