Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence

Arslan Chaudhry et al.

Presented by Miloš Prágr

Pattern Recognition and Computer Vision Reading Group
Faculty of Electrical Engineering
Czech Technical University in Prague

January 14, 2020

Outline

- Incremental Learning
- Elastic Weight Consolidation
- Path Integral
- Riemannian Walk

Online learning approaches use training samples one by one, without knowing their number in advance, to optimise their internal cost function

Incremental learning refers to online learning strategies which work with limited memory resources

Gepperth and Hammer, *Incremental learning algorithms and applications*, **ESANN 2016**

Challenges of Incremental Learning

1. Online model parameter adaptation
2. Concept drift
3. Stability-plasticity dilema
4. Adaptive model complexity and meta-parameters
5. Efficient memory models
6. Model benchmarking

Gepperth and Hammer, *Incremental learning algorithms and applications*, **ESANN 2016**

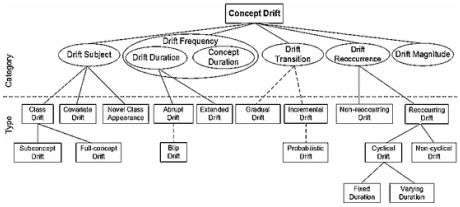
1. Online model parameter adaptation
2. Concept drift
3. Stability-plasticity dilema
4. Adaptive model complexity and meta-parameters
5. Efficient memory models
6. Model benchmarking

medium.com/starschema-blog

Fritzke, *A Growing Neural Gas Network Learns Topologies*, NIPS 1994

$$M_t \leftarrow update(M_{t-1}, (\boldsymbol{x}_t, y_t))$$

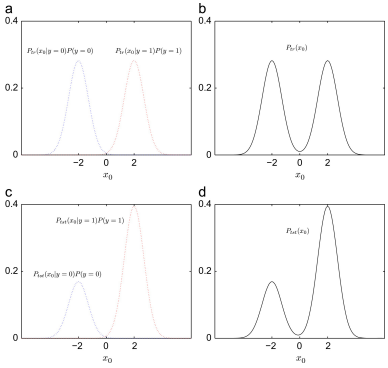
1. Online model parameter adaptation
2. Concept drift
3. Stability-plasticity dilema
4. Adaptive model complexity and meta-parameters
5. Efficient memory models
6. Model benchmarking



Webb et al., 2016

- The distribution underlying the data changes during learning

1. Online model parameter adaptation
2. Concept drift
3. Stability-plasticity dilema
4. Adaptive model complexity and meta-parameters
5. Efficient memory models
6. Model benchmarking

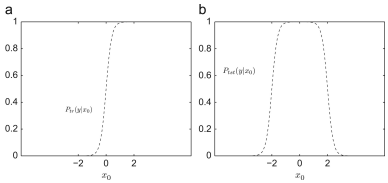


Moreno-Torres et al., 2012

Covariate shift of $p(\boldsymbol{x})$

- The distribution underlying the data changes during learning

1. Online model parameter adaptation
2. Concept drift
3. Stability-plasticity dilema
4. Adaptive model complexity and meta-parameters
5. Efficient memory models
6. Model benchmarking



Moreno-Torres et al., 2012

Concept shift of $p(y|\boldsymbol{x})$

- The distribution underlying the data changes during learning

- Quick updates cause old information to be forgotten equally quickly
- Gradual forgetting is natural component of both artificial and natural systems
- **Catastrophic forgetting** - completely disrupting or erasing previously learned information

French, *Catastrophic forgetting in connectionist networks*, **Trends in Cognitive Sciences 1999**

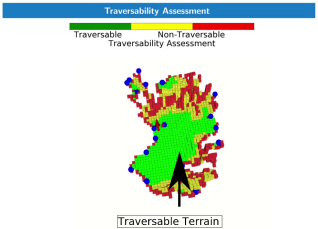
- It is impossible to estimate the model complexity in advance
- Minimal complexity increased by concept drift
- Maximal complexity bounded by resources

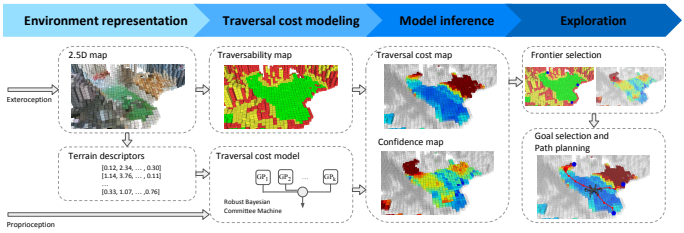
1. Online model parameter adaptation
2. Concept drift
3. Stability-plasticity dilema
4. Adaptive model complexity and meta-parameters
5. Efficient memory models
6. Model benchmarking

1. Online model parameter adaptation
2. Concept drift
3. Stability-plasticity dilema
4. Adaptive model complexity and meta-parameters
5. Efficient memory models
6. Model benchmarking

1. Incremental vs non-incremental
2. Incremental vs incremental

Motivation: Deployment of Incremental Learning





Online Incremental Learning of the Terrain Traversal Cost in Autonomous Exploration, **RSS 2019**

- **Forgetting:** catastrophically forgetting knowledge of previous tasks
- **Intransigence:** inability to update the knowledge to learn the new task

Chaudhry et al., *Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence*, **ECCV 2018**

- General setup: stream of tasks, each corresponding to a set of labels
- Let the dataset D_k corresponding to the k-th task be as follows

$$D_k = \{(\boldsymbol{x}_i^k, y_i^k)\}_{i=1}^{n_k},$$

  where k is the task identifier, $\boldsymbol{x}_i^k \in \mathcal{X}$ the inputs, and $y_i^k \in \mathcal{Y}$ the ground truth labels
- **Single-head evaluation** - the task identity k is unknown in testing
- **Multi-head evaluation** - the task identity k is given in testing

- General setup: stream of tasks, each corresponding to a set of labels
- Let the dataset $\mathcal{D}_k$ corresponding to the k-th task be as follows

$$\mathcal{D}_k = \{(\boldsymbol{x}_i^k, y_i^k)\}_{i=1}^{n_k},$$

  where k is the task identifier, $\boldsymbol{x}_i^k \in \mathcal{X}$ the inputs, and $y_i^k \in \mathcal{Y}$ the ground truth labels
- **Single-head evaluation** - the task identity k is unknown in testing
- **Multi-head evaluation** - the task identity k is given in testing

- **Accuracy** $a_{k,j}$ on the test set of the j-th task after training incrementally to task k is

$$a_{k,j} \quad s.t.\ j \leq k$$

- **Average accuracy** A_k at task k is defined as

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

- **Forgetting** f_j^k for the j-th task training up to task k is

$$f_j^k = \max_{l \in 1,\cdots,k-1} a_{l,j} - a_{k,j}, \quad s.t.\ j < k$$

- **Average forgetting** F_k at the k-th task is defined as

$$F_k = \frac{1}{k-1} \sum_{j=1}^{k-1} f_j^k$$

- **Backward transfer** - influence of learning task k has on performance of task $j < k$

  $f_j^k < 0$ implies **positive backward transfer**: the performance on a previous task was improved by learning additional tasks

- **Reference model accuracy** a_k^* is learned using the whole dataset as

$$\cup_{l=1}^{k} \mathcal{D}_l$$

- **Intransigence** I_k at the k-th task is defined as

$$I_k = a_k^* - a_{k,k}$$

- $I_j^k < 0$ implies **positive forward transfer**: learning incrementally up to task k positively influences model's knowledge about it

Outline

- Incremental Learning
- Elastic Weight Consolidation
- Path Integral
- Riemannian Walk

Motivation: continual learning in the neocortex relies on task-specific synaptic consolidation, where knowledge is encoded by rendering a proportion of synapses less plastic

Remember old tasks by selectively slowing down learning on the weights important for those tasks

Aim for fast learning rates on parameters unconstrained by the previous tasks and slow rate from crucial parameters

Kirkpatrick et al., *Overcoming catastrophic forgetting in neural networks*, **PNAS 2016**

Remember old tasks by selectively slowing down learning on the weights important for those tasks

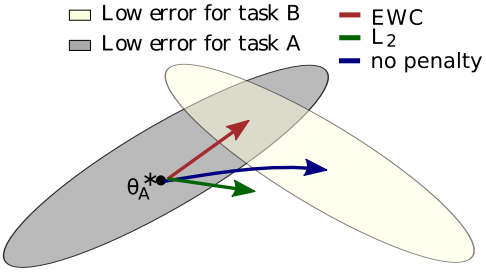
- Given dataset $\mathcal{D}$, select the configuration θ^* as

$$\theta^* = argmax_\theta \; p(\theta|\mathcal{D})$$

- Bayes gives the conditional probability $p(\theta|\mathcal{D})$ as

$$log \; p(\theta|\mathcal{D}) = log \; p(\mathcal{D}|\theta) + log \; p(\theta) - log \; p(\mathcal{D})$$

negative loss function $-\mathcal{L}(\theta)$



- Spliting the data into tasks A and B gives

$$log\ p(\theta|\mathcal{D}) = log\ p(\mathcal{D}_B|\theta) + log\ p(\theta|\mathcal{D}_A) - log\ p(\mathcal{D}_B)$$

negative loss function for task B $-\mathcal{L}_B(\theta)$

intractable posterior of task A

- Approximating the posterior as a Gaussian distribution given as

$$\mathcal{N}(\theta_A^*, (diag(F))^{-1})$$

MacKay, *A practical Bayesian framework for backpropagation networks*, **Neural Computing 1992**

where the precision $diag(F)$ is the diagonal of the Fisher information matrix F defined as

$$[F]_{ij} = E_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\left(\frac{\delta}{\delta\theta_i} log\ p_\theta(y|\boldsymbol{x})\right)\left(\frac{\delta}{\delta\theta_j} log\ p_\theta(y|\boldsymbol{x})\right)\right]$$

- The Fisher information measures sensitivity of function $f(x|\theta)$ to changes of θ
- Approximating the posterior as a Gaussian distribution given as

$$\mathcal{N}(\theta_A^*, (diag(F))^{-1})$$

MacKay, *A practical Bayesian framework for backpropagation networks*, **Neural Computing 1992**

where the precision $diag(F)$ is the diagonal of the Fisher information matrix F defined as

$$[F]_{ij} = E_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\left(\frac{\delta}{\delta\theta_i} log\ p_\theta(y|\boldsymbol{x})\right)\left(\frac{\delta}{\delta\theta_j} log\ p_\theta(y|\boldsymbol{x})\right)\right]$$

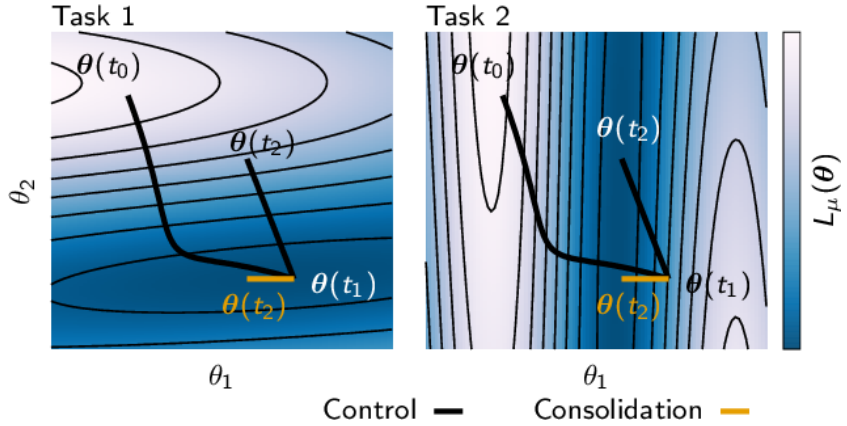
- The Fisher matrix is equivalent to the second derivative of the loss near a minimum and is always positive semidefinite

Pascanu and Bengio, *Revisiting natural gradient for deep networks*, **2013**

- The loss function to be minimized is

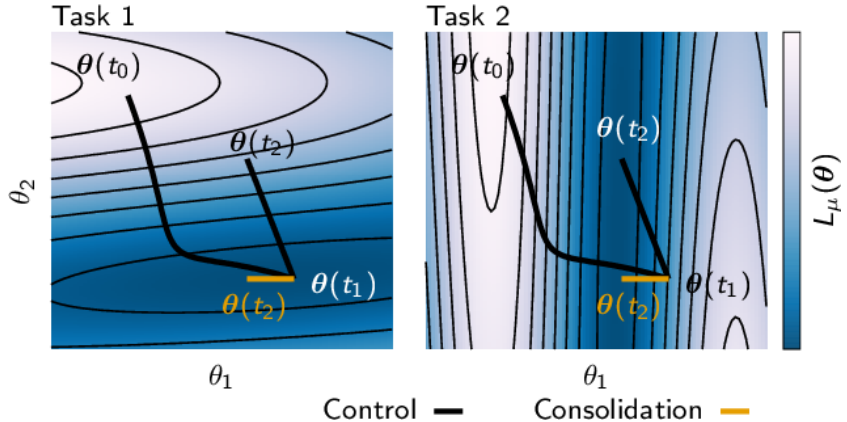
$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2}(F_{ii})(\theta_i - \theta_{A,i}^*)^2$$

- Incremental Learning
- Elastic Weight Consolidation
- Path Integral
- Riemannian Walk



- Accumulate parameter importance over the learning trajectory
- Avoid drastic changes to weights which were particularly influential in the past

Zenke et al., *Continual Learning Through Synaptic Intelligence*, **ICML 2017**



- When learning task μ, the importance Ω_k^μ of θ_k depends for each previous task ν on
  - The contribution over the entire training trajectory, i.e., ω_k^ν where $\nu < \mu$
  - How far θ_k moved when learning ν: $\Delta_k^\nu = \theta_k(t^\nu) - \theta_k(t^{\nu-1})$

  We want large gains (contribution) for little work (motion)

- When learning task μ, the importance Ω_k^μ of θ_k depends for each previous task ν on
  - The contribution over the entire training trajectory, i.e., ω_k^ν where $\nu < \mu$
  - How far θ_k moved when learning ν: $\Delta_k^\nu = \theta_k(t^\nu) - \theta_k(t^{\nu-1})$

  which leads to

$$\Omega_k^\mu = \sum_{\nu<\mu} \frac{\omega_k^\nu}{(\Delta_k^\nu)^2 + \epsilon}$$

- The surrogate loss for two tasks is used as

$$\widetilde{\mathcal{L}}_\mu = \mathcal{L}_\mu + c \sum_k \Omega_k^\mu (\theta_k(t^{\mu-1}) - \theta_k)^2$$

Why is this a path integral? And what is ω_k^ν?

- For infinitesimal parameter update $\delta(t)$ the loss change is approximated as loss gradient

$$\mathcal{L}(\theta(t) + \delta(t)) - \mathcal{L}(\theta(t)) \approx \sum_k g_k(t)\delta_k(t)$$

- Consider the path integral of the loss gradient along the parameter trajectory

$$\int_C \boldsymbol{g}(\boldsymbol{\theta}(t))d\boldsymbol{\theta} = \int_{t_0}^{t_1} \boldsymbol{g}(\boldsymbol{\theta}(t))\boldsymbol{\theta}'(t)dt,$$

and its decomposition as a sum over individual parameters

$$\int_{t^{\mu-1}}^{t^\mu} \boldsymbol{g}(\boldsymbol{\theta}(t))\boldsymbol{\theta}'(t)dt = \sum_k \int_{t^{\mu-1}}^{t^\mu} g_k(\theta(t))\theta_k'(t)dt \equiv -\sum_k \omega_k^\mu$$

ω_k^μ may be approximated as the running sum of the product of the gradient $g_k(t)$

Outline

- Incremental Learning
- Elastic Weight Consolidation
- Path Integral
- Riemannian Walk

1. Idea: Fisher Matrix may be infeasible for large number of parameters in deep neural networks
2. Idea: Ground the approach using KL-divergence between network likelihoods instead of empirical Fisher matrix

   Second-order Taylor approximation of the KL divergence is

$$D_{KL}(p_\theta \| p_{\theta+\Delta\theta}) \approx \frac{1}{2} \sum_{i=0}^{P} F_{\theta_i} \Delta\theta_i^2 \quad s.t. \quad \Delta\theta \to 0$$

3. Idea: combine the EWC update with PI parameter importance

- EWC stores the Fisher matrix for each task independently and uses joint regularization
- To estimate the Fisher matrix, pass over the dataset is needed for each task
- A single diagonal Fisher matrix may be constructed using a batch approach as follows

$$F^t = \alpha F'^t + (1 - \alpha) F^{t-1}$$

Chaudhry et al., *Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence*, **ECCV 2018**

- The EWC loss for the k-th task is generalized as

$$\widetilde{\mathcal{L}}^k(\theta) = \mathcal{L}^k(\theta) + \lambda D_{KL}(p_{\theta^{k-1}}, p_\theta)$$

$$D_{KL}(p_\theta \| p_{\theta+\Delta\theta}) \approx \frac{1}{2}\sum_{i=0}^{P} F_{\theta_i}\Delta\theta_i^2$$

whereas the EWC loss is

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

- EWC++ batch updated Fisher is used

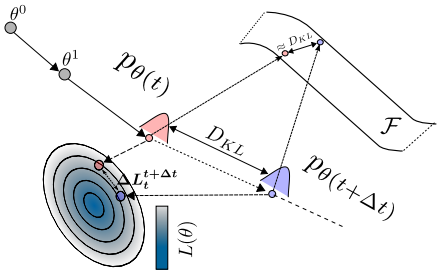
$$F^t = \alpha F'^t + (1-\alpha)F^{t-1}$$

- Path integral parameter importance: We want large gains (contribution) for little work (motion)

$$\Omega_k^\mu = \sum_{\nu<\mu} \frac{\omega_k^\nu}{(\Delta_k^\nu)^2 + \epsilon}$$

- RWalk parameter importance: Large loss improvement for small change of distribution

$$s_{t_1}^{t_2} = \sum_{t=t_1}^{t_2} \frac{\Delta \mathcal{L}_t^{t+\Delta t}(\theta_i)}{\frac{1}{2} F_{\theta_i}^t \Delta(\theta_i(t))^2 + \epsilon}$$



- The EWC loss for the k-th task is generalized as

$$\widetilde{\mathcal{L}}^k(\theta) = \mathcal{L}^k(\theta) + \lambda D_{KL}(p_{\theta^{k-1}}, p_\theta)$$

- The path integral parameter change is interpreted as a change in the model distribution, and the importance change between t_1 and t_2 is defined as

$$s_{t_1}^{t_2} = \sum_{t=t_1}^{t_2} \frac{\Delta \mathcal{L}_t^{t+\Delta t}(\theta_i)}{\frac{1}{2} F_{\theta_i}^t \Delta(\theta_i(t))^2 + \epsilon}$$

- The combined Riemannian walk loss function is

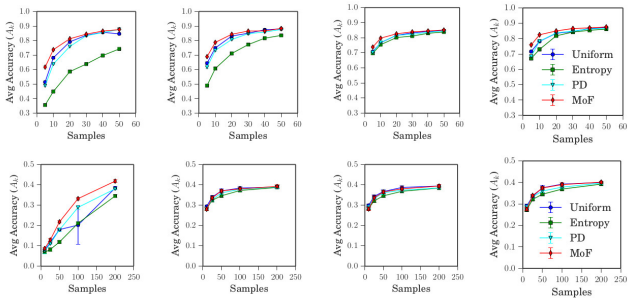
$$\widetilde{\mathcal{L}}^k(\theta) = \mathcal{L}^k(\theta) + \lambda \sum_{i=1}^{P} (F_{\theta_i^{k-1}} + s_{t_0}^{t_{k-1}}(\theta_i))(\theta_i - \theta_i^{k-1})^2$$

where

$$s_{t_0}^{t_{k-1}}(\theta_i)) = \frac{1}{2}(s_{t_0}^{t_{k-2}}(\theta_i) + s_{t_{k-2}}^{t_{k-1}}(\theta_i))$$

Note, the F and s should be normalized.

- Observation: learning task k with only $\mathcal{D}_k$ leads to poor accuracy in single head
- Avoid confusion by adding a set representative samples of previous tasks
- Strategies
  - Uniform
  - Plane-distance based
  - Entropy-based (soft-max)
  - Mean-of-features

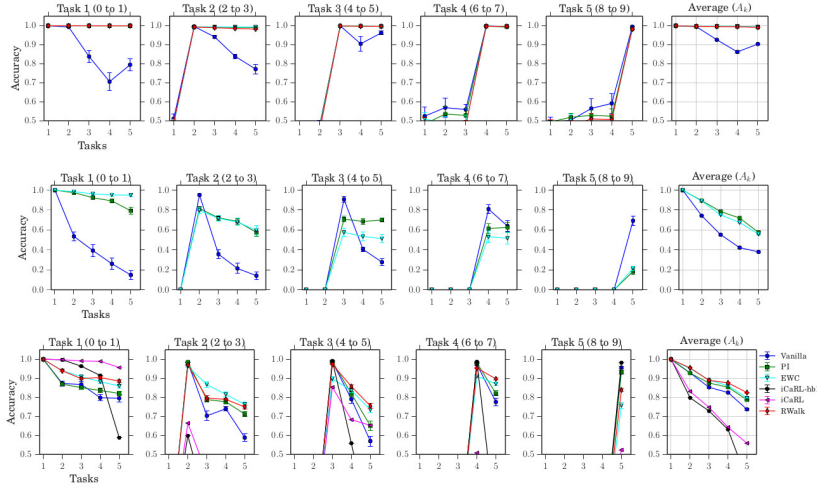


Riemannian Walk: MNIST and CIFAR-100 Tests

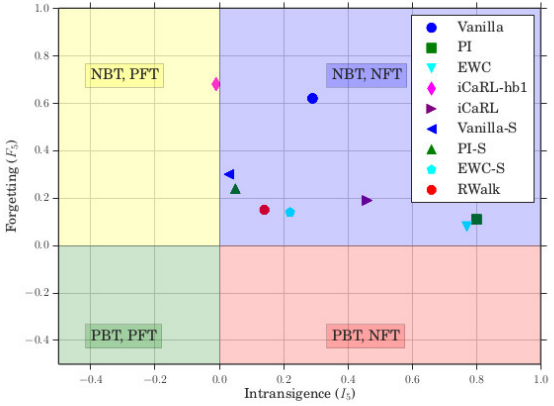
Methods	MNIST				CIFAR			
	Multi-head Evaluation							
	λ	$A_5(\%)$	F_5	I_5	λ	$A_{10}(\%)$	F_{10}	I_{10}
Vanilla	0	90.3	0.12	6.6×10^{-4}	0	44.4	0.36	0.02
EWC	75000	**99.3**	0.001	0.01	3×10^6	72.8	0.001	0.07
PI	0.1	**99.3**	0.002	0.01	10	73.2	0	0.06
RWalk (Ours)	1000	**99.3**	0.003	0.01	1000	**74.2**	0.004	0.04
	Single-head Evaluation							
Vanilla	0	38.0	0.62	0.29	0	10.2	0.36	-0.06
EWC	75000	55.8	0.08	0.77	3×10^6	23.1	0.03	0.17
PI	0.1	57.6	0.11	0.8	10	22.8	0.04	0.2
iCaRL-hb1	-	36.6	0.68	-0.01	-	7.4	0.40	0.06
iCaRL	-	55.8	0.19	0.46	-	9.5	0.11	0.35
Vanilla-S	0	73.7	0.30	0.03	0	12.9	0.64	-0.3
EWC-S	75000	79.7	0.14	0.22	15×10^5	33.6	0.27	-0.05
PI-S	0.1	78.7	0.24	0.05	10	33.6	0.27	-0.03
RWalk (Ours)	1000	**82.5**	0.15	0.14	500	**34.0**	0.28	-0.06

MNIST tasks: $\{0,1\}, \cdots, \{8,9\}$

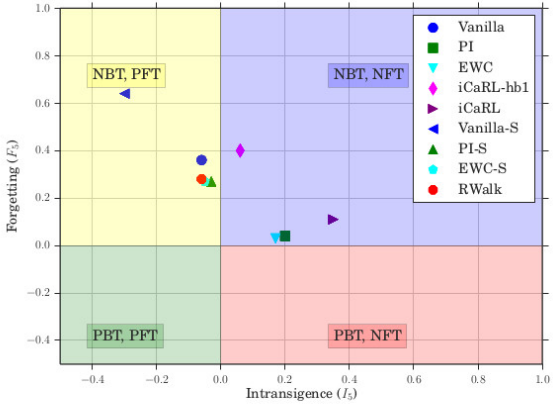
CIFAR-100 tasks: $\{0-9\}, \cdots, \{90-99\}$



Top to bottom: multi-head, single-head, single-head with samples



(a) MNIST



(b) CIFAR