Continual Learning

Strategies – Regularization

Antonio Carta

antonio.carta@unipi.it

Outline

Regularization Methods

- Motivations
- Classic regularization: weight decay, lr, sparsity

Regularization methods in CL:

- Functional Regularization: knowledge distillation and LwF
- Constraints: GEM

Regularization – Motivations

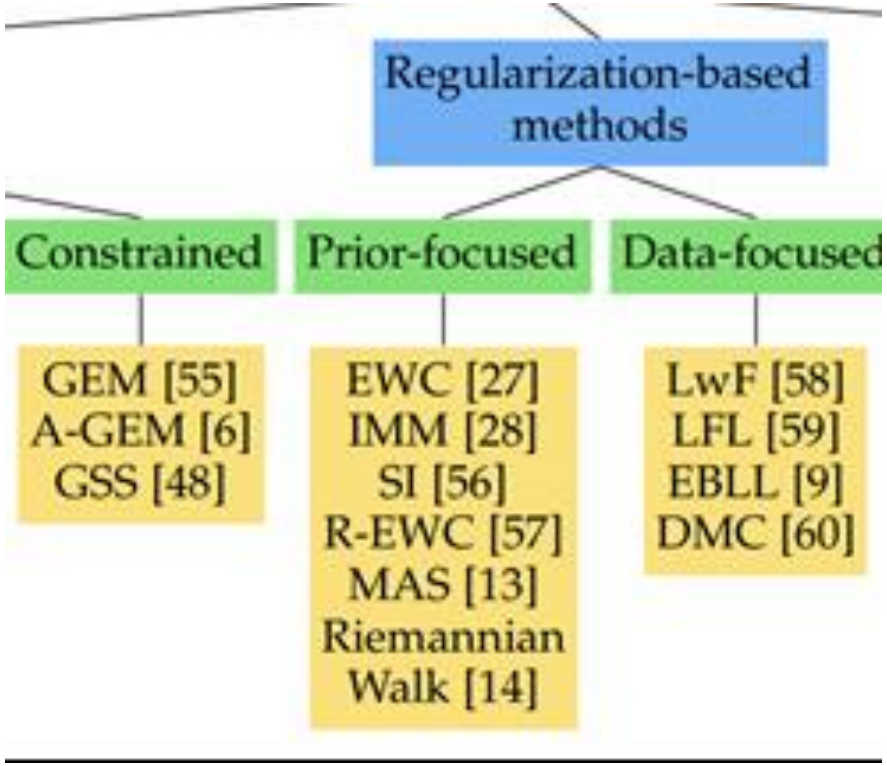
- Sometimes, we cannot store the data for replay
- Even if we can, the limited buffer does not contain all the information about the previous experiences

Goals for this lecture:

- Can we model forgetting as a constraint to the new training step?
- Can we use the old model to rehearse the old experiences?

Regularization – Categories

- **Constrained**:
  - Find a mathematical definition of forgetting as an optimization constraint
- **Prior-focused** (previous lecture):
  - Model loss function of previous tasks with surrogate losses.
  - Bayesian CL: prior used to model previous tasks
- **Functional/Data Regularization**:
  - Replicate the behavior of the old model on the previous tasks while learning the new ones



A continual learning survey: Defying forgetting in classification tasks. De Lange et al, TPAMI 2021.

Classic Regularization

Early-Stopping, L1 & L2, Dropout

- Parameters such as early stopping, l1/l2 weight decay and dropout can mitigate forgetting
- The more you regularize
  - -> the less the model is changing
  - -> the model has less forgetting
- In practice, these hyperparameters alone are not enough to prevent forgetting

- See the ref for experimental results on the impact of activation functions, different optimizers and L2/L1 & Dropout regularizations

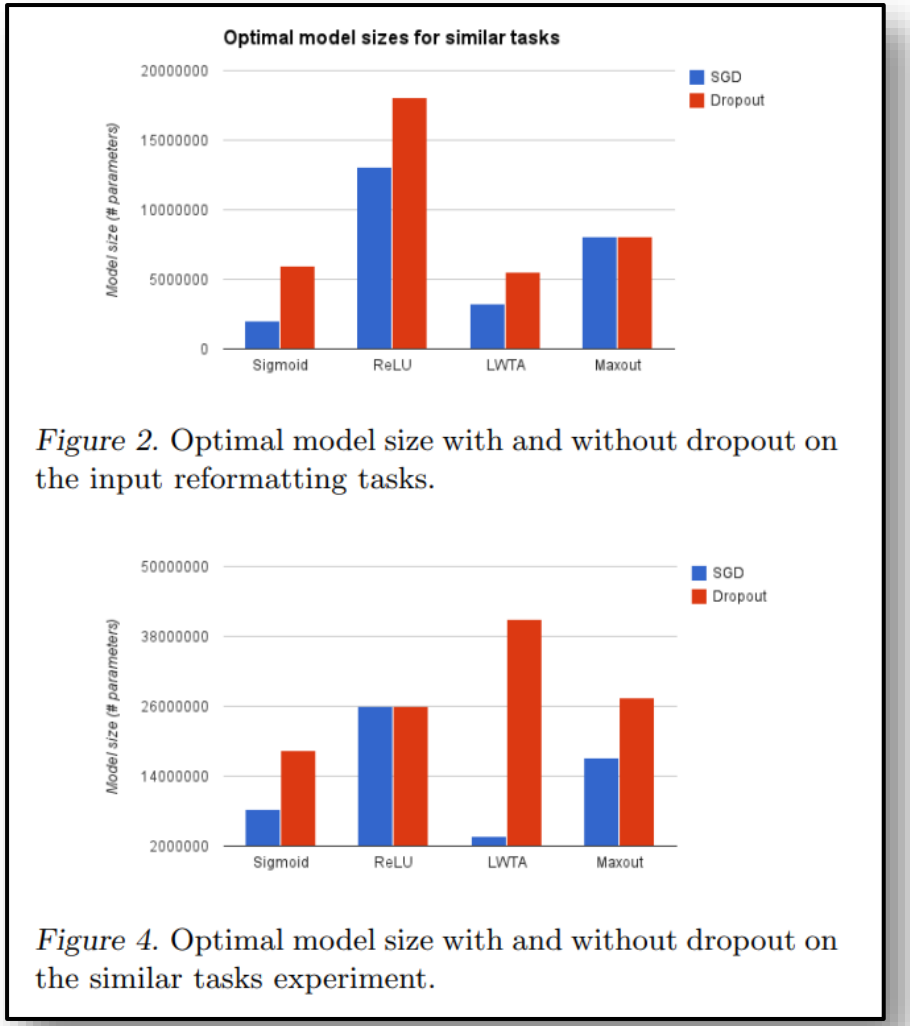


Figure 2. Optimal model size with and without dropout on the input reformatting tasks.

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

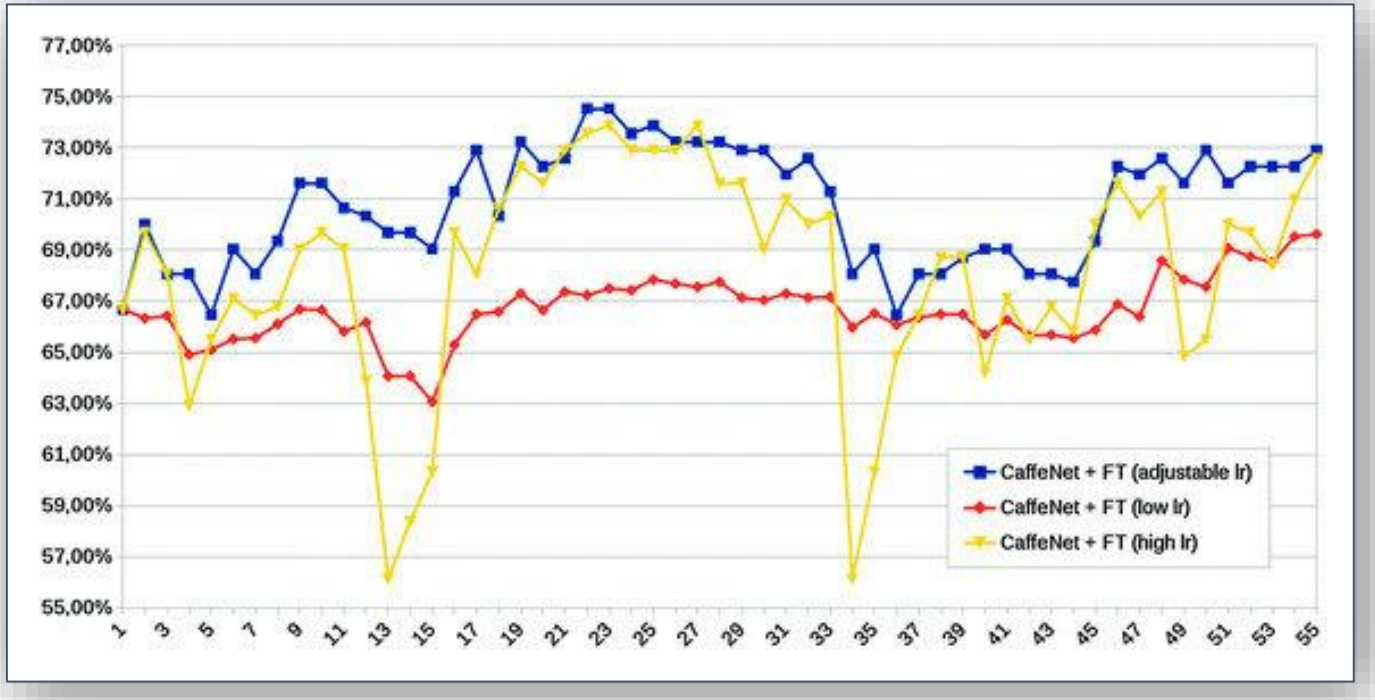
An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks, Goodfellow et al, 2015.

Early-Stopping, L1 & L2, Dropout

Example: Big Brother Experiment

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

- In general, custom lr schedules can work well
- We can recognize distribution drifts using the instantaneous loss and adjust the learning rate based on them



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

CL – An Optimization Perspective

- In offline training, the model optimizes a single loss $L(D, \theta)$ that is constant during the entire training loop
- In replay-free CL, the model optimizes $L(D_t, \theta)$, while the true objective is $\sum_t L(D_t, \theta)$
- We need to ensure that the new minima remains a good solution for the old experiences

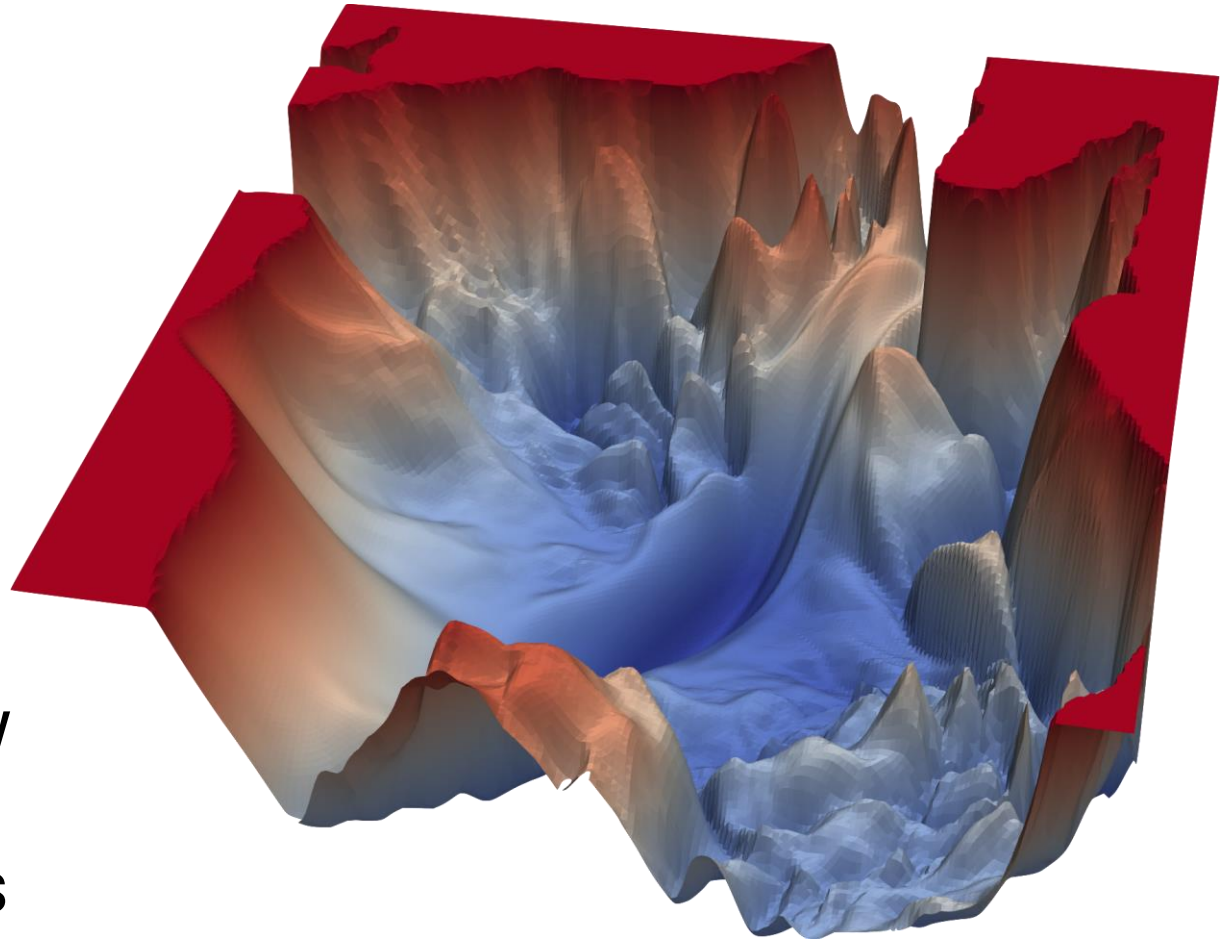


Image: H. Li et al. 2018. "Visualizing the Loss Landscape of Neural Nets." NIPS 2018

CL and Curvature

- Given the Taylor approximation around the solution

$$\mathcal{L}^{\text{old}}(\theta) \approx \mathcal{L}^{\text{old}}(\theta_t) + \nabla\mathcal{L}^{\text{old}}(\theta_t)^T(\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)^T Hf(\theta_t)(\theta - \theta_t)$$

- We argued that we can use the curvature to model the loss on past data around the solution
- DNN losses have multiple minima, each one with a different local curvature
- **IDEA**: Can we find a minima with a flat curvature? Even without any regularization, flat minima should mitigate forgetting

CL Training Regimes

The choice of optimizer, lr, scheduling, and batch size are critical

- First experience: needs to learn a stable solution
  - This is a strong motivation for pretraining whenever it's feasible
- Later: incremental steps must not destroy previously learned knowledge
- The two requirements are quite different.

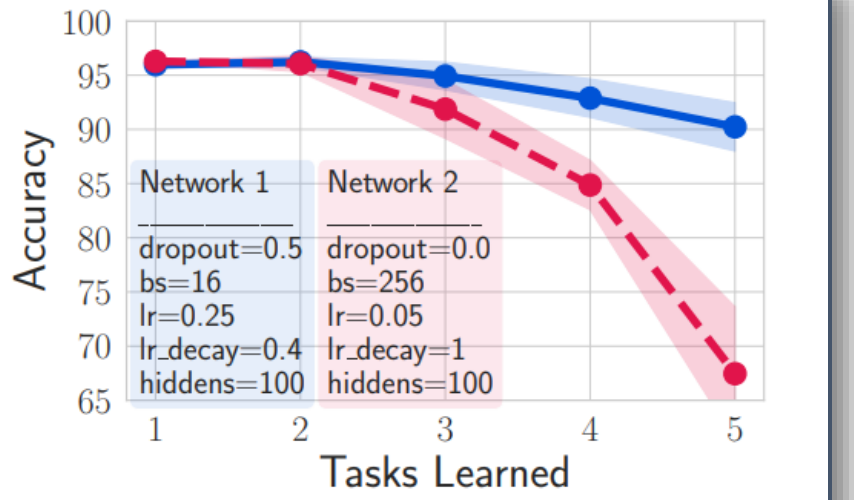


Figure 1: For the same architecture and dataset (Rotation MNIST) and only changing the training regime, the forgetting is reduced significantly at the cost of a relatively small accuracy drop on the current task. Refer to appendix C for details.

Understanding the Role of Training Regimes in Continual Learning, Mirzadeh et al. 2020.

CL Loss Approximation

The geometry of the loss landscape matters:

- w_1^* solution after exp.1, w_2^* solution after exp2, $\Delta w = w_2^* - w_1^*$
- If the new solution is close enough, we can linearly approximate the loss on the previous experiences (second-order Taylor expansion)
- the wider the curvature of the first task, the less the forgetting
  - This depends on eigenvalues of the hessian at w_1^*

$$F_1 = L_1(w_2^*) - L_1(w_1^*) \approx \frac{1}{2}\Delta w^\top \nabla^2 L_1(w_1^*)\Delta w \leq \frac{1}{2}\lambda_1^{max} \|\Delta w\|^2$$

Understanding the Role of Training Regimes in Continual Learning, Mirzadeh et al. 2020.

CL Loss Approximation

The geometry of the loss landscape matters:

- If the new solution is close enough, we can linearly approximate the loss on the previous experiences
  - **IDEA1**: try to change the model as little as possible
  - **IDEA2**: try to learn as much as possible in the first task
- the wider the curvature of the first task, the less the forgetting
  - **IDEA3**: try to guide the optimization trajectory towards flatter minima

$$F_1 = L_1(w_2^*) - L_1(w_1^*) \approx \frac{1}{2}\Delta w^\top \nabla^2 L_1(w_1^*)\Delta w \leq \frac{1}{2}\lambda_1^{max} \|\Delta w\|^2$$

Understanding the Role of Training Regimes in Continual Learning, Mirzadeh et al. 2020.

Learning Rate, Batch Size, Scheduling

By setting the SGD hyperparameters appropriately, we can guide the algorithm towards better solution

- It's a form of inductive bias of the SGD

General Rule: learning rate, batch size, and optimizer

- **start** with a high initial learning rate for the first task to obtain a wide and stable minima.
- **for each subsequent task**, slightly decrease the learning rate but also decrease the batch-size
- prefer SGD to Adam

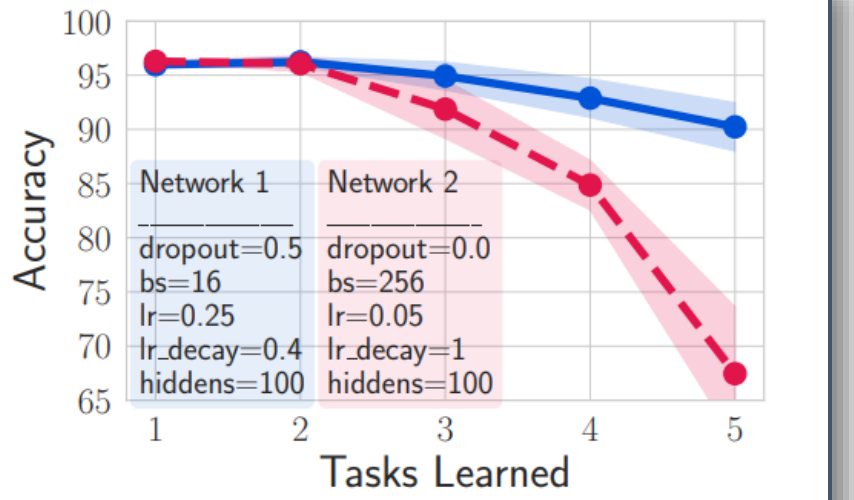


Figure 1: For the same architecture and dataset (Rotation MNIST) and only changing the training regime, the forgetting is reduced significantly at the cost of a relatively small accuracy drop on the current task. Refer to appendix C for details.

Understanding the Role of Training Regimes in Continual Learning, Mirzadeh et al. 2020.

Benefits of Sparsity

- Sparse representations have several helpful properties for CL
- Separate different task into independent subnetworks
  - Independent => no interference => zero forgetting
  - High level of sparsity make it possible to store many networks even with limited memory
- The goal is to activate different subnetworks for each task to reduce interference
- Biologically plausible: 10^{12} neurons in the brain and 10^{15} synapses $\approx$ 1000 connections for each neuron

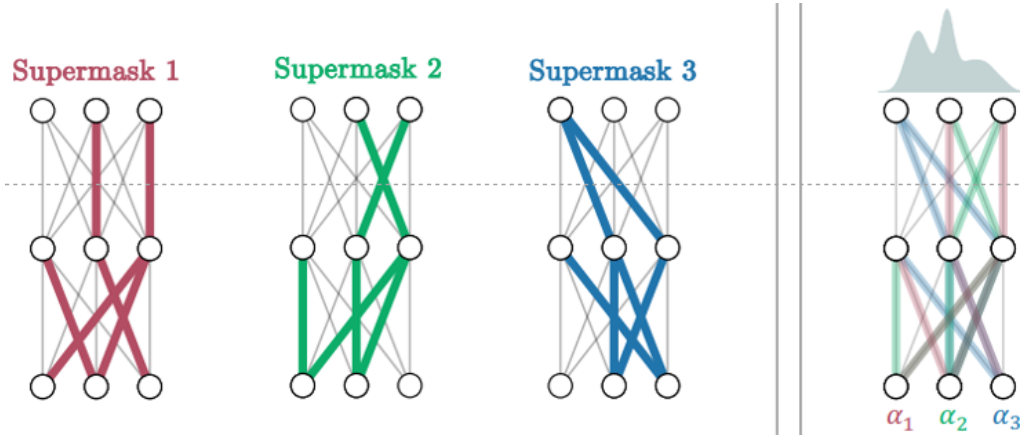


Image from Supermasks in Superposition, Wortsman et al. 2020.

Types of Sparsity

- Weight sparsity:
  - Small sparse networks for each task help to design cheap ensemble models
  - A single (weight) sparse network for all the tasks does not help
- Activation sparsity:
  - Sparse representation should be more orthogonal with each other and interfere less
- Gradient/update sparsity:
  - Sparse gradients should be more orthogonal and interfere less
- NOTE: sparsity is also concerned with stability, not plasticity. Often, sparse methods will also reduce forward transfer

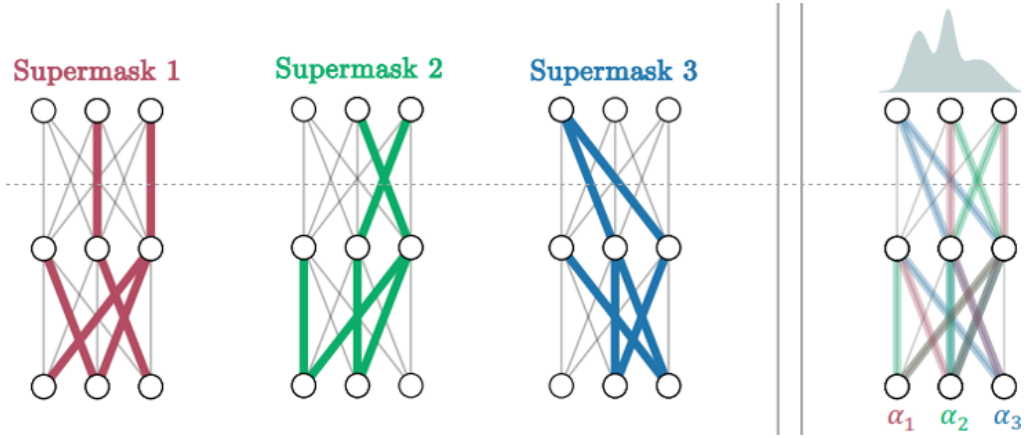
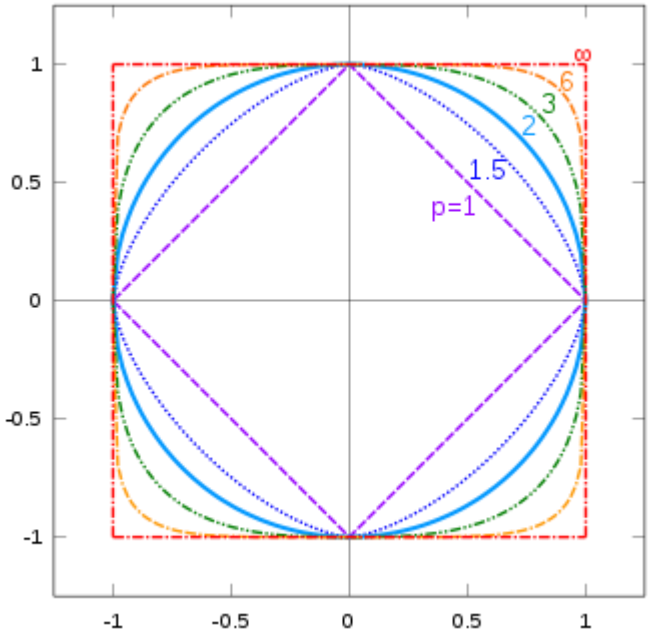


Image from Supermasks in Superposition, Wortsman et al. 2020.

Sparsity in CL Methods

- **Two approaches**:
  - **Implicit**: regularization method that implicitly force sparsity (today)
  - **Explicit**: architectural methods that explicitly build sparse subnetworks (next lecture)
- **Regularization methods can implicitly force sparsity**
  - l1 encourages sparsity $|\theta|$
  - l2 encourages small weights (not sparsity) $|\theta|^2$



Lp norm unit circles
Source: Wikipedia

LWTA – Local Winner Take All

- **WTA:** competition mechanism where only the highest activated is propagated
  - Losers are masked to zero
- **L:** Local competition. Each layer is divided into separate blocks.
- **Sparsity** induced by WTA
- Competition is a biologically plausible mechanism

$$y_i^j = \begin{cases} h_i^j & \text{if } h_i^j \geq h_i^k, \ \forall k = 1..n \\ 0 & \text{otherwise.} \end{cases}$$

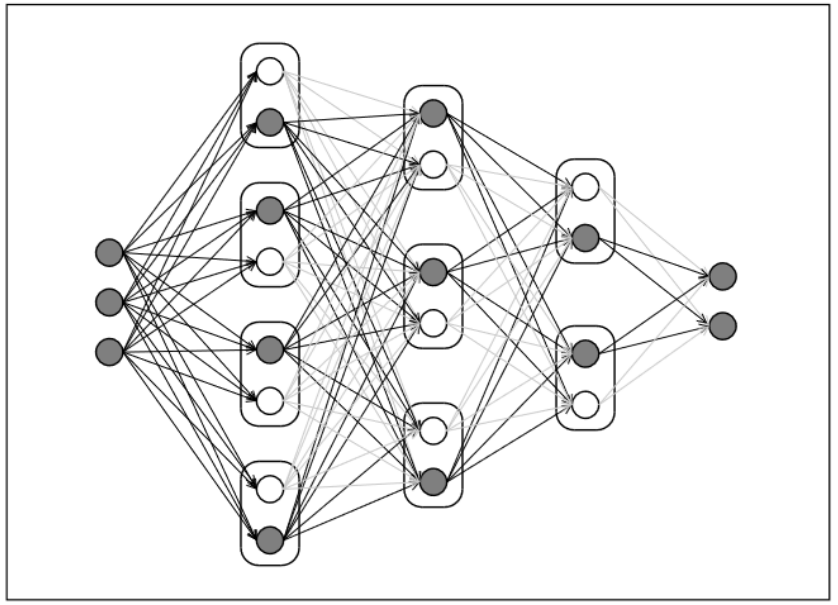
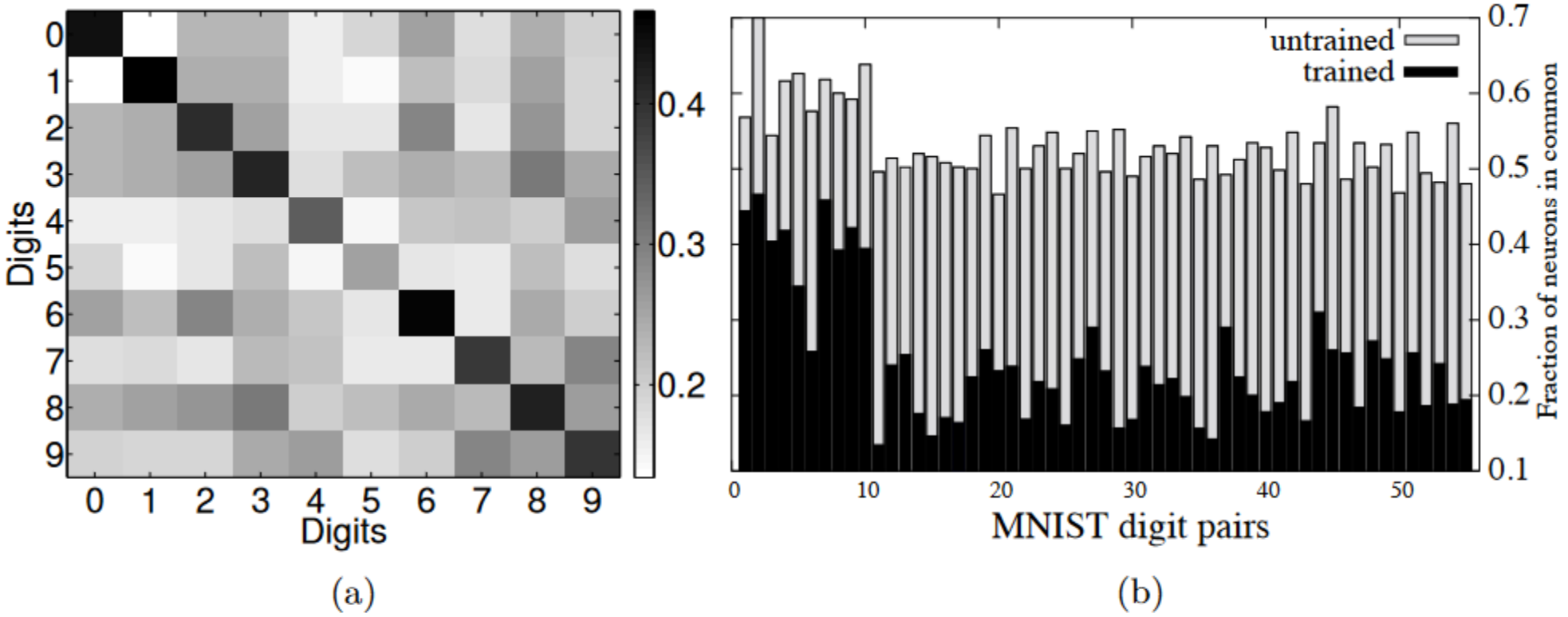


Figure 1: A Local Winner-Take-All (LWTA) network with blocks of size two showing the winning neuron in each block (shaded) for a given input example. Activations flow forward only through the winning neurons, errors are backpropagated through the active neurons. Greyed out connections do not propagate activations. The active neurons form a subnetwork of the full network which changes depending on the inputs.

Srivastava, Rupesh K, et al.. "Compete to Compute," NIPS 2013

Sparsity



Subnetwork: The set of units that activate for a particular task/class
Two observations:

- Similar digits have similar subnetworks
- Subnetwork are less similar after training than before => Competition helps learning specialized and separate subnetworks

Srivastava, Rupesh K, et al.. "Compete to Compute," NIPS 2013

Functional Regularization

Functional Regularization

- we have access to a model f_{i-1}^{CL} (the previous model) that learned experiences $S^{train}[1:i-1]$
- **IDEA**: let's replicate the old model behavior and update it only on the new examples
- **PROBLEMS**:
  - What objective do we use?
  - What data do we use?

$$f_i^{CL}(\boldsymbol{x}, k) = f_{i-1}^{CL}(\boldsymbol{x}, k), \quad \forall \boldsymbol{x} \in \mathbb{R}^N, k \neq i$$
$$f_i^{CL}(\boldsymbol{x}, i) = f_i^{Exp}(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbb{R}^N.$$

Equations in a Multi-Task Scenario
x=input
k=task label
i=task label for new task
f^{Exp}=model for new task

First Eq: copy old CL model on the first i-1 tasks
Second Eq: copy new model on the new task

Knowledge Distillation (KD)

KD: Offline training method to replicate the output of a pretrained model

- **Teacher**: frozen pretrained model
- **Student**: the new model that we want to train

KD is a general method with many applications outside CL:

- Example: Reducing the size of a model:
  - Example teacher: ResNet101 pretrained on ImageNet
  - Example student: ResNet18 trained with KD

Why does it work?

- Supervised training provides hard targets (i.e. the correct class)
- KD provides soft targets, which are more informative
  - Example: soft targets encode similarities between classes
  - Informally called «dark knowledge»

KD Objective – KL-Divergence and MSE

- The KL-Divergence measures the similarity between two probability distributions (teacher and student)
  - We are measuring the distance between the pdf of the teacher and the student
- $\hat{y}$ teacher, y student

$$E(\mathbf{x} \mid t) = -\sum \hat{y}_i(\mathbf{x} \mid t) \log y_i(\mathbf{x} \mid t)$$

- Alternative: MSE between the logits $||\hat{y} - y||_2^2$
  - Often more robust in CL
- Another alternative: KD + crossentropy

Learning Without Forgetting (LwF)

LwF implements functional regularization with:

- **Objective**: knowledge distillation
- **Data**: current data

Key Aspects

- Straightforward application of KD in CL
- Originally designed for Task-Incremental settings then extended to single task.
- Efficient: requires only an additional forward pass with the teacher.
- Easy to implement and commonly used

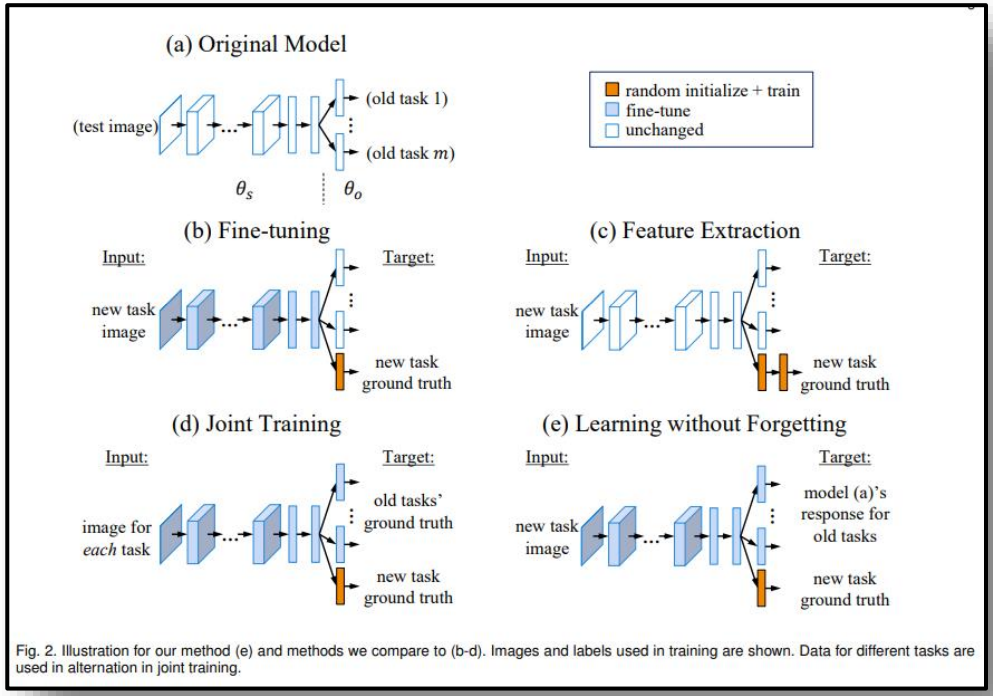


Fig. 2. Illustration for our method (e) and methods we compare to (b-d). Images and labels used in training are shown. Data for different tasks are used in alternation in joint training.

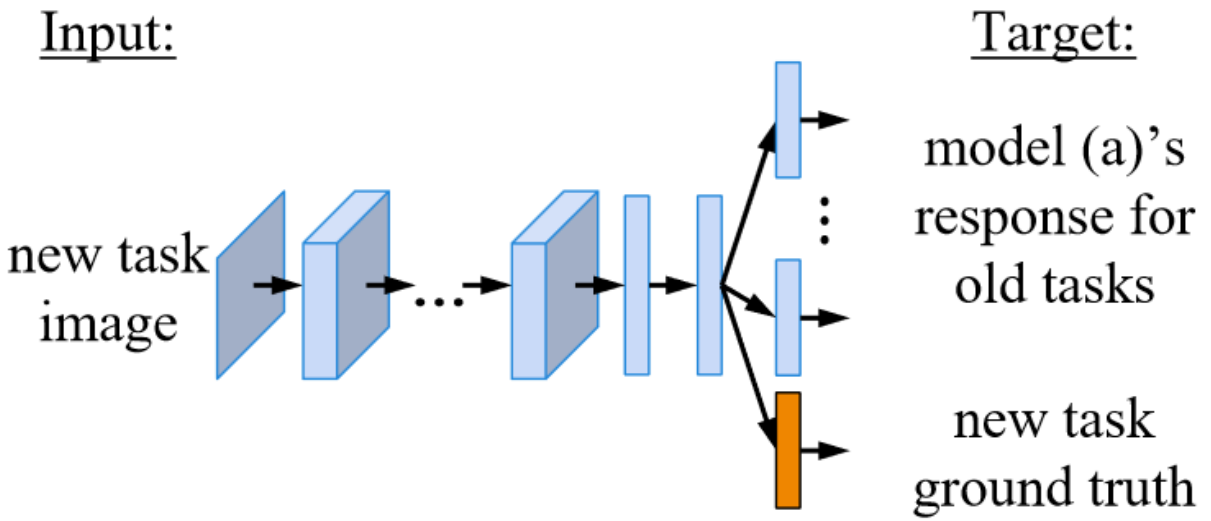
Learning without Forgetting, Li et al, TPAMI 2017
Distilling the knowledge in a neural network, Hinton et al, 2015.
Continuous Learning in Single-Incremental Tasks, Maltoni & Lomonaco, Neural Networks, 2019.

LwF in Task-Incremental CL

- **Multihead**: separate output layer for each task.
- **KD** computed on the old head using the new data (L_{old})
- **CE** computed on the new head using the new data (L_{new})

$$\mathcal{L}_{old}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o) = -\sum_{i=1}^{l} y'^{(i)}_o \log \hat{y}'^{(i)}_o$$

$$\mathcal{L}_{new}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = -\mathbf{y}_n \cdot \log \hat{\mathbf{y}}_n$$

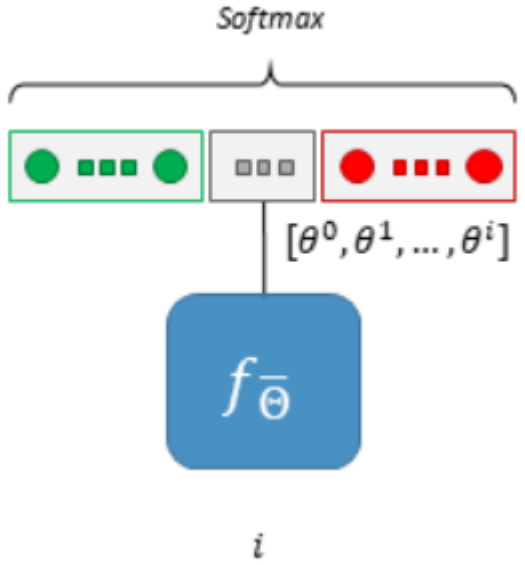


$$\lambda_o \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n)$$

Learning without Forgetting, Li et al, TPAMI 2017

LwF in Class-Incremental CL

- **Single Head**: a single output layer.
  - We distinguish between **new** units, **old** units, and **all** units.
- **KD** computed on the *old units* using the new data (L_{old})
- **CE** computed on *all the units* using the new data (L_{new})
- Many alternatives are studied in the literature



Single head in single incremental tasks. New and old units have different losses.

Distillation in Practice

- Even outside CL, KD is surprisingly competitive

Implementation «tricks»

- Long aumentation pipelines
- Long training Schedules
- Consistent teacher and student inputs (i.e. same augmentation for both)

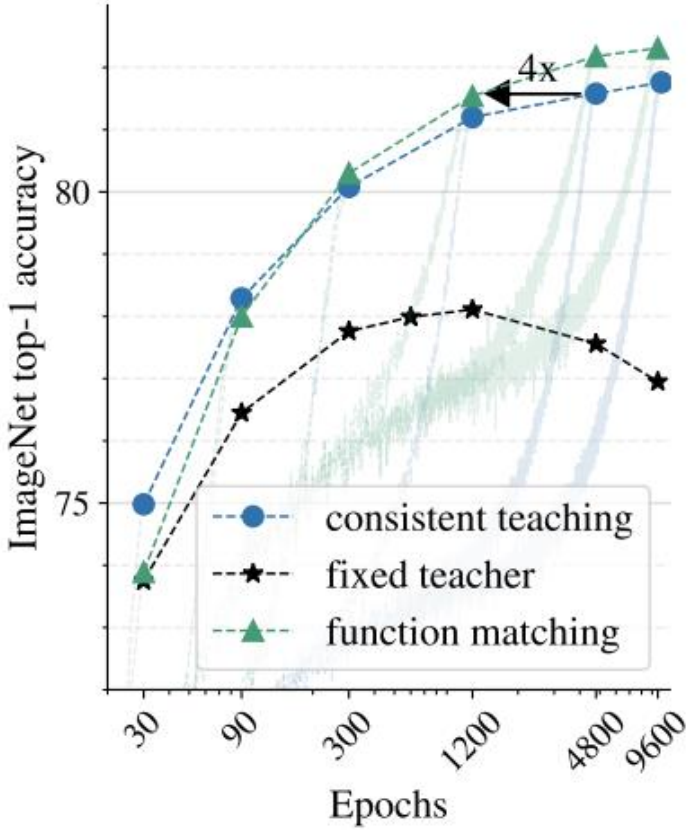


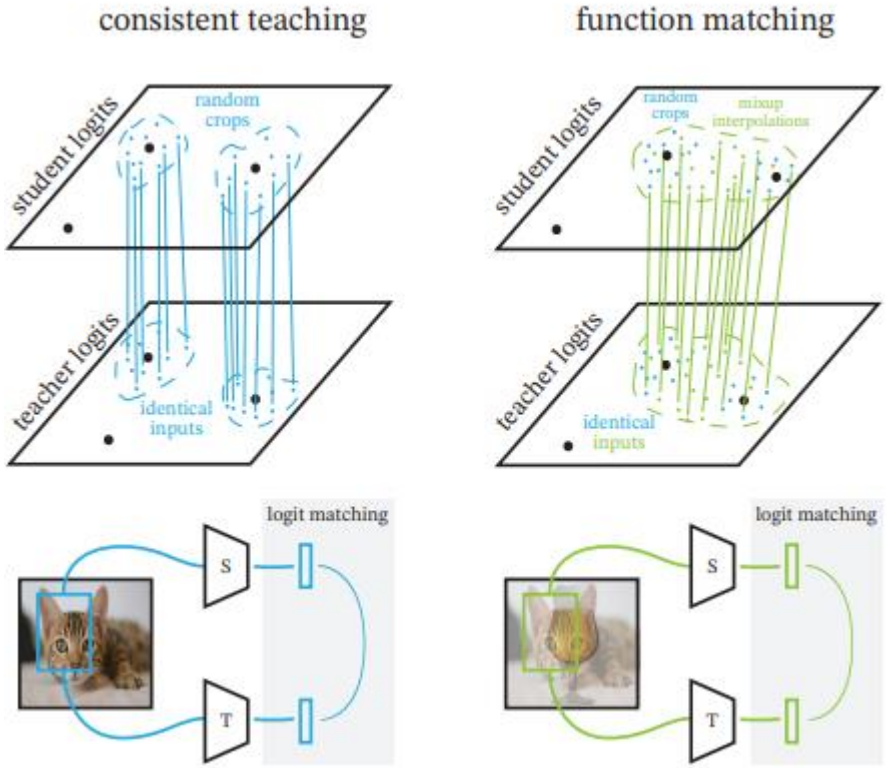
Figure 1. We demonstrate that distillation works the best when we train *patiently* for a large number of epochs and provide *consistent* image views to teacher and student models (green and blue lines). This can be contrasted to a popular setting of distilling with precomputed teacher targets (black line), which works much worse.

Beyer, Lucas, et al. "Knowledge distillation: A good teacher is patient and consistent." *CVPR*. 2022.

KD as Function Matching

KD is a **function matching** problem

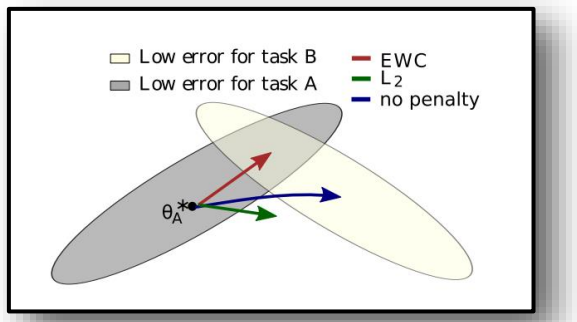
- For any input, we know exactly what output we want
  - The data is much less important in this problem than in a typical ML problem
  - Having diverse data close to the domain of interest helps but it's not a necessity
- KD works with any inputs (with different convergence speeds)
  - The teacher training data
  - Out-of-domain-data with augmentations
  - A new task data (like in LwF!!!)

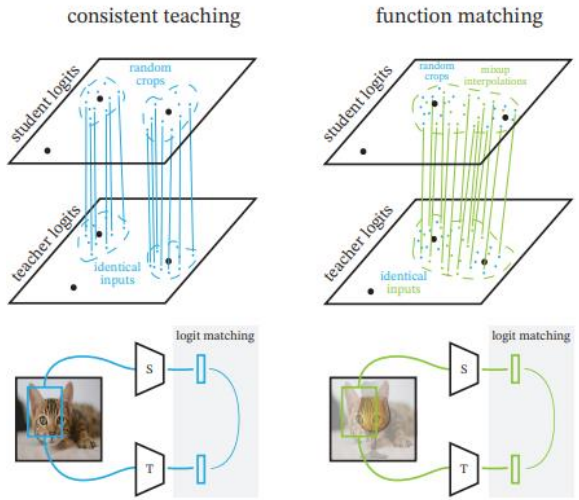


Beyer, Lucas, et al. "Knowledge distillation: A good teacher is patient and consistent." *CVPR*. 2022.

Weight vs Functional Regularization

- **Prior-based methods require that the new models don't move «too far» from the previous solutions**
  - We need a large model or a pretrained one to satisfy this requirement
  - It's only an approximation of the real objective
- **Functional regularization is much less restrictive**
  - The weights can change, as long as the output for the previous units is the same
  - The output of new units is completely unconstrained
  - The previous model provides the exact outputs that we want (no approximation)





KD with a Single Image

Figure 1: **Extrapolating from one image.** Strongly augmented patches from a single image are used to train a student (S) to distinguish semantic classes, such as those in ImageNet. The student neural network is initialized randomly and learns from a pretrained teacher (T) via KL-divergence. Although almost none of target categories are present in the image, we find student performances of $>69\%$ for classifying ImageNet's 1000 classes. In this paper, we develop this single datum learning framework and investigate it across datasets and domains.

Asano, Yuki M., and Aaqib Saeed. "Extrapolating from a single image to a thousand classes using distillation." *arXiv preprint* (2021).

Single image in CL

Table 3: Test accuracy of DAC with different data sources on SplitCIFAR100 (10 Tasks).

	Original Domain	Single Image	Natural	ACC
Current Data ($\mathcal{D}_i$)	✓		✓	84.2±0.5
ImageNet			✓	84.8±0.6
animals		✓	✓	82.1±0.5
city		✓	✓	80.5±0.8
bridge		✓	✓	79.0±0.6
hubble		✓		65.1±0.3
DeepInversion				64.9±3.3
noise		✓		10.7±0.5

Carta, Antonio, et al. "Projected Latent Distillation for Data-Agnostic Consolidation in Distributed Continual Learning." arXiv preprint (2023).

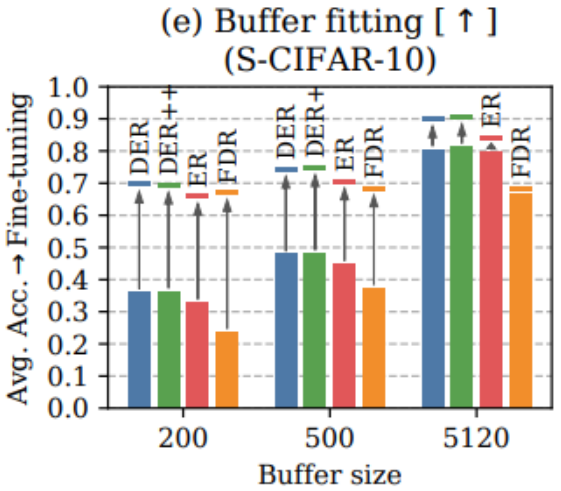
LwF

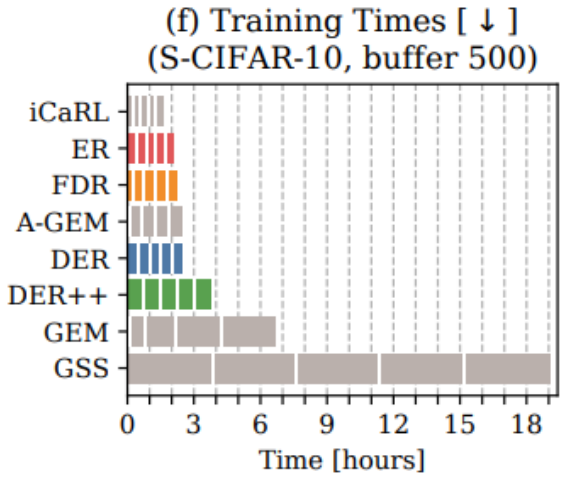
- We need **task boundaries** to know **when** to store the previous model
- The data that we use for KD (new data) is different from the one we used to train the teacher (old data)
- **Augmentations** help KD, even when they distort the image

KD with Replay

Dark Experience Replay:

- Save tuples x, z of <inputs, logits> in the buffer (class-balanced reservoir sampling)
- Rehearse with MSE loss on logits (KD):
- $reg \leftarrow \alpha \left\| z' - h_\theta(x'_t) \right\|_2^2$



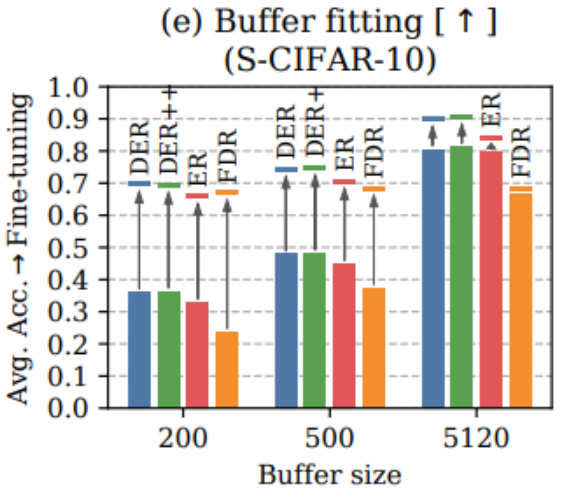


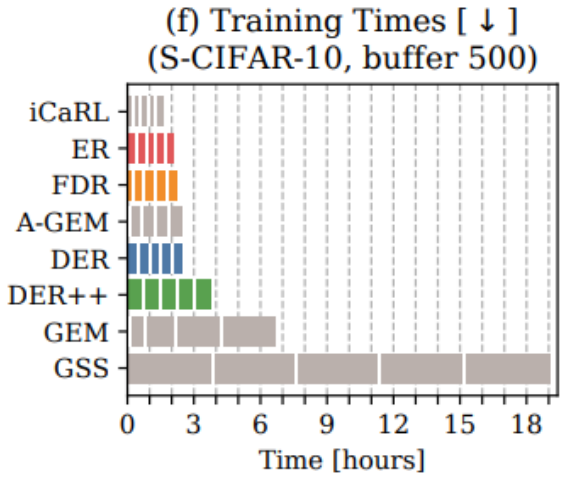
Buzzega, Pietro, et al. "Dark experience for general continual learning: a strong, simple baseline." *NeurIPS 2020*

KD with Replay

Dark Experience Replay:

- Doesn't require a teacher. We save the logits of the current teacher without updating them.
- Alternative: save also the label y and rehearse on KD loss + crossentropy loss
- Very competitive results and low computational cost





Buzzega, Pietro, et al. "Dark experience for general continual learning: a strong, simple baseline." *NeurIPS 2020*

KD in Continual SSL

- We can also use KD to train SSL models
- SSL loss on new data + KD loss on old embeddings

$$\mathcal{L} = \mathcal{L}_{SSL}(\boldsymbol{z}^A, \boldsymbol{z}^B) + \mathcal{L}_D\left(\boldsymbol{z}^A, \bar{\boldsymbol{z}}^A\right)$$
$$= \mathcal{L}_{SSL}(\boldsymbol{z}^A, \boldsymbol{z}^B) + \mathcal{L}_{SSL}\left(g(\boldsymbol{z}^A), \bar{\boldsymbol{z}}^A\right).$$

- A and B images, z^A, z^B embeddings of new model
- $\bar{\boldsymbol{z}}^A$ embedding of old model
- $\mathcal{L}_{SSL}$ is used as the distillation loss
- g is a projection network that maps from the new representation to the old ones
  - Allows to update the representations and change them

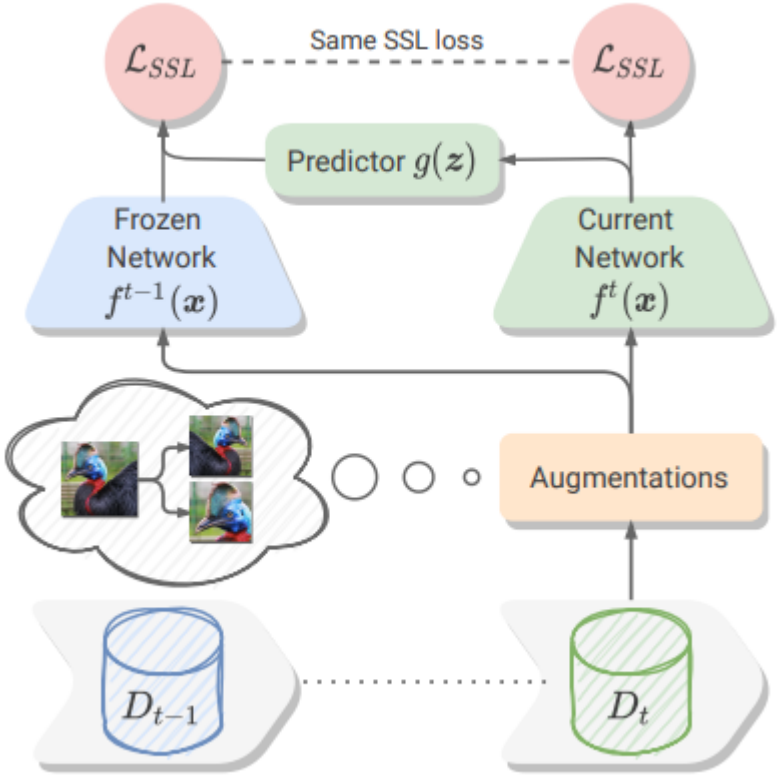


Figure 2. Overview of the CaSSLe framework.

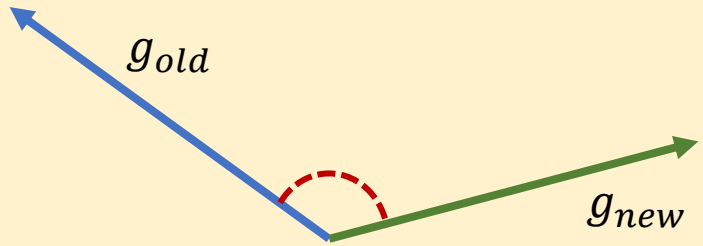
CaSSLe 2112.04215 (arxiv.org)

Constraints

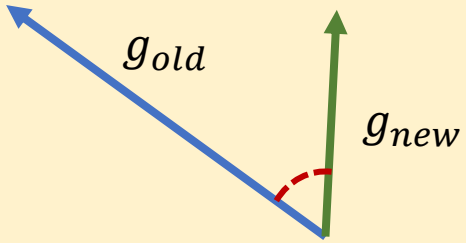
Regularization Constraints

- Compute interference explicitly with a mathematical definition
- Modify learning step to remove interference
- Gradient-based definition of interference based on the cosine similarity

Negative cosine similarity: interference
Loss on the old task will increase.



Positive cosine similarity: positive backward transfer. The loss on the old task will increase.



Lopez-Paz, David, and Marc'Aurelio Ranzato. "Gradient Episodic Memory for Continual Learning." NIPS 2017

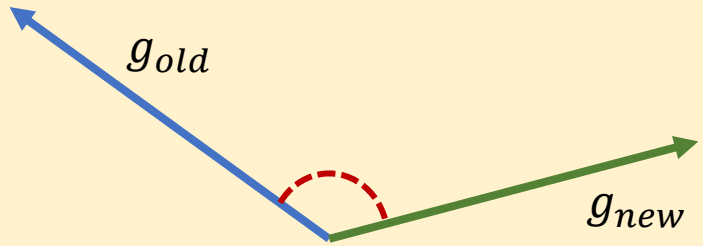
Regularization Constraints

Gradient-based definition of interference based on the cosine similarity

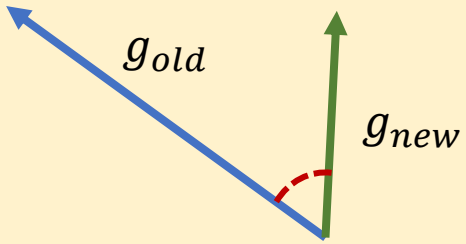
- Compute gradient for the old and new task
- Compute cosine similaritiy between the gradients
- **Negative**: interference!
- **Positive**: positive backward transfer
- **Zero**: orthogonal tasks without interference

$$\langle g, g_k \rangle := \left\langle \frac{\partial \ell(f_\theta(x,t),y)}{\partial \theta}, \frac{\partial \ell(f_\theta, \mathcal{M}_k)}{\partial \theta} \right\rangle \geq 0, \text{ for all } k < t.$$

Negative cosine similarity: interference
Loss on the old task will increase.



Positive cosine similarity: positive backward transfer. The loss on the old task will decrease.



Lopez-Paz, David, and Marc'Aurelio Ranzato. "Gradient Episodic Memory for Continual Learning." NIPS 2017

Estimating Gradient Interference

- **Setting**: Online Continual Learning with known task boundaries
- **Episodic Memory**: keep a balanced buffer
  - Sample from the buffer and compute gradients for the old tasks
  - We compute a separate gradient for each task
- **Interference**: compute the cosine similarity between gradients of the new task and old tasks
  - Positive: backward transfer!
  - Negative: interference

$$\langle g, g_k \rangle := \left\langle \frac{\partial \ell(f_\theta(x,t),y)}{\partial \theta}, \frac{\partial \ell(f_\theta, \mathcal{M}_k)}{\partial \theta} \right\rangle \geq 0, \text{ for all } k < t.$$

Lopez-Paz, David, and Marc'Aurelio Ranzato. "Gradient Episodic Memory for Continual Learning." NIPS 2017

Removing Interference

- Given the interference as the cosine similarity between gradients of the new task and old tasks
  - Positive: backward transfer!
  - Negative: interference
- We want to remove the component of the new gradient that interferes with previous tasks
- **Solution: Project** the gradient s.t. the constraint is satisfied for all the tasks.
  - Quadratic programming (QP) problem solved with off-the-shelf methods.
  - g current gradient, g_k gradient for old task k

$$\begin{aligned} \text{minimize}_{\tilde{g}} \quad & \frac{1}{2} \|g - \tilde{g}\|_2^2 \\ \text{subject to} \quad & \langle \tilde{g}, g_k \rangle \geq 0 \text{ for all } k < t. \end{aligned}$$

$$\langle g, g_k \rangle := \left\langle \frac{\partial \ell(f_\theta(x,t),y)}{\partial \theta}, \frac{\partial \ell(f_\theta, \mathcal{M}_k)}{\partial \theta} \right\rangle \geq 0, \text{ for all } k < t.$$

Lopez-Paz, David, and Marc'Aurelio Ranzato. "Gradient Episodic Memory for Continual Learning." NIPS 2017

Gradient Episodic Memory (GEM)

- At each iteration:
  - Update the memory with new data
  - Sample from memory
  - Compute gradient for each old task
  - Compute current gradient
  - Project gradient the remove the interference
  - Update the model

Gradient Episodic Memory

procedure TRAIN(f_θ, Continuum$_{\text{train}}$, Continuum$_{\text{test}}$)
  $\mathcal{M}_t \leftarrow \{\}$ for all $t = 1, \ldots, T$.
  $R \leftarrow 0 \in \mathbb{R}^{T \times T}$.
  for $t = 1, \ldots, T$ **do:**
    for (x, y) in Continuum$_{\text{train}}(t)$ **do**
      $\mathcal{M}_t \leftarrow \mathcal{M}_t \cup (x, y)$
      $g \leftarrow \nabla_\theta \ell(f_\theta(x, t), y)$
      $g_k \leftarrow \nabla_\theta \ell(f_\theta, \mathcal{M}_k)$ for all $k < t$
      $\tilde{g} \leftarrow$ PROJECT$(g, g_1, \ldots, g_{t-1})$, see (11).
      $\theta \leftarrow \theta - \alpha \tilde{g}$.
    end for
    $R_{t,:} \leftarrow$ EVALUATE$(f_\theta$, Continuum$_{\text{test}})$
  end for
  return f_θ, R
end procedure

$$\begin{aligned} \text{minimize}_{\tilde{g}} \quad & \frac{1}{2} \|g - \tilde{g}\|_2^2 \\ \text{subject to} \quad & \langle \tilde{g}, g_k \rangle \geq 0 \text{ for all } k < t. \end{aligned}$$

$$\langle g, g_k \rangle := \left\langle \frac{\partial \ell(f_\theta(x, t), y)}{\partial \theta}, \frac{\partial \ell(f_\theta, \mathcal{M}_k)}{\partial \theta} \right\rangle \geq 0, \text{ for all } k < t.$$

Lopez-Paz, David, and Marc'Aurelio Ranzato. "Gradient Episodic Memory for Continual Learning." NIPS 2017

GEM

Pros:

- Constraint formulation of forgetting
- Assuming the gradient estimate is correct (we only use few samples to estimate it) we can completely remove interference

Cons:

- Slow due to QP problem at every iteration
- Requires samples!
- The QP problem may not have a solution
  - This will happen at some point because the more tasks we have, the more constraints we will have to add to the model
  - GEM constraints reduce plasticity by constraining some update directions

Applications of Constraints

- Interference is a meta-objective
  - We can optimize directly, as in GEM
  - We can design a method that implicitly reduces the gradient interference

- **Meta continual learning**: minimize task interference explicitly with a loss or implicitly via MAML-like loop
- Online approximations of GEM constraints

Take-Home Messages

- Thinking about the effect of CL on the optimization trajectory allows to design methods that explicitly counteract forgetting, even without the previous data
- Knowledge distillation provides a general purpose tool for knowledge transfer
  - in our case, transfer from the old to the new model
  - Very effective in many different problems
- Constraint formulation of forgetting

Next Lecture

Architectural methods

- Task-Incremental methods
- Modular Architectures
- Sparsity