MIDTERM 4

Continual Learning with Deep Generative Replay

Pasquali Alex

The setting

- Deep learning is the state-of-the-art tool to solve many complex tasks.
- The trend has been to go towards **more and more complicated problems**, but in **narrow domains.**
- The scenario was usually **offline learning**, where the training data is all available prior to the deployment of the model.

This creates problems in terms of **adaptability** and **versatility** of the model, as well as of **scalability** and **feasibility** in some real-world applications.

Continual Learning

- **Continual learning** is a framework where the model learns sequentially from a stream of experiences.
- Data can become available during time
- No access to previously encountered data is necessary (depends on the approach)
- Constant computational and memory resources (depends on the approach)

The problem

"Attempts to train a comprehensive artificial intelligence capable of solving multiple tasks have been impeded by a ***chronic problem*** *called* ***catastrophic forgetting****"* [1]

Catastrophic Forgetting

- Problem where the model's **performance on previously learnt tasks degrades** when the model is trained on a new task because the parameters are optimized for the new one without considering the previous knowledge.
- **Stability** vs **plasticity** tradeoff
- It is **not sustainable** to train the model **from scratch** for every new experience

Some solution approaches

Regularization-based

- Elastic weight consolidation (Kirkpatrick et al, 2017)
- Learning without forgetting (Li et al, 2018)

Rehersal-based

- Rehearsal and Pseudorehersal (Robins, 1994)

More advanced replay strategies

- ***Generative Replay*** (Shin et al, 2017)
- Distilled Replay (Rosasco, Carta, Cossu, Lomonaco, Bacciu, 2021)

[1] Hanul Shin, Jung Lee, Jaehong Kim, Jiwon Kim. *Continual Learning with Deep Generative Replay*, 2017

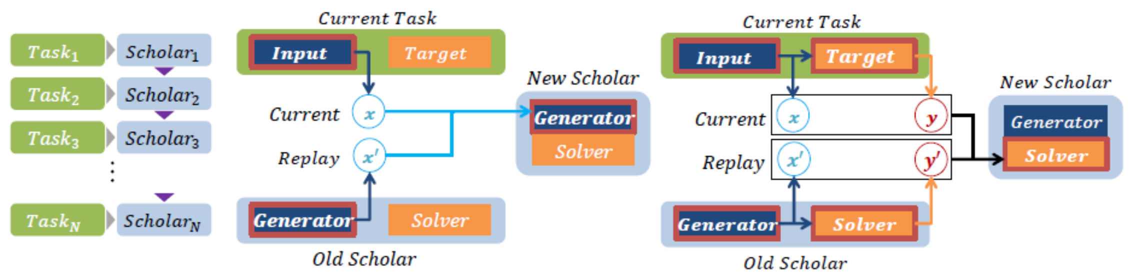
Deep Generative Replay

Scholar model
- capable of **learning** a new task and **teaching** its knowledge to other networks.
- Composed by a **generator** and a **solver**

Solver: Task-solving model (neural network)

Generator
- Deep generative model → **GAN**
- **Adversarial training** of generator and discriminator
- **Generator** learns to **mimic** real data distribution
- **Discriminator** learns to **distinguish** between real and generated data

Training
- **Sequential** training on **scholar** model
- Train current scholar while referring to recent copy of the network → current scholar learns from past one
- Equivalent to train a sequence of scholars



(a) Sequential Training (b) Training Generator (c) Training Solver

1) Generator training
- New generator (G) receives current task's input (**x**) and *replayed* inputs from previous tasks (**x'**)
- **x'** generated by the previous scholar's generator
- G learns to reconstruct **cumulative** input space

2) Solver training
- New solver (S) learns to couple inputs and targets from mix of real and replayed data
- Replayed targets are past solver's response to replayed inputs

More in depth

$$\underbrace{L_{train}(\theta_i) = r}_{}\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_i}\underbrace{[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})]}_{} + (1-r)\,\mathbb{E}_{\boldsymbol{x}'\sim G_{i-1}}\underbrace{[L(S(\boldsymbol{x}';\theta_i),S(\boldsymbol{x}';\theta_{i-1}))]}_{}$$

Previous generator

r is the **ratio** between the importance of the **new task** compared to the **older ones**

Loss of the solver with **real data** and **real targets**

Loss of the solver where the **data** is **generated** by the previous generator and the **targets** are the **previous solver's response** to those same inputs

$$L_{test}(\theta_i) = r\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_i}[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})] + (1-r)\,\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_{past}}[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})]$$

Cumulative distribution of past data

At test time, no self-generated inputs are used

Preliminary experiment

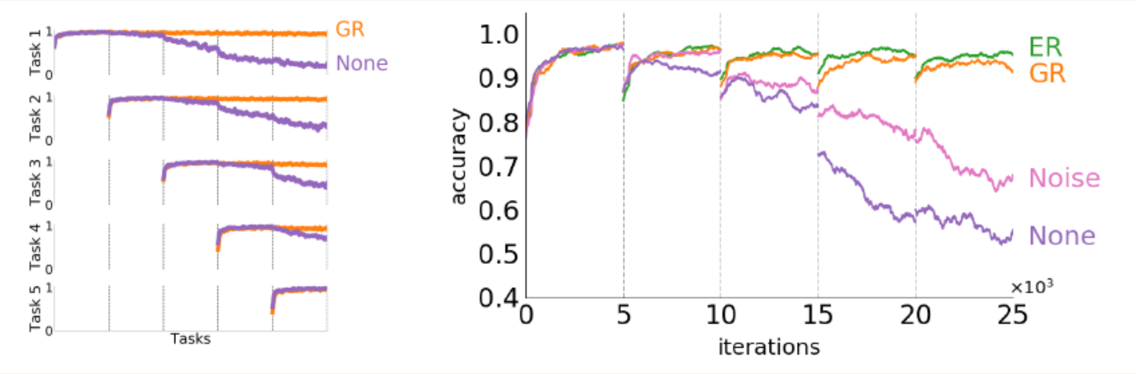
- A trained scholar model alone suffices to train an "empty" network.
- It's shown that scholar models transfer knowledge without loosing information.

Table 1: Test accuracy of sequentially learned solver measured on full test data from MNIST database. The first solver learned from real data, and subsequent solvers learned from previous scholar networks.

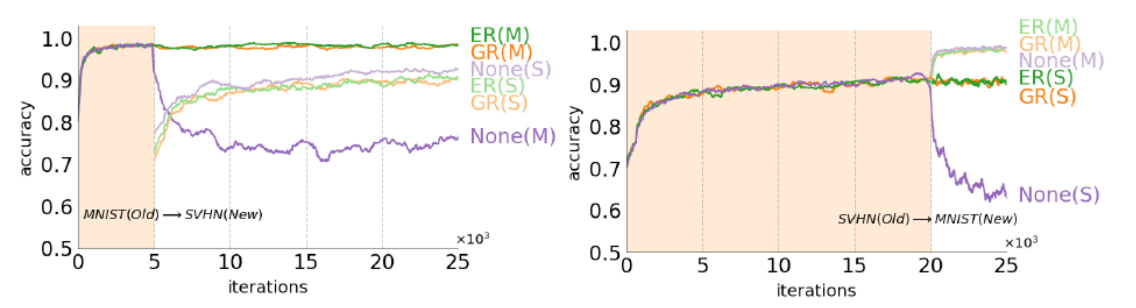
	$Solver_1$	$\rightarrow$	$Solver_2$	$\rightarrow$	$Solver_3$	$\rightarrow$	$Solver_4$	$\rightarrow$	$Solver_5$
Accuracy(%)	98.81%		98.64%		98.58%		98.53%		98.56%

Experiments

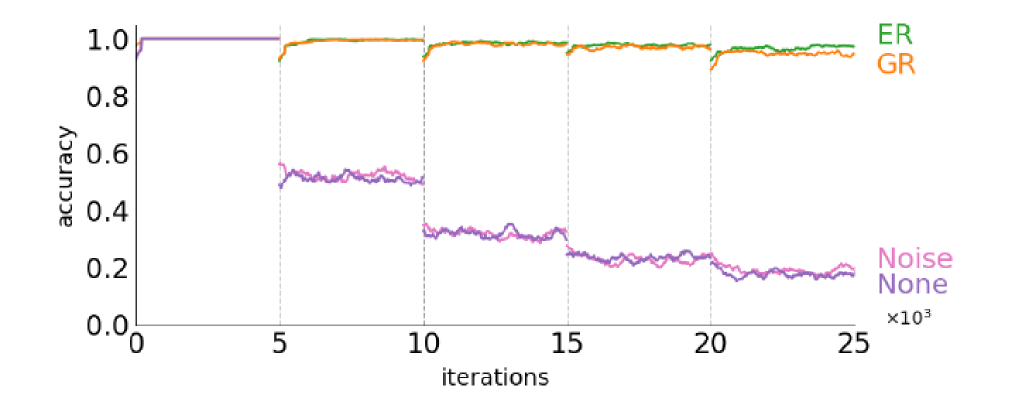
Independent tasks
- Image classification on MNIST
- Pixels shuffled by permutation unique to each task
- Solver must classify permuted inputs to original class



Learning new domains
- Model sequentially trained to classify MNIST and SVHN numbers



Learning new classes
- Model sequentially trained on mutually exclusive subsets of classes
- Solver must eventually classify examples from all classes



Conclusion

- GR has the advantage of **not needing any memory**
  - Unlike rehearsal strategies
  - Maintain former knowledge solely from input-target pairs produced by the saved network
  - Ease of balancing performances on former and new tasks, but old tasks are balanced as a whole

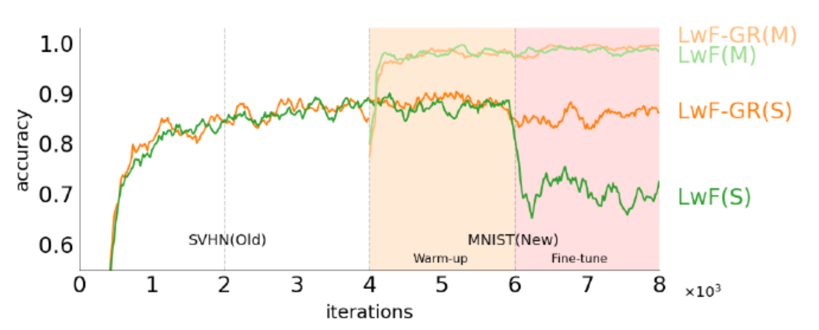
- GR has a **constant computational time**
  - No extra parameters for each task like in LwF, where training time linearly increases for each new task

- GR is **not** regularization-based
  - Approaches such as EWC might suffer from the tradeoff between performances on old tasks and new tasks

- The **quality of GR** heavily **depends** on the quality of the **generator**
  - But improvements in training generative models can be directly reflected into improvements in GR

- Generated samples can represent well the data distribution
  - But they might be less dense of information compared to *Distilled Replay*, thus more samples would be necessary to mitigate forgetting

- LwF needs context about the task that's being performed (to use a task-specific output layer), GR does not.
  - GR can be used to augment LwF, adding samples similar to former tasks inputs (instead of using only current task's inputs to revoke past knowledge)



THANK YOU FOR YOUR ATTENTION

Pasquali Alex