Apprentissage continu de représentations visuelles

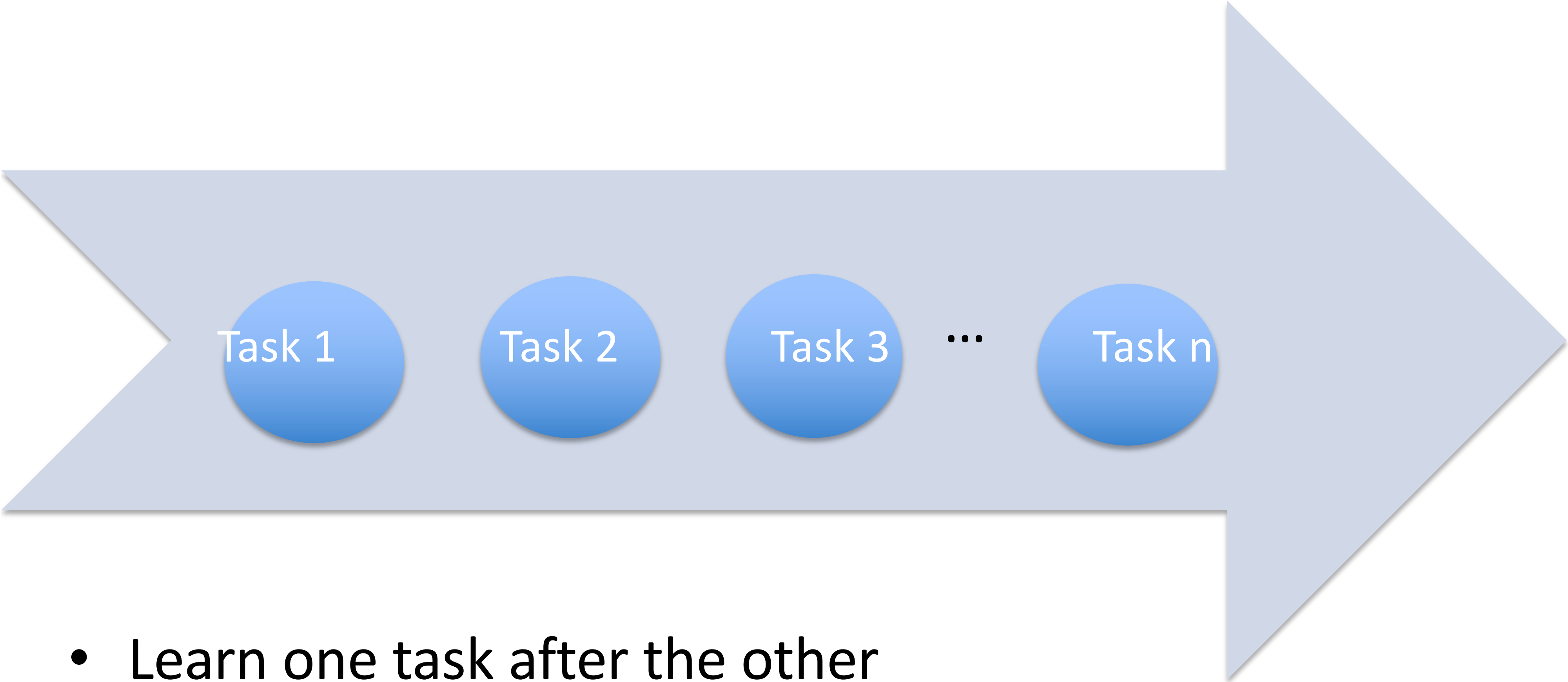
ENSIMAG
2023-2024

Karteek Alahari & Diane Larlus

Apprentissage continu

https://project.inria.fr/bigvisdata/

Incremental Learning: The Rules !



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

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

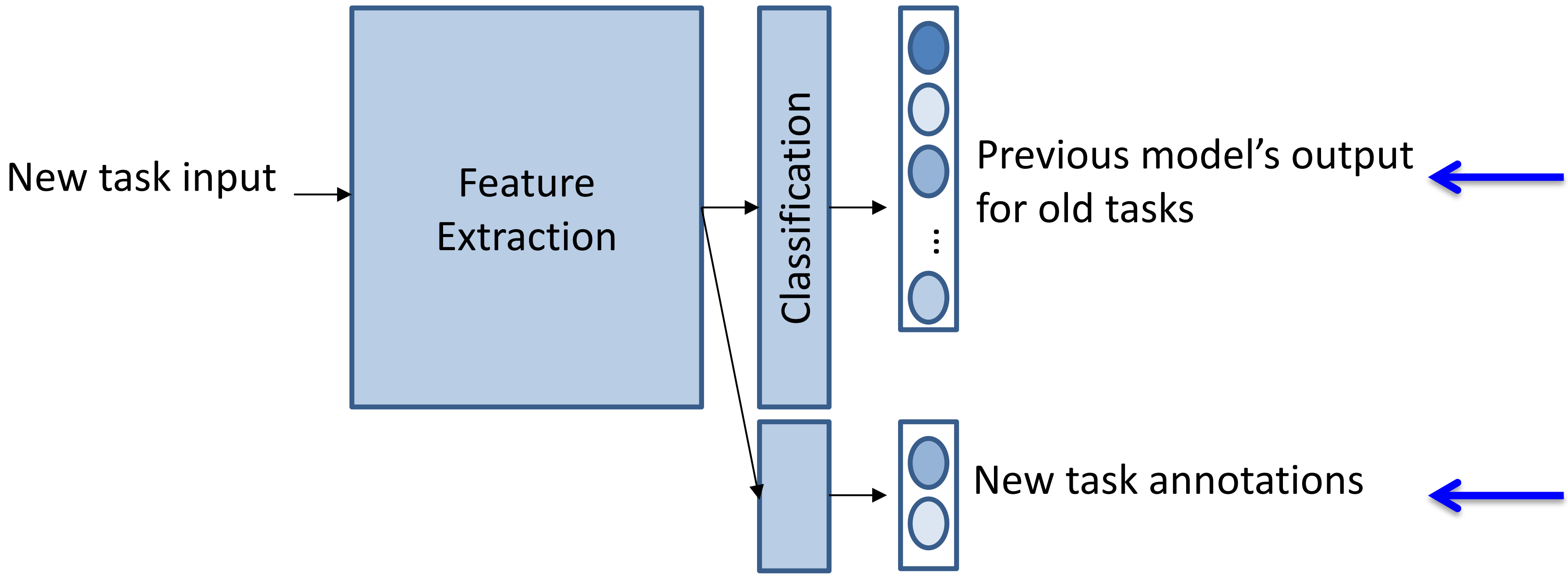
- When training a new task,
  - add a regularization term to the loss
  - i.e., term to penalize catastrophic forgetting

- 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]

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
  - Fisher information matrix for λ

$$\sum_{k} \sum_{i<n} \lambda_k^{n-i} (\theta_k^n - \theta_k^{n-i})^2$$

Model-focused Regularization

- Elastic weight consolidation [Kirkpatrick et al., 2017]
  - Indiv. penalty for each previous task
  - Fisher information matrix for λ

$$\sum_{k} \sum_{i<n} \lambda_k^{n-i} (\theta_k^n - \theta_k^{n-i})^2$$

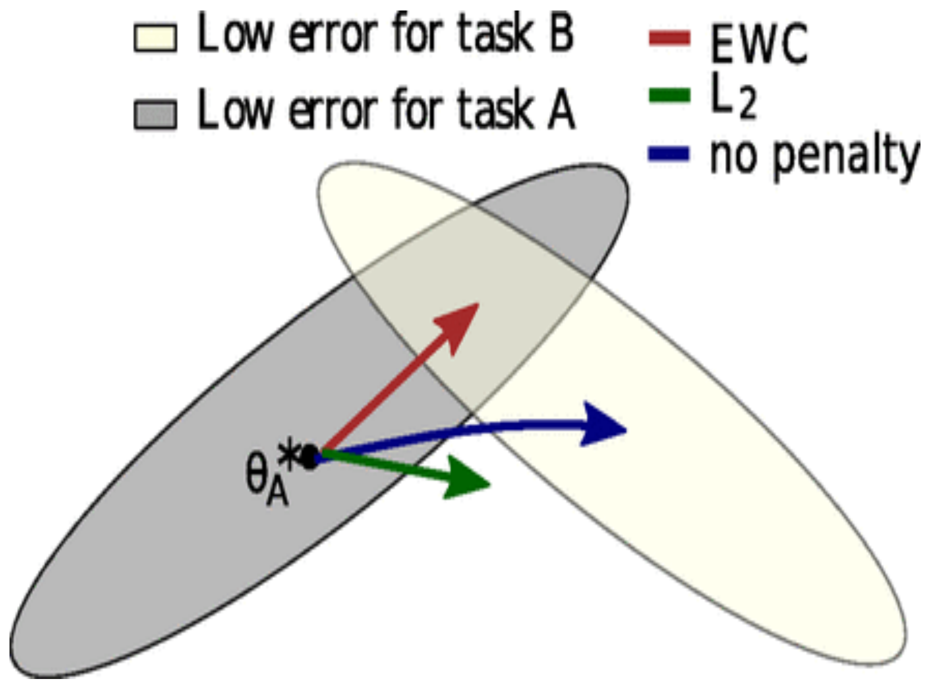


Figure from paper

Model-focused Regularization

- Elastic weight consolidation [Kirkpatrick et al., 2017]
  - Indiv. penalty for each previous task
  - Fisher information matrix for λ

$$\sum_{k} \sum_{i<n} \lambda_k^{n-i} (\theta_k^n - \theta_k^{n-i})^2$$

Agnostic to architecture; Good results empirically

Only valid locally

? Need to store importance weights

Model-focused Regularization

- Two examples
  - Elastic weight consolidation [Kirkpatrick et al., 2017]
  - Memory aware synapses [Aljundi et al., 2018]

- Other alternatives
  - Path Integral / Synaptic Intelligence: large changes during training [Zenke et al., 2017]
  - Moment matching [Lee et al., 2017]
  - Pathnet [Fernando et al., 2017]
  - …

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]

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

$\Theta \leftarrow \text{UpdateRepresentation}(X^s, \ldots, X^t; \mathcal{P}, \Theta)$

$m \leftarrow K/t$ // number of exemplars per class

for $y = 1, \ldots, s-1$ **do**
  $P_y \leftarrow \text{ReduceExemplarSet}(P_y, m)$
end for
for $y = s, \ldots, t$ **do**
  $P_y \leftarrow \text{ConstructExemplarSet}(X_y, m, \Theta)$
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 $\varphi : \mathcal{X} \to \mathbb{R}^d$ // 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

$$y^* \leftarrow \operatorname*{argmin}_{y=1,\ldots,t} \|\varphi(x) - \mu_y\|$$ // 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 [Rebuffi et al.'17]

Algorithm iCaRL CONSTRUCTEXEMPLARSET

---

input image set $X = \{x_1, \ldots, x_n\}$ of class y
input m target number of exemplars
require current feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
  $\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x)$ // current class mean
  for $k = 1, \ldots, m$ **do**
    $p_k \leftarrow \underset{x \in X}{\operatorname{argmin}} \left\| \mu - \frac{1}{k}[\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|$
  end for
  $P \leftarrow (p_1, \ldots, p_m)$
output exemplar set P

Algorithm iCaRL REDUCEEXEMPLARSET

---

input m // target number of exemplars
input $P = (p_1, \ldots, p_{|P|})$ // current exemplar set
  $P \leftarrow (p_1, \ldots, p_m)$ // *i.e.* keep only first m
output exemplar set P

[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]

What else will we see today?

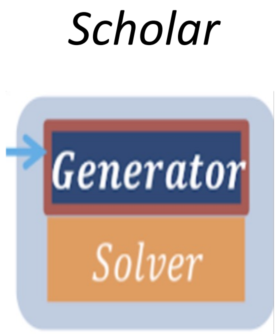
- Flavour of different approaches:
  1. Regularization based: LwF, EBLL, EWC, SI, MAS, IMM, …
  2. **Rehearsal / Replay**: iCaRL, DGR, GEM, …
  3. Architecture based: PackNet, progressive nets , HAT, …

- More than classification?

- Takeaways

Deep Generative Replay

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



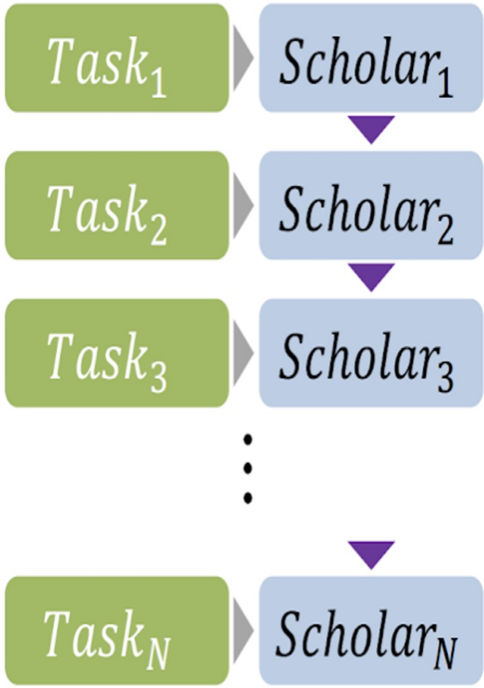
- The generator and the solver are updated in every incremental step

[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*



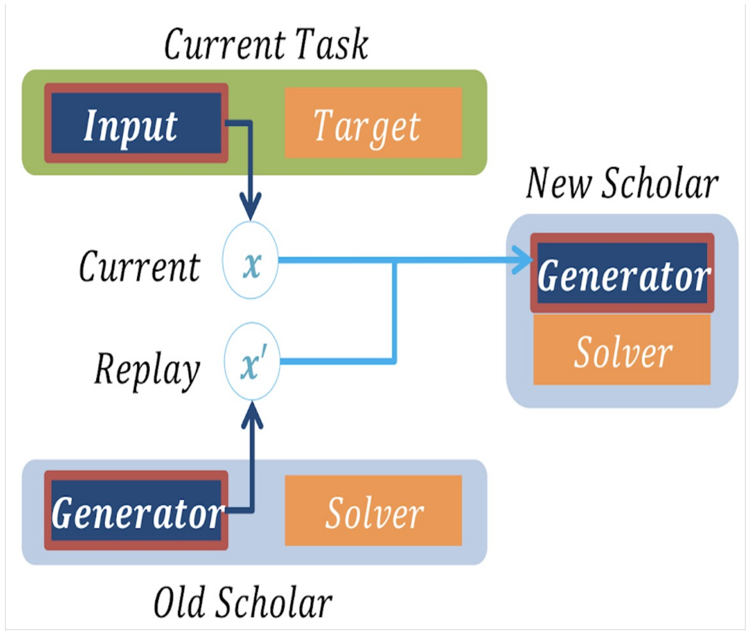
[Shin et al. 2017]
Figure from the paper

Slide courtesy: A. Massenet

Deep Generative Replay

Training procedure (Generator):

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



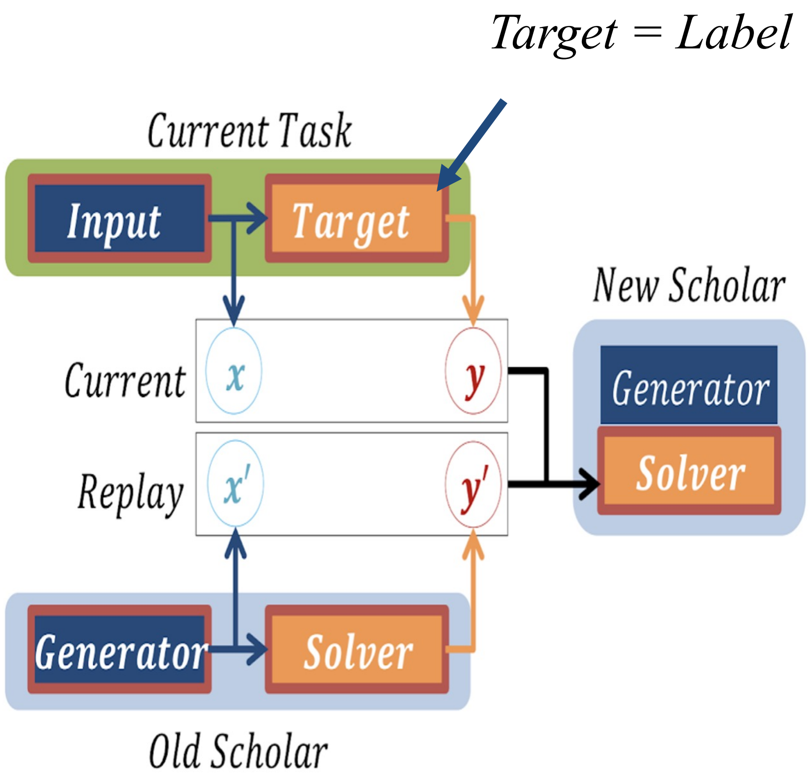
[Shin et al. 2017]
Figure from the paper

Slide courtesy: A. Massenet

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

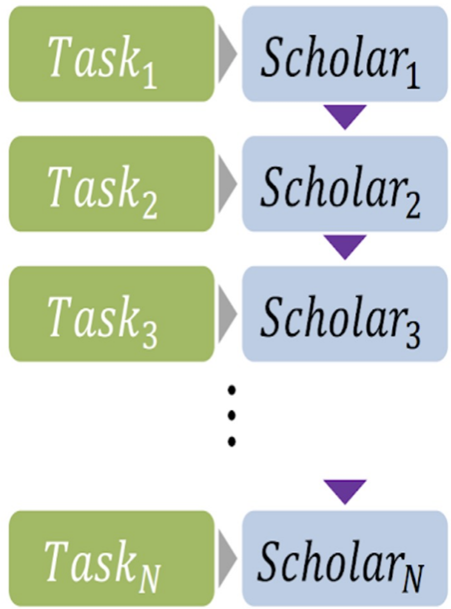
Slide courtesy: A. Massenet

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

Slide courtesy: A. Massenet

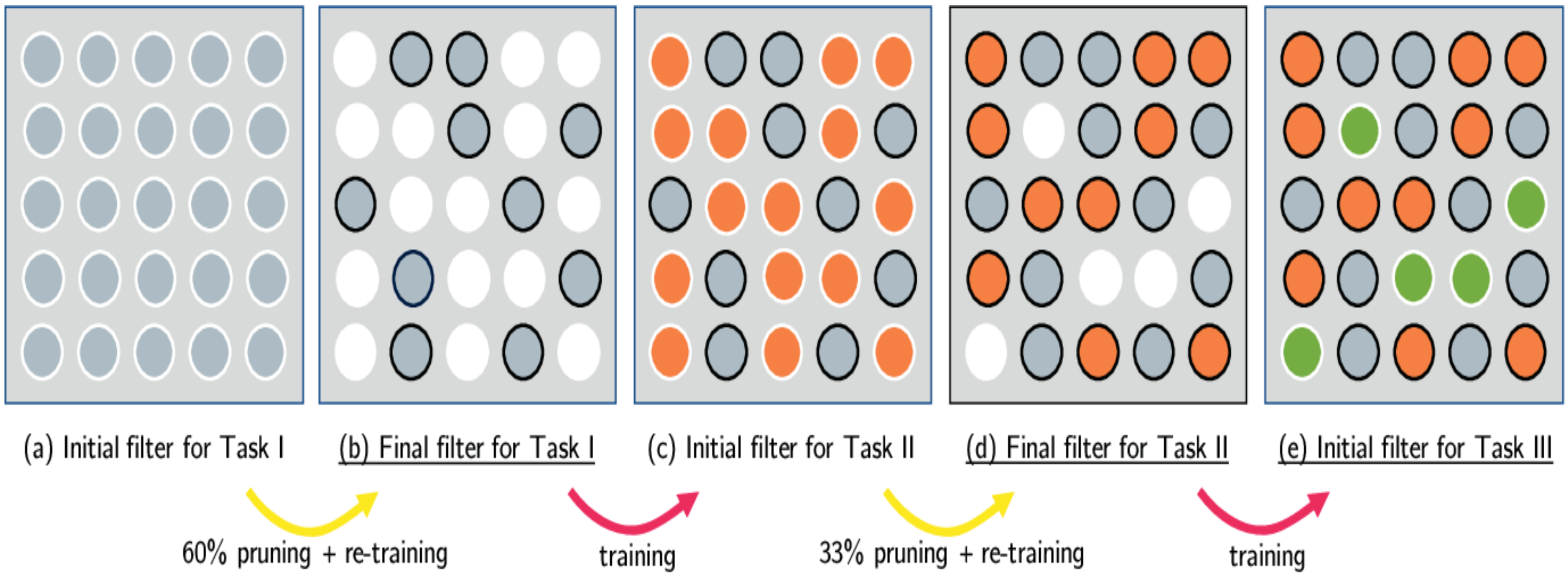
What else will we see today?

- 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

Architecture-based

Fixed memory consumption

Needs the total number of tasks

Avoids forgetting

PackNet [Mallya & Lazebnik’17]

A Comparative Analysis

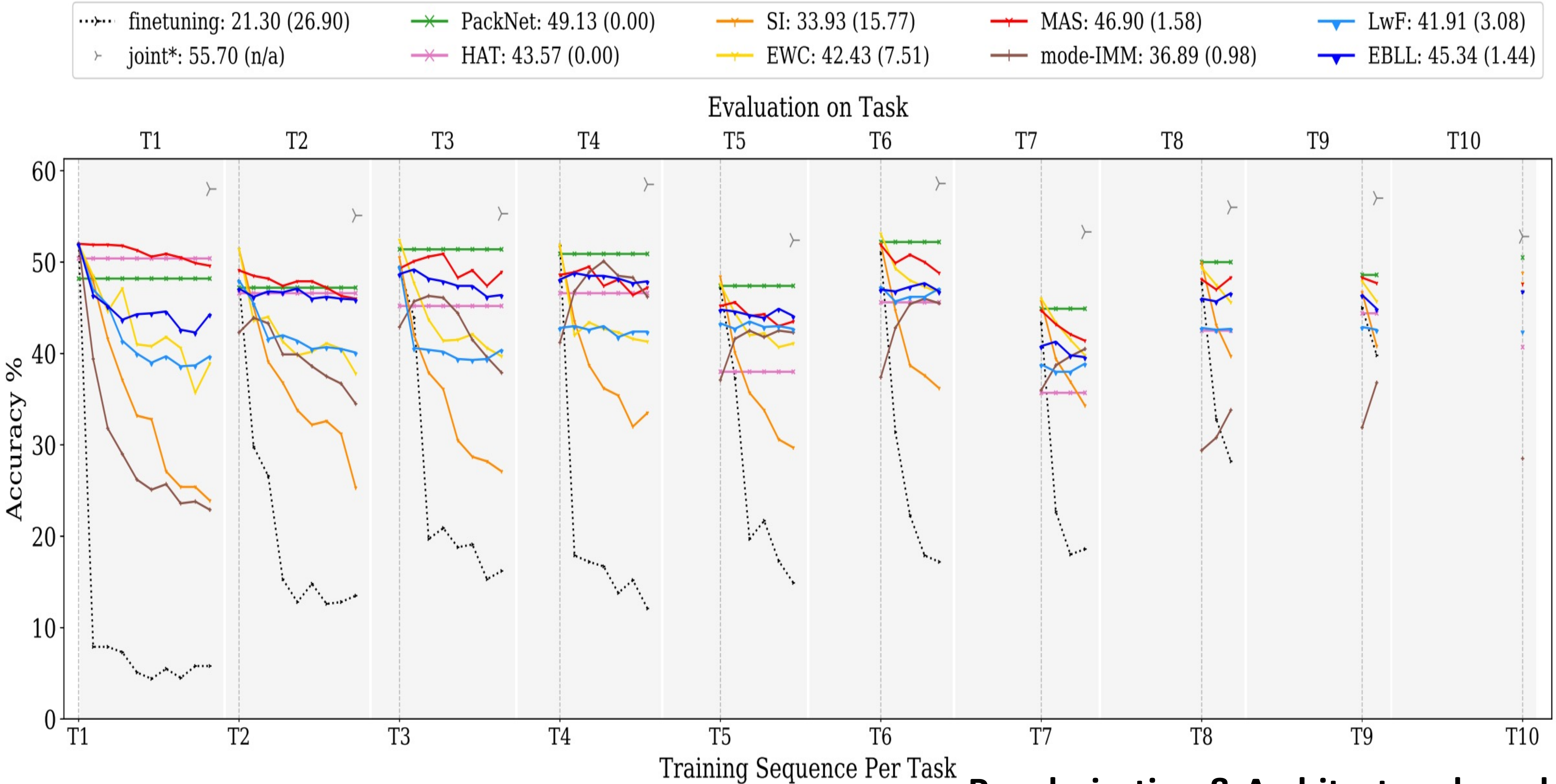
- TinyImagenet: small, balanced, class-incremental
- iNaturalist: large-scale, unbalanced, task-incremental

	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

- rs

[Lange et al., 2020]

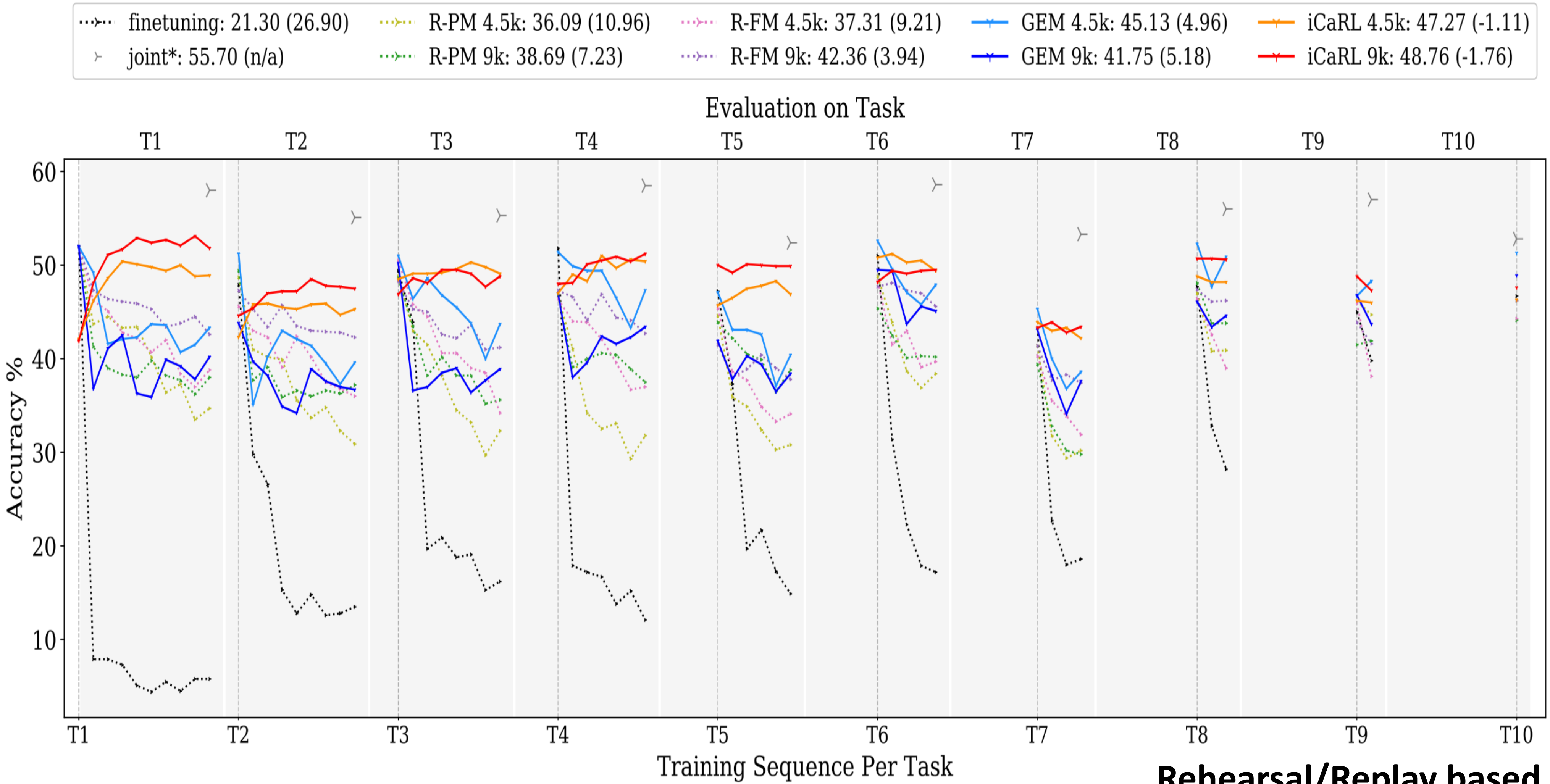
Comparative Evaluation (TinyImagenet)



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

What kind of model should I use ?

- Larger models give more capacity (but: overfitting)
- Wide is better than deep

- Regularization may interfere with incremental learning
- Dropout usually better than weight decay

Slide credit: T. Tuytelaars