Day 4

Speaker: Kartek Alahari, Inria

Title: Incremental Visual Learning

Incremental Learning ?

- Continual learning
- Lifelong learning
- Sequential learning
- Never-ending Learning

Synonyms

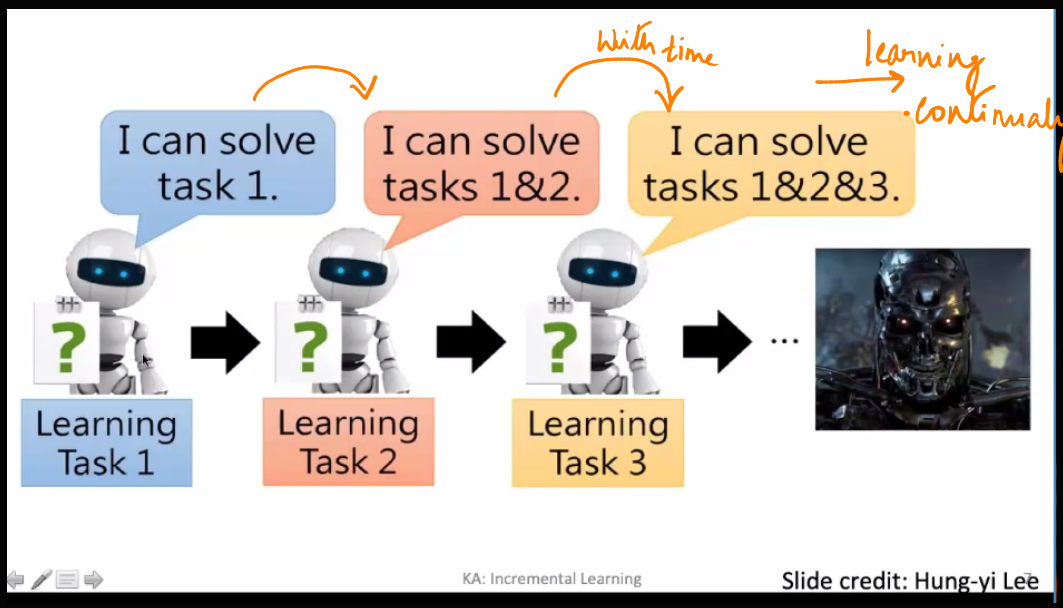
An Incremental Learning Scenario

- Growing up in India

Overtime

An Incremental Learning Scenario

- And then during travels



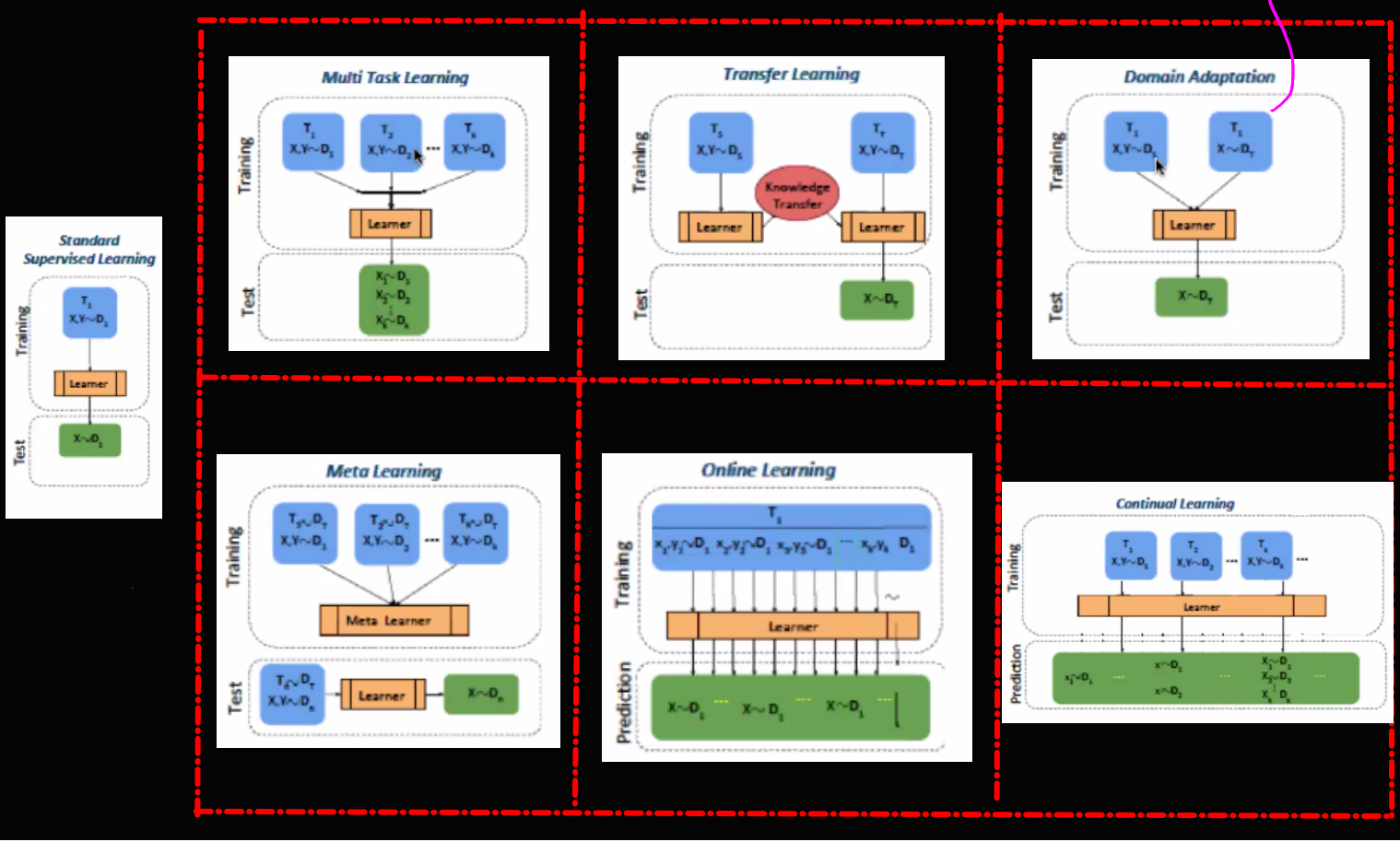
Standard Machine Learning

TRAIN – VALIDATION – TEST

All sampled from the same distribution
-> benchmarks and academic datasets ☺ ✓✓ Able to do
-> real-world systems ↝ Under Research
-> embodied learning → Not yet

KA: Incremental Learning
Slide credit: T. Tuytelaars

Don't have all labelled images.



Incremental Learning Setup

- Task-incremental learning (when T_1, T_2 are different tasks)
- Class-incremental learning
- Domain-incremental learning

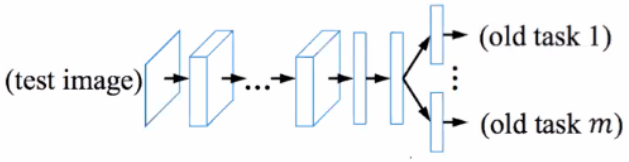
Incremental Learning

- A classical problem in machine learning, e.g.,
  [Carpenter et al. '92, Cauwenberghs and Poggio '00, Polikar et al. '01, Schlimmer and Fisher '86, Thrun '96]

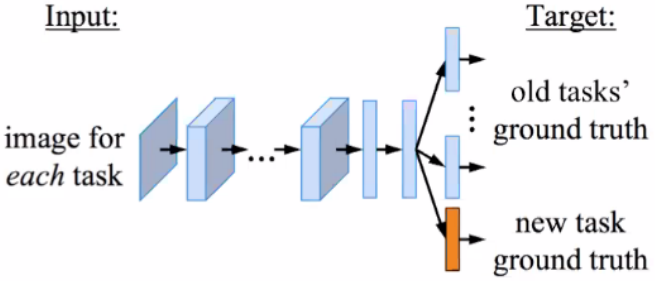
- Some methods
  - Zero-shot learning, e.g., [Lampert et al. '13]
    No training step for unseen classes
  - Continuously update the training set, e.g., [Chen et al. '13]
    Keep data and retrain
  - Use a fixed data representation, e.g., [Mensink et al. '13]
    Simplify the learning problem

Brute Force Solution (non-incremental)

Original model



Joint training



"golden" baseline

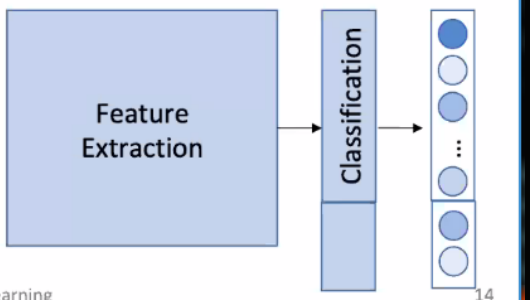
- random initialize + train
- fine-tune
- unchanged

Figures from [Li and Hoiem 2016]

Brute Force Solution (non-incremental)

Retrain full model with both old and new data

- Computationally expensive
- Needs access to old data
  - Storage capacity limitations
  - Privacy issues
  - Scalability issues

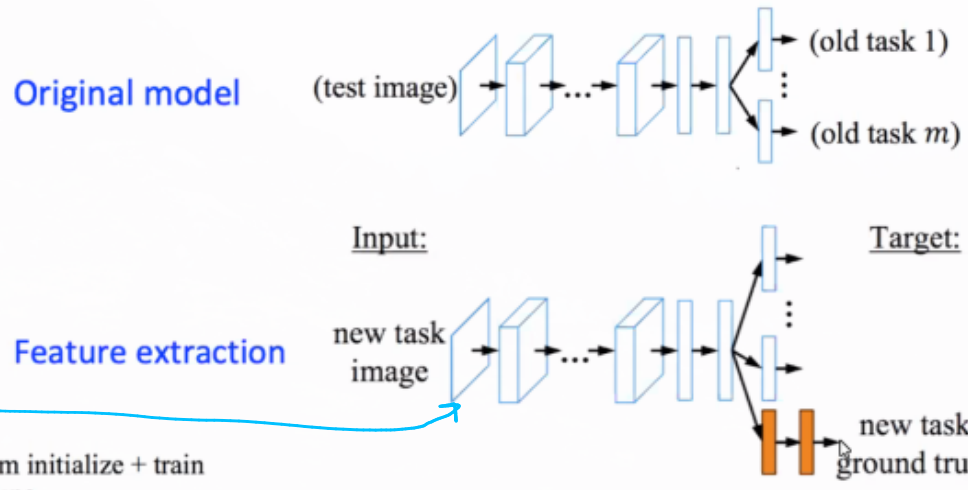


Slide credit: T. Tuytelaars

Why not brute force ?

- No access to all the data
- Can not store all the data
- Access to only a previously learned model, e.g., trained by others

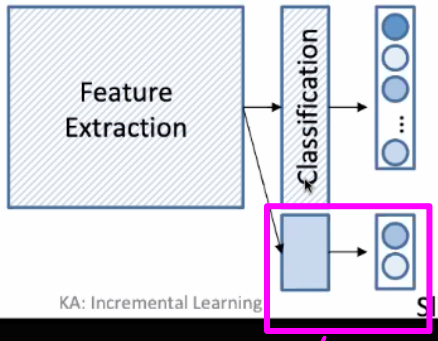
Naïve Solution 1



Figures from [Li and Hoiem 20

Naïve Solution 1

- Finetune only last layer using new data only
- Suboptimal Results

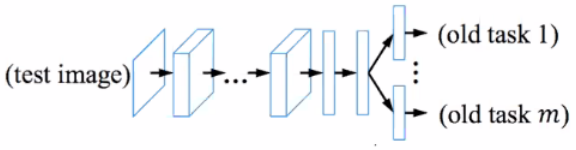


learn only this part

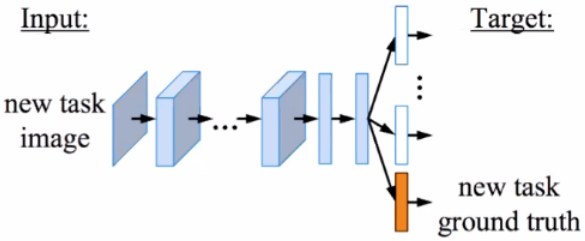
Slide credit: T. Tuytelaars

Naïve Solution 2

Original model



Fine tuning

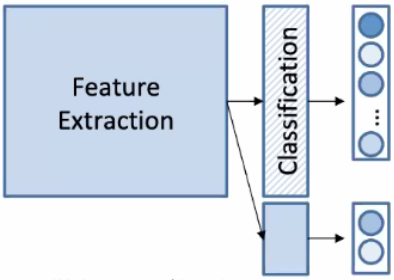


random initialize + train
fine-tune
unchanged

Figures from [Li and Hoiem 2016]

Naïve Solution 2

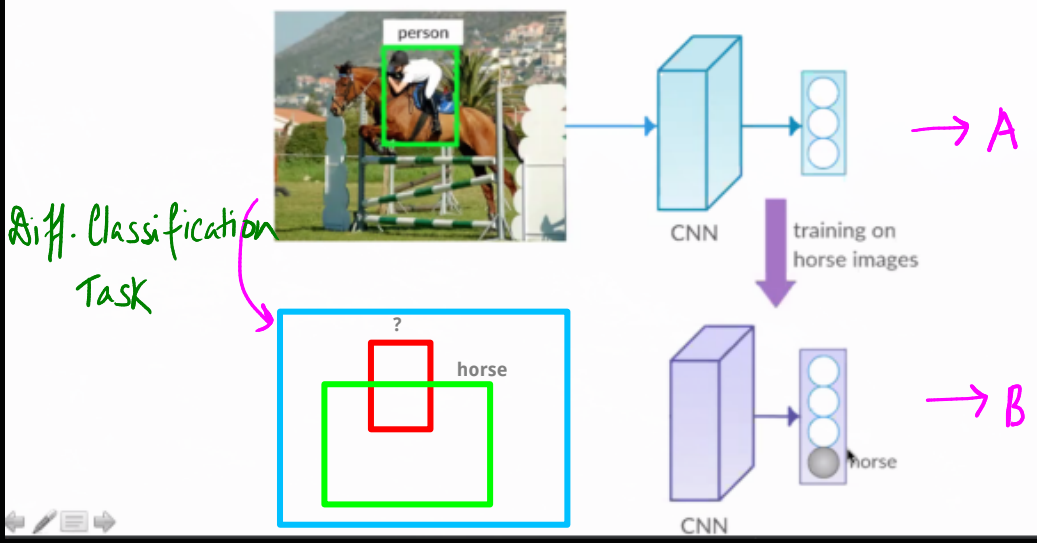
- Finetune the network using new data only
- Leads to catastrophic forgetting



Slide credit: T. Tuytelaars

Incremental Learning: Computer Vision Task

{To be updated}



21

How well does network B perform ?

method	old	new	all
A(1-10)	65.8	-	-
+B(11-20)	12.8	64.5	**38.7**
A(1-20)	68.4	71.3	**69.8**

Training with the **initial** set of classes

Training with the **new** set of classes

Baseline, i.e., training with all the classes

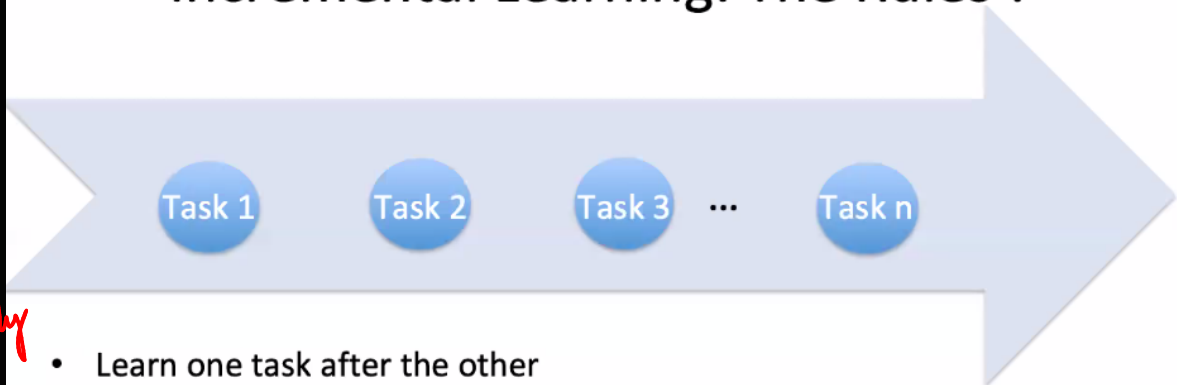
No guidance for retaining the old classes

[Catastrophic forgetting: McCloskey and Cohen 1989, Ratcliff 1990]

26

* Not retaining what is learnt earlier (∴ low accuracy)

Incremental Learning: The Rules !



Practically *

- Learn one task after the other
- Without storing (many) data from previous tasks
- Without memory footprint growing (significantly) over time
- Without (completely) forgetting old tasks

Slide credit: T. Tuytelaars

What else will we see today?

- Flavour of different approaches:
  1. **Regularization based**: LwF, EBLL, EWC, SI, MAS, IMM, …
  2. Rehearsal / Replay: iCaRL, DGR, GEM, …
  3. Architecture based: PackNet, progressive nets , HAT, …
- More than classification?
- Takeaways

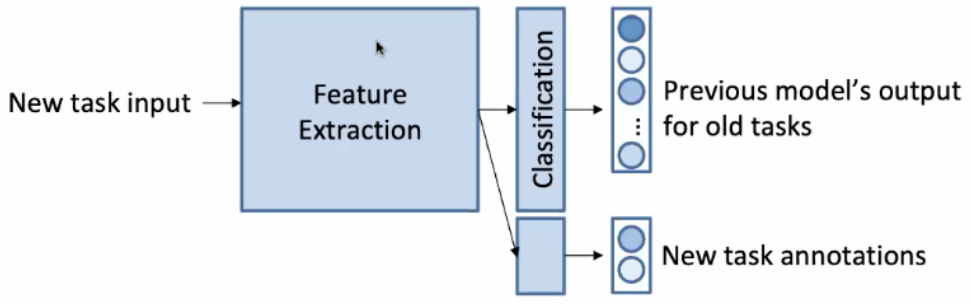
Regularization-based Models

- When training a new task,
  - add a regularization term to the loss
  - i.e., term to penalize catastrophic forgetting
- R1: data-focused methods
- R2: model/prior-focused methods

Slide credit: T. Tuytelaars

Data-focused Regularization: Learning without Forgetting

- Knowledge distillation loss
  - i.e., preservation of responses



[Li & Hoiem 2016]

Slide credit: T. Tuytelaars

Data-focused Regularization: Learning without Forgetting

Simple method; good results for related tasks

Poor results for unrelated tasks

? Need to store the old model

[Li & Hoiem 2016]

Model-focused Regularization

- Penalize changes to 'important' parameters

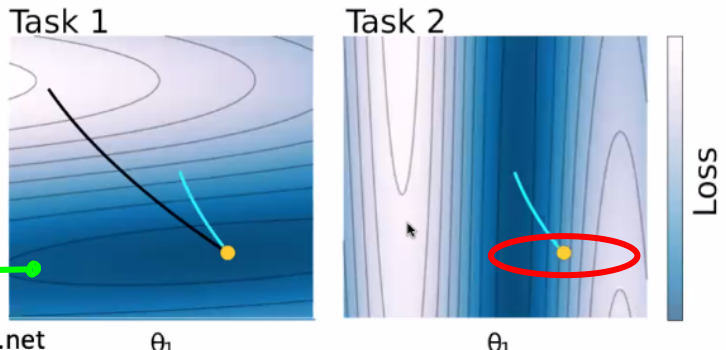
$$\mathcal{L}(\theta) = \underbrace{\mathcal{L}_B(\theta^n)}_{\text{Loss on new task(s)}} + \alpha \sum_k \underbrace{\lambda_k(\theta_k^n - \theta_k^{n-1})^2}_{\text{Regularization}}$$

Different variants possible for "importance" and regularization

Model-focused Regularization

- Elastic weight consolidation [Kirkpatrick et al., 2017]
  - Indiv. penalty for each previous task $\sum_k \sum_{i<n} \lambda_k^{n-i}(\theta_k^n - \theta_k^{n-i})^2$
  - Fisher information matrix for λ

→ To read!



loss

local minima

Figure credit: fzenke.net

Model-focused Regularization

- Elastic weight consolidation [Kirkpatrick et al., 2017]
  - Indiv. penalty for each previous task $\sum_k \sum_{i<n} \lambda_k^{n-i} (\theta_k^n - \theta_k^{n-i})^2$
  - Fisher information matrix for λ

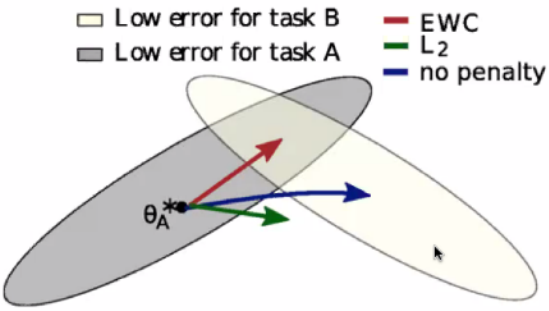


Figure from paper

Model-focused Regularization

- Elastic weight consolidation [Kirkpatrick et al., 2017]
  - Indiv. penalty for each previous task $\sum_k \sum_{i<n} \lambda_k^{n-i} (\theta_k^n - \theta_k^{n-i})^2$
  - Fisher information matrix for λ

Agnostic to architecture; Good results empirically

Only valid locally

?

Need to store importance weights

Model-focused Regularization

- Memory aware synapses [Aljundi et al., 2018]
  - Considers only the previous task $\sum_k \lambda_k (\theta_k^n - \theta_k^{n-1})^2$
  - Change in gradients for λ

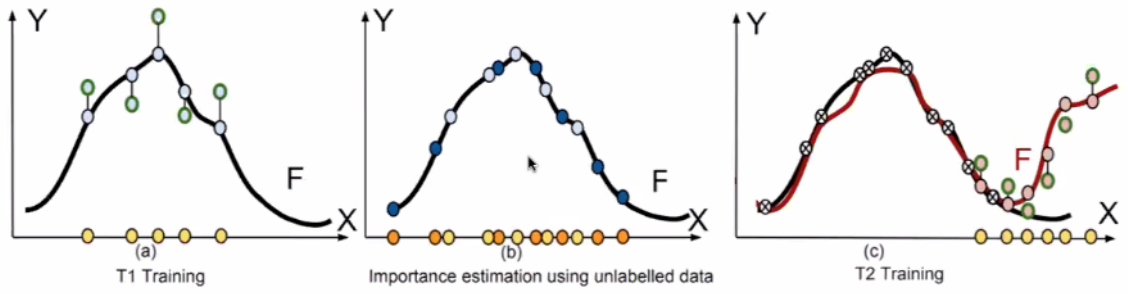


Figure from paper

Model-focused Regularization

- Memory aware synapses [Aljundi et al., 2018]
  - Considers only the previous task $\sum_k \lambda_k(\theta_k^n - \theta_k^{n-1})^2$
  - Change in gradients for λ

Agnostic to architecture; Leverages data & output

Only valid locally

?

Need to store importance weights

Other examples of model focused regularization other than EWS & MAS

① Synaptic Intelligence

② Moment Matching

③ PackNet.

What else will we see today?

- Flavour of different approaches:
  1. Regularization based: LwF, EBLL, EWC, SI, MAS, IMM, …
  2. **Rehearsal / Replay**: iCaRL, DGR, GEM, …
  3. Architecture based: PackNet, progressive nets , HAT, …
- More than classification?
- Takeaways

Rehearsal / Replay-based methods

- Store a couple of examples from previous tasks
- Or produce samples from a generative model
- But
  - How many?
  - How to select them?
  - How to use them?

iCaRL: Incremental classifier and representation learning

- Selects samples that are closest to the feature mean of each class
- Knowledge distillation loss [Hinton et al.'14]
- Clever use of available memory (see the following)

[Rebuffi et al. 2017]

→ To Study.
{To be updated}
{iCARL Algorithm}
*

iCaRL: Incremental classifier and representation learning

Split the problem into:
- learning features, and then
- using NCM classifier

Algorithm iCaRL INCREMENTALTRAIN

input $X^s, \ldots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \ldots, P_{s-1})$ // current exemplar sets
$\Theta \leftarrow$ UPDATEREPRESENTATION$(X^s, \ldots, X^t; \mathcal{P}, \Theta)$
$m \leftarrow K/t$ // number of exemplars per class
for $y = 1, \ldots, s-1$ **do**
$P_y \leftarrow$ REDUCEEXEMPLARSET(P_y, m)
end for
for $y = s, \ldots, t$ **do**
$P_y \leftarrow$ CONSTRUCTEXEMPLARSET(X_y, m, Θ)
end for
$\mathcal{P} \leftarrow (P_1, \ldots, P_t)$ // new exemplar sets

[Rebuffi et al. 2017]

iCaRL: Incremental classifier and representation learning

Algorithm iCaRL CLASSIFY

input x // image to be classified
require $\mathcal{P} = (P_1, \ldots, P_t)$ // class exemplar sets
require $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$ // feature map
for $y = 1, \ldots, t$ **do**
$\mu_y \leftarrow \frac{1}{|P_y|} \sum_{p \in P_y} \varphi(p)$ // mean-of-exemplars
end for
$y^* \leftarrow \underset{y=1,\ldots,t}{\operatorname{argmin}} \|\varphi(x) - \mu_y\|$ // nearest prototype
output class label y^*

[Rebuffi et al. 2017]

iCaRL: Incremental classifier and representation learning [Rebuffi et al.'17]

Algorithm iCaRL UPDATEREPRESENTATION

input $X^s, \ldots, X^t$ // training images of classes $s, \ldots, t$
require $\mathcal{P} = (P_1, \ldots, P_{s-1})$ // exemplar sets
require Θ // current model parameters
// form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s,\ldots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\ldots,s-1} \{(x,y) : x \in P^y\}$$

// store network outputs with pre-update parameters:
for $y = 1, \ldots, s-1$ **do**
  $q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
end for
run network training (*e.g.* BackProp) with loss function

$$\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[\sum_{y=s}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y\neq y_i} \log(1-g_y(x_i)) + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1-q_i^y) \log(1-g_y(x_i)) \Big]$$

that consists of *classification* and *distillation* terms.

Classification loss

Distillation loss: Comparing old vs new

[Rebuffi et al. 2017]

iCaRL: Incremental classifier and representation learning

Clever use of available memory

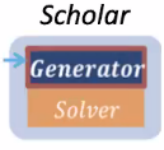
Potential issues with storing data, e.g., privacy

Limited by the memory capacity (the more the better)

[Rebuffi et al. 2017]

Deep Generative Replay

- The model "Scholar" is composed of:
  - a generator + a solver (classifier)



- The generator and the solver are updated in every incremental step

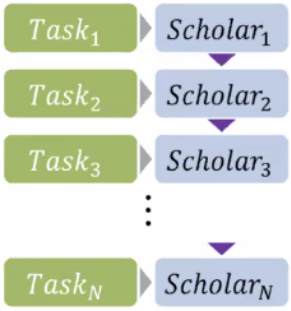
* Just like GAN's generator & discriminator

[Shin et al. 2017]
Figure from the paper

Deep Generative Replay

Training procedure:

- At task *t*, we train a new Scholar
  - with data from the task *t*, and
  - data generated by the previously trained Scholar at task *t-1*

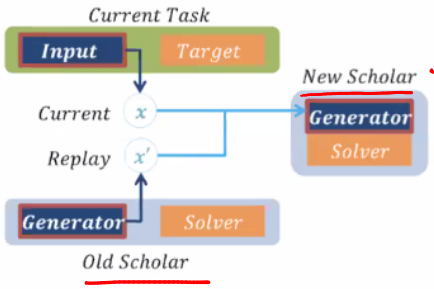


[Shin et al. 2017]
Figure from the paper

Deep Generative Replay

Training procedure (Generator):

- With data from task *t*, and
- data generated by the previously trained Scholar for task *t-1*

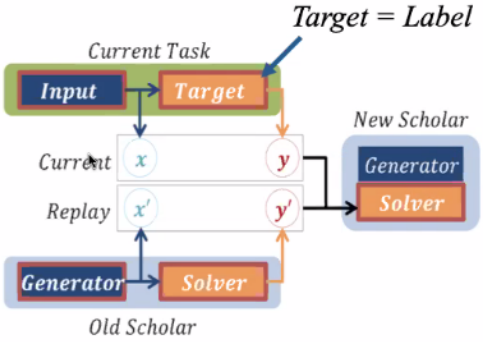


[Shin et al. 2017]
Figure from the paper

Deep Generative Replay

Training procedure (Solver):

- With data from task *t*, and
- Data from generator and solver of the previously trained Scholar for task *t-1*



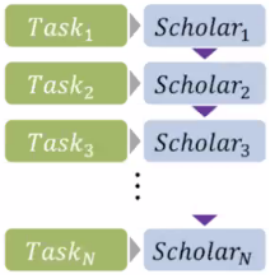
[Shin et al. 2017]
Figure from the paper

Deep Generative Replay

Avoids memory issues

Accumulation of errors

No control over the class of the generated samples

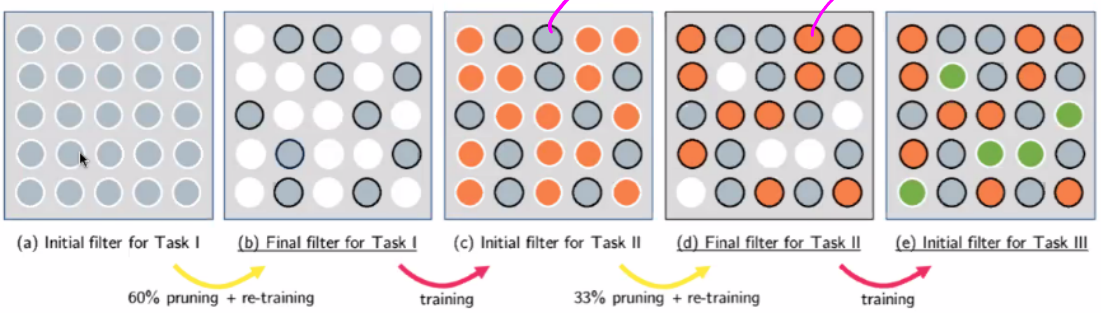


[Shin et al. 2017]
Figure from the paper

What else will we see today?

- Flavour of different approaches:
  1. Regularization based: LwF, EBLL, EWC, SI, MAS, IMM, …
  2. Rehearsal / Replay: iCaRL, DGR, GEM, …
  3. **Architecture based**: PackNet, progressive nets , HAT, …
- More than classification?
- Takeaways

Architecture-based



PackNet [Mallya & Lazebnik'17]
Figure from the paper

But needs to know prior the total no. of tasks.

Architecture-based

Fixed memory consumption

Needs the total number of tasks

Avoids forgetting

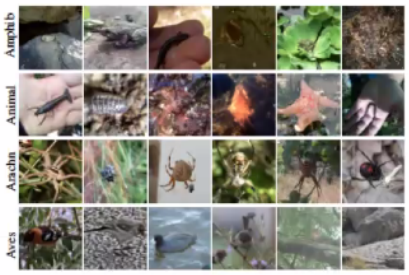
PackNet [Mallya & Lazebnik'17]

A Comparative Analysis

- TinyImagenet: small, balanced, class-incremental
- iNaturalist: large-scale, unbalanced, task-incremental

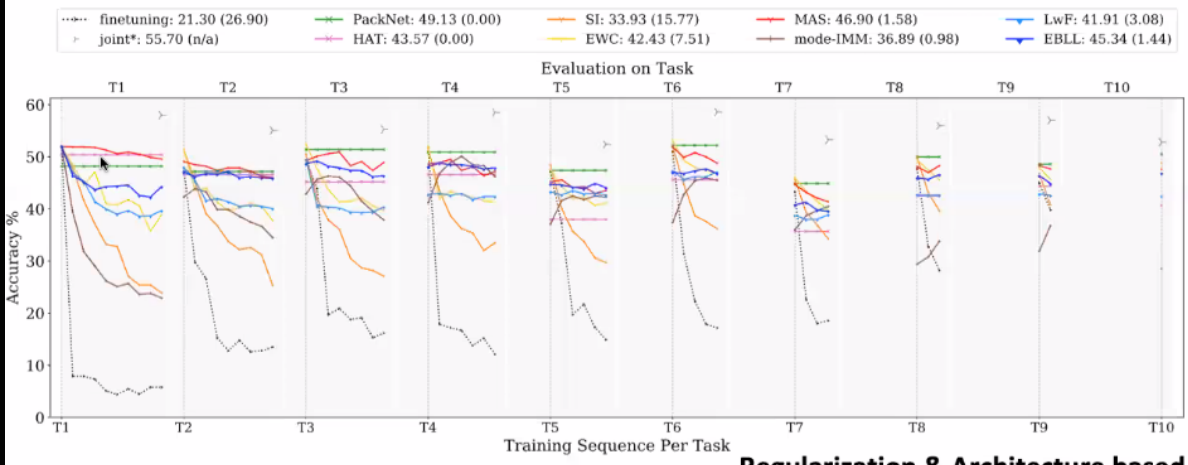
	Tiny Imagenet	iNaturalist
Tasks	10	10
Classes per task	20	5 to 314
Training data per task	8k	0.6k to 66k
Validation data per task	1k	0.1k to 9k
Task Constitution	random class selection	supercategory

- Fair way of setting hyperparameters (stability-plasticity tradeoff)



[Lange et al., 2020]

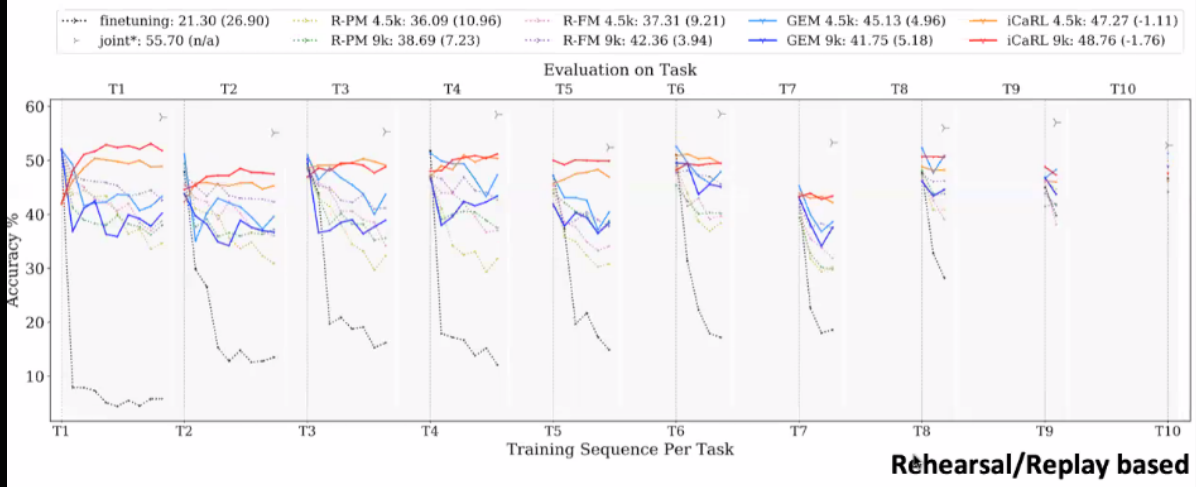
Comparative Evaluation (TinyImagenet)



Regularization & Architecture based

Image credit: [Lange et al., 2020]

Comparative Evaluation (TinyImagenet)



Rehearsal/Replay based

Image credit: [Lange et al., 2020]

General Trends

- Rehearsal/replay based methods only pay off when storing significant amount of exemplars
- PackNet results in no-forgetting and produces top results
- MAS more robust than EWC

Slide credit: T. Tuytelaars

Summary

- Flavour of different approaches:
  1. Regularization based: LwF, EBLL, EWC, SI, MAS, IMM, …
  2. Rehearsal / Replay: iCaRL, DGR, GEM, …
  3. Architecture based: PackNet, progressive nets , HAT, …
- More than classification?
- **Takeaways**

Future Scopes:

Looking to the future

- Desiderata
  - Constant memory
  - Task agnostic: Some recent advances [Rao et al., NeurIPS'19]
  - Forgetting gracefully
  - Datasets

"I don't like datasets, it's more a problem than a solution" – heard at ICCV 2019