
Hw14

Lifelong Learning

ML TAs

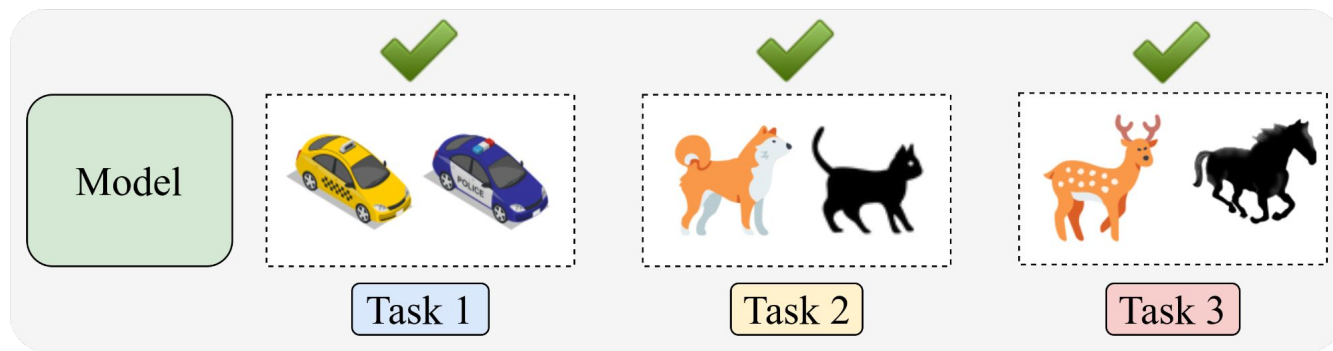
ntu-ml-2021spring-ta@googlegroups.com

Outline

- Introduction
- Dataset
- Sample Code
- COOL Quiz
- Grading
- Submission

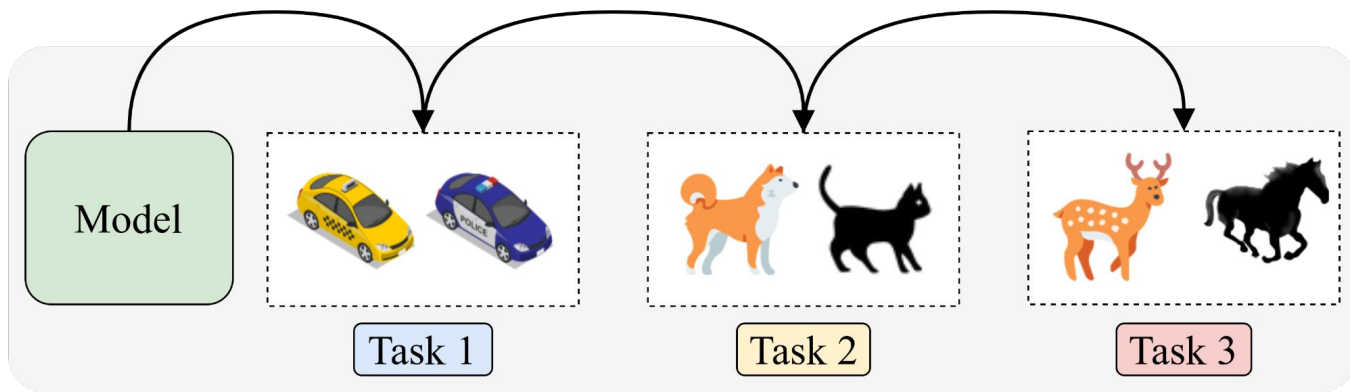
Introduction - LifeLong Learning

Goal: A model can beat all task!



Introduction - LifeLong Learning

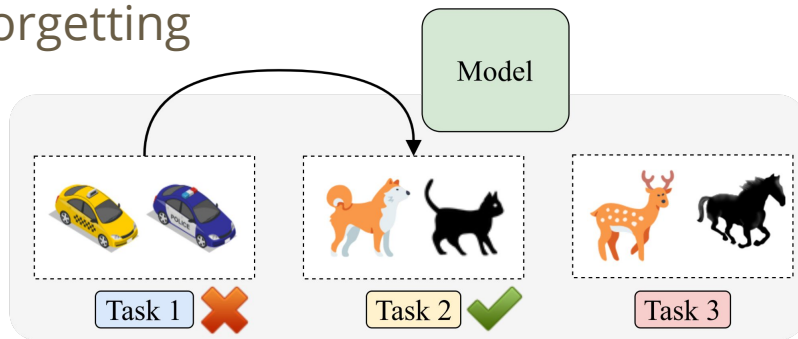
Condition: Model Sequentially Learn Different Task! (In Training Time)



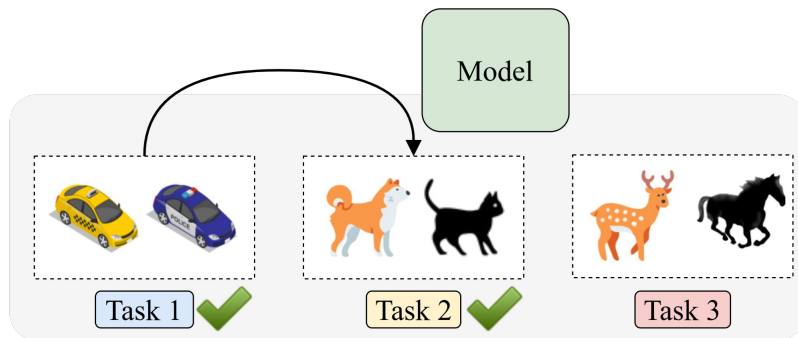
Introduction - LifeLong Learning

KeyPoint: Avoid Catastrophic Forgetting

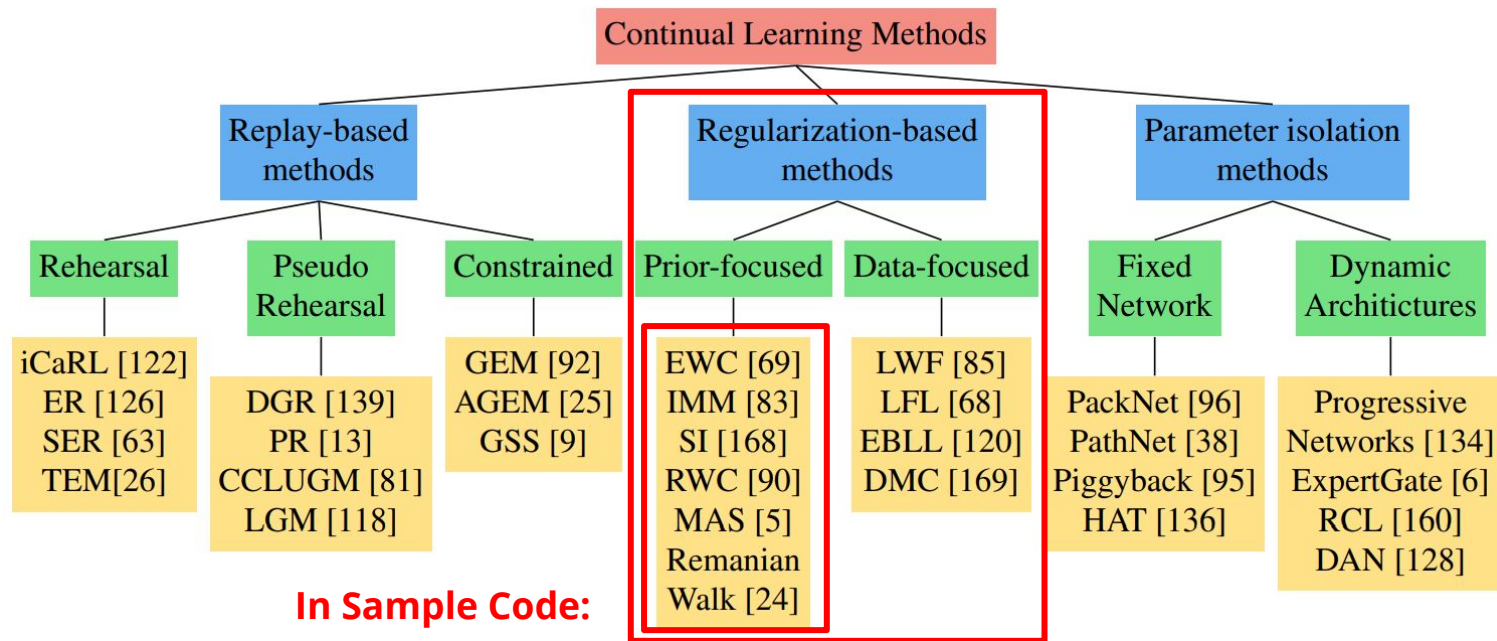
Catastrophic Forgetting



Avoid Catastrophic Forgetting



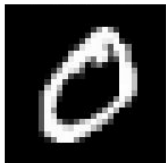
Introduction



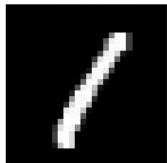
Dataset

Permuted MNIST

task: 1 label: 0



task: 1 label: 1



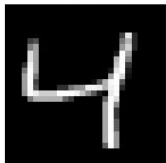
task: 1 label: 2



task: 1 label: 3



task: 1 label: 4



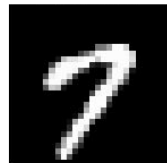
task: 1 label: 5



task: 1 label: 6



task: 1 label: 7



task: 1 label: 8



task: 1 label: 9



task: 2 label: 0



task: 2 label: 1



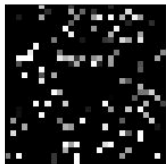
task: 2 label: 2



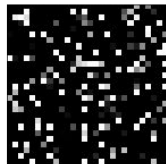
task: 2 label: 3



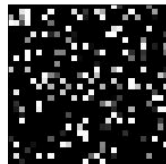
task: 2 label: 4



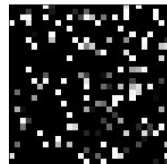
task: 2 label: 5



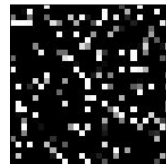
task: 2 label: 6



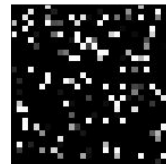
task: 2 label: 7



task: 2 label: 8



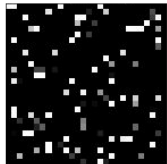
task: 2 label: 9



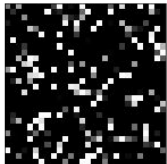
task: 3 label: 0



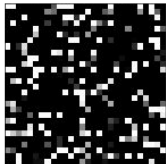
task: 3 label: 1



task: 3 label: 2



task: 3 label: 3



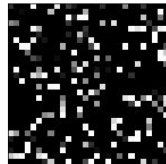
task: 3 label: 4



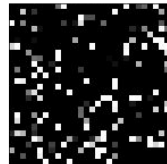
task: 3 label: 5



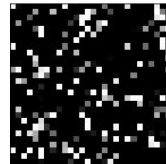
task: 3 label: 6



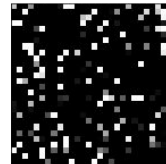
task: 3 label: 7



task: 3 label: 8



task: 3 label: 9

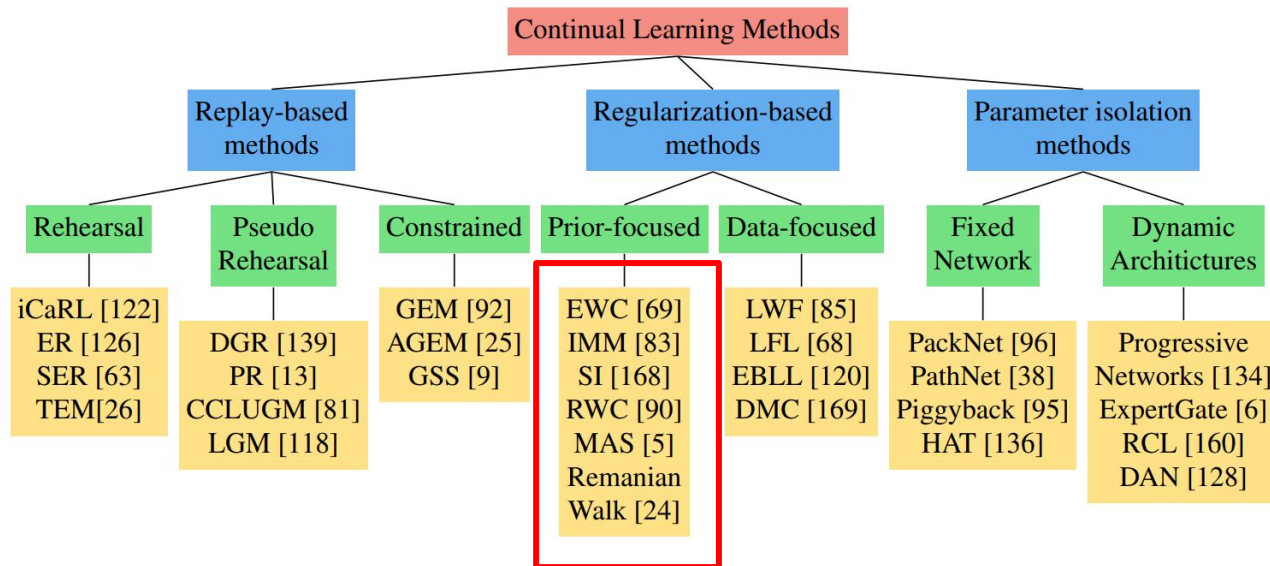


Sample Code - Training Detail

- 5 task / Each task has 10 epoches for training.
- Each method cost ~20 minutes for training model.
- [CoLab Link](#) (copy to your drive first! Don't simply run colab.)

Sample Code - Methods

- Baseline
- EWC
- MAS
- SI
- RWalk
- SCP



Sample Code - Guideline

- Utility
- Visualization
- Methods

Sample Code

- Utility
 - Permutation
 - Dataloader and Training Argument
 - Model
 - train
 - evaluate
 - evaluate metric
- Visualization
- Methods

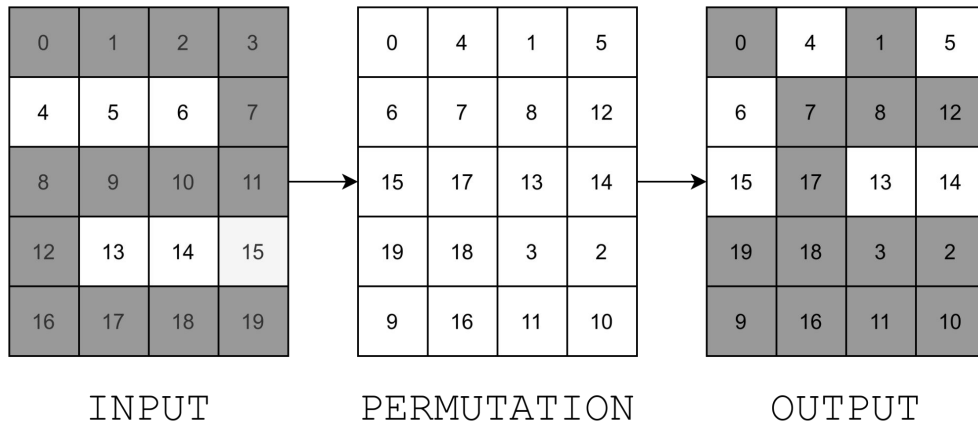
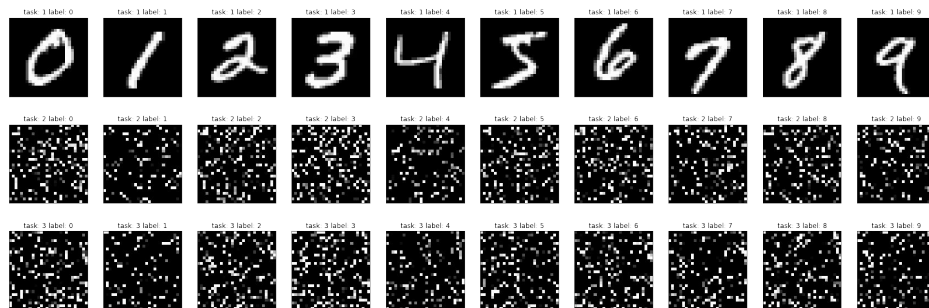
Sample Code

- Utility

- Permutation
- Dataloader and Training Argument
- Model
- train
- evaluate
- evaluate metric

- Visualization

- Methods



5 PERMUTATION = 5 TASK

Sample Code

Fixed model size!

- Utility
 - Permutation
 - Dataloader and
 - **Model**
 - train
 - evaluate
 - evaluate metric
- Visualization
- Methods

```
Model(  
    (fc1): Linear(in_features=784, out_features=1024, bias=True)  
    (fc2): Linear(in_features=1024, out_features=512, bias=True)  
    (fc3): Linear(in_features=512, out_features=256, bias=True)  
    (fc4): Linear(in_features=256, out_features=10, bias=True)  
    (relu): ReLU()  
)
```

Sample Code

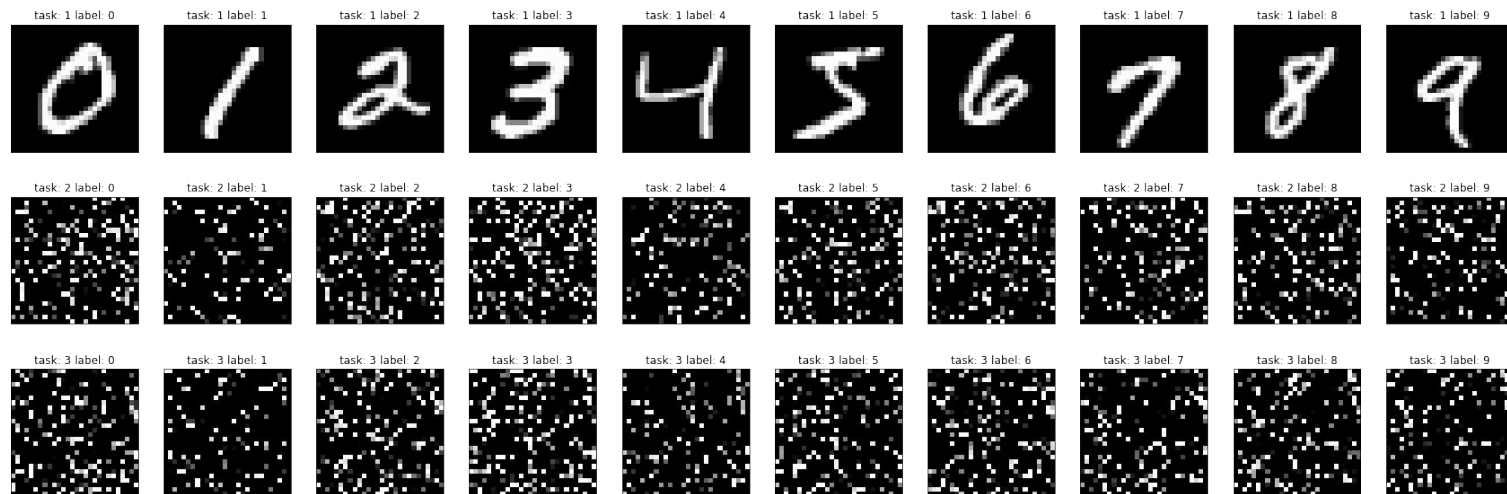
- Utility
 - Permutation
 - Dataloader and Training Argument
 - Model
 - train
 - evaluate
 - evaluate metric
- Visualization
- Methods

In k th task:

$$AverageAccuracy_k = \frac{1}{k} \sum_{j=1}^k a_{k,j}$$

Sample Code

- Utility
- Visualization
- Methods



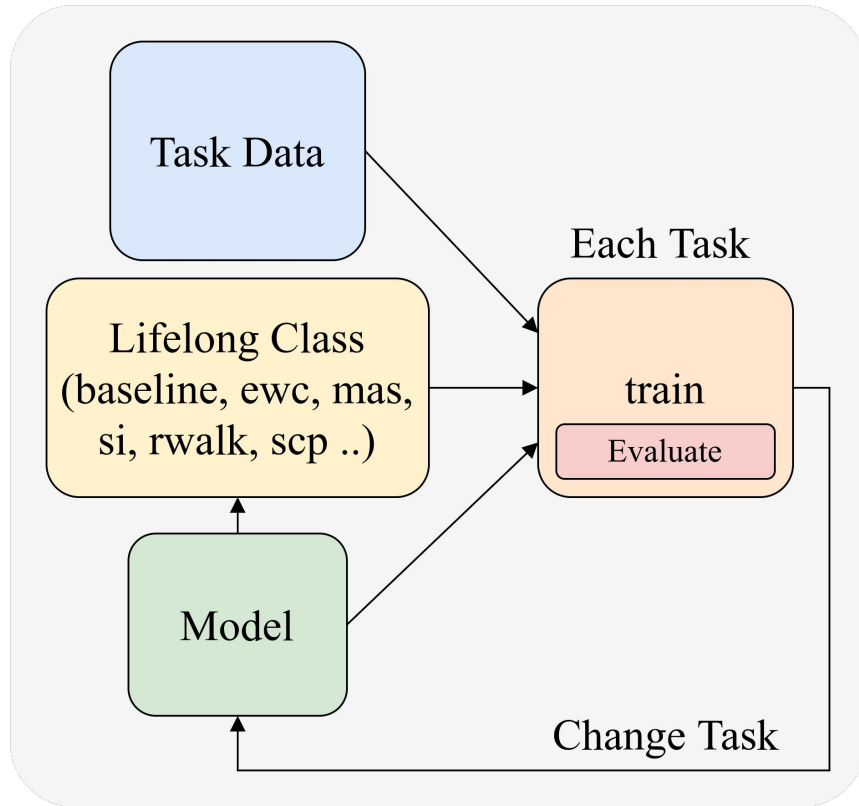
Sample Code - Guideline

- Utility
- Visualization
- Methods
 - Baseline
 - EWC
 - MAS
 - SI
 - RWalk
 - SCP

Baseline

Sequentially Train

Training Pipeline:



Lifelong learning class

Lifelong Class
(baseline, ewc, mas,
si, rwalk, scp ..)

```
6 class baseline(object):
7     """
8     baseline technique: do nothing in regularization term [initialize and all weight is zero]
9     """
10    def __init__(self, model, dataloaders, device):
11
12        self.model = model
13        self.dataloaders = dataloaders
14        self.device = device
15
16        self.params = {n: p for n, p in self.model.named_parameters() if p.requires_grad} #extract all parameters in models
17        self.p_old = {} # store current parameters
18        self._precision_matrices = self._calculate_importance() # generate weight matrix
19
20        for n, p in self.params.items():
21            self.p_old[n] = p.clone().detach() # keep the old parameter in self.p_old
22
23    def _calculate_importance(self):
24        precision_matrices = {}
25        for n, p in self.params.items(): # initialize weight matrix (fill zero)
26            precision_matrices[n] = p.clone().detach().fill_([0])
27
28        return precision_matrices
29
30    def penalty(self, model: nn.Module):
31        loss = 0
32        for n, p in model.named_parameters():
33            _loss = self._precision_matrices[n] * (p - self.p_old[n]) ** 2
34            loss += _loss.sum()
35        return loss
36
37    def update(self, model):
38        # do nothing
39        return
```

Lifelong learning class

Lifelong Class
(baseline, ewc, mas,
si, rwalk, scp ..)

train

```
6 class baseline(object):
7     """
8     baseline technique: do nothing in regularization term [initialize and all weight is zero]
9     """
10    def __init__(self, model, dataloaders, device):
11
12        self.model = model
13        self.dataloaders = dataloaders
14        self.device = device
15
16        self.params = {n: p for n, p in self.model.named_parameters() if p.requires_grad} #extract all parameters in models
17        self.p_old = {} # store current parameters
18        self._precision_matrices = self._calculate_importance() # generate weight matrix
19
20        for n, p in self.params.items():
21            self.p_old[n] = p.clone().detach() # keep the old parameter in self.p_old
22
23    def _calculate_importance(self):
24        precision_matrices = {}
25        for n, p in self.params.items(): # initialize weight matrix (fill zero)
26            precision_matrices[n] = p.clone().detach().fill_(0)
27
28        return precision_matrices
29
30    def penalty(self, model: nn.Module):
31        loss = 0
32        for n, p in model.named_parameters():
33            _loss = self._precision_matrices[n] * (p - self.p_old[n]) ** 2
34            loss += _loss.sum()
35        return loss
36
37    def update(self, model):
38        # do nothing
39        return
```

EWC - Elastic Weight Consolidation

1. You need to know how to generate Guardiance weight from EWC!
2. Do this method need to use label ?
3. Hint: (trace the class ewc and its calculate_importance function)

Paper Link: <https://arxiv.org/pdf/1612.00796.pdf>

MAS - Memory Aware Synapse

1. You need to know how to generate Guardianship weight from MAS!
2. Does this method need to use labels?
3. Hint: (trace the class mas and its calculate_importance function)

Paper Link: <https://arxiv.org/abs/1711.09601>

SI - Synaptic Intelligence

1. You need to know how to generate Guardianship weight from SI!
2. Does this method need to use labels?
3. Hint: (Accumulated loss change in each update step)

Paper Link: <https://arxiv.org/abs/1703.04200>, [Talk Slide](#)

SI - Main Idea

$$L(\theta) = L_2(\theta) + c \sum_i \Omega_i (\theta_i - \theta_{1,i}^*)^2$$

From learning trajectory

Parameter importance on-line from learning trajectory!

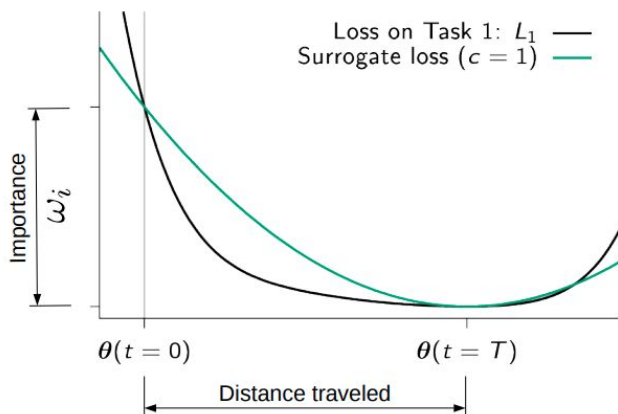
Picture comes from: [Talk Slide](#)

SI - Abstract

Leveraging per-parameter importance for continual learning

$$L(\theta) = L_2(\theta) + c \sum_i \Omega_i (\theta_i - \theta_{1,i}^*)^2$$

$$\Omega_i \equiv \frac{\omega_i}{(\Delta_i)^2 + \epsilon}$$



Picture comes from: [Talk Slide](#)

SI - Method

Total change in loss is given by the path integral over the gradient field

$$\begin{aligned}\int_C \mathbf{g}(\boldsymbol{\theta}(t)) d\boldsymbol{\theta} &= \int_{t_0}^{t_1} \mathbf{g}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\theta}'(t) dt = L(t_1) - L(t_0) \\ &= \sum_k \underbrace{\int_{t_0}^{t_1} g_k(t) \theta'_k(t) dt}_{\omega_k} \equiv - \sum_k \omega_k\end{aligned}$$

- Is a parameter-specific quantity
- Can be computed on-line during training (running sum)

\mathbf{g} : Gradient

$\boldsymbol{\theta}$: Parameters

$\boldsymbol{\theta}'$: Updates

Natural way of assigning credit for a global change to local parameters

$$L(t_1) - L(t_0) = - \sum_k \omega_k^\mu$$

Picture comes from: [Talk Slide](#)

RWalk - Rermanian Walk

1. Trace class rwalk and its update function!
2. Do this method need to use label ?
3. Hint:(The code is similar to two method which mentioned in sample code)

Paper Link: <https://arxiv.org/abs/1801.10112>

SCP - Sliced Cramer Preservation

1. Paper Link: <https://openreview.net/pdf?id=Blge3TNKwH>

2. Do this method need to use label ?

SCP - Main Idea

- Propose Distributed-based Distance to **prevent fast intransigence** and **avoid overestimate the importance of parameter.**

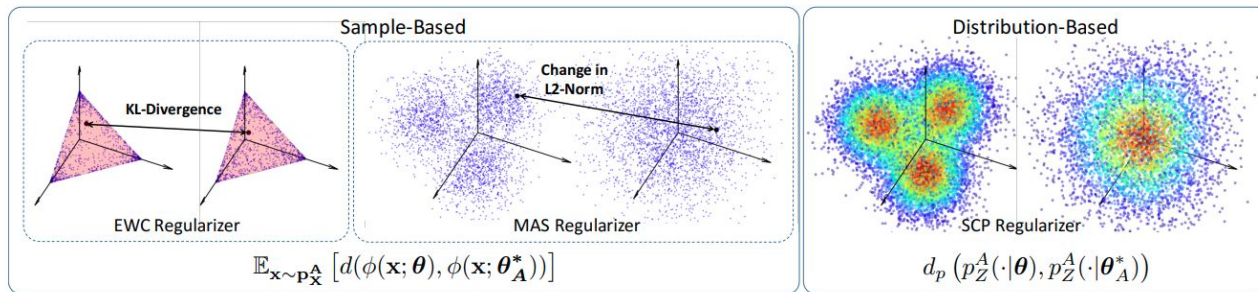
intransigence



Model do not want to learn new task,
and it just keep old task performance

SCP - Sliced Cramer Preservation (Hint)

1.



Paper Link: <https://openreview.net/pdf?id=BJge3TNKwH>

COOL Quiz

- 25 multiple choice questions
- Basic Concept & Dataset : 4 Questions
- Sample Code: 15 Questions
 - EWC: 3 Questions
 - MAS: 3 Questions
 - SI: 3 Questions
 - Remanian Walk: 3 Questions
 - Sliced Cramer Preservation: 3 Questions
- Other Methods & scenario: 6 Questions
 - ICaRL, LwF, GEM, DGR
 - Three Scenario

Grading

- All Questions (0.4pt)
- You have to choose ALL the correct answers for each question
- Warning: The answers in NTU Cool are not correct, NTU grading score before deadline is not the final grading score.
- Please do not select the choice depends on the grading score.

Submission

- No late submission!
- Pick the latest submission!
- Deadline: 2021/07/02 23:59
 - Warning: The answers in NTU Cool are not correct, NTU grading score before deadline is not the final grading score.
 - Please do not select the choice depends on the grading score.

Links

- [CoLab Link](#)
- NTU Cool Multiple Choice Question
- Warning: The answers in NTU Cool are not correct, NTU grading score before deadline is not the real grading score.
- Please do not select the choice depends on the grading score.

If any questions, you can ask us via...

- NTU COOL (recommended)
 - <https://cool.ntu.edu.tw/courses/4793>
- Email
 - ntu-ml-2021spring-ta@googlegroups.com
 - The title must begin with "[HW14]"
- TA hours
 - Each Monday 19:00~21:00 線上
 - Each Friday 13:30~14:20 線上
 - Each Friday During Class