CVPR 2020 Tutorial
Towards Annotation-Efficient Learning

Incremental Learning

Karteek Alahari
Inria, France
Incremental Learning?

- Continual learning
- Lifelong learning
- Sequential learning
- Never-ending Learning
An Incremental Learning Scenario

• Growing up in India
An Incremental Learning Scenario

- And then during travels
An Incremental Learning Scenario

- And then during travels

But can still remember *holy* cows!
KA: Incremental Learning

Learning Task 1

I can solve task 1.

Learning Task 2

I can solve tasks 1&2.

Learning Task 3

I can solve tasks 1&2&3.

...
Standard Machine Learning

TRAIN – VALIDATION – TEST

All sampled from the same distribution
- benchmarks and academic datasets 😊
- real-world systems ⚡
- embodied learning 🌀

KA: Incremental Learning

Slide credit: T. Tuytelaars
KA: Incremental Learning

Slide credit: T. Tuytelaars
Incremental Learning Setup

- Task-incremental learning
- Class-incremental learning
- Domain-incremental learning

KA: Incremental Learning
Slide credit: T. Tuytelaars
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

Input:
image for each task

Target:
- old tasks’ ground truth
- new task ground truth

KA: Incremental Learning

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

KA: Incremental Learning
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

Original model

Feature extraction

 KA: Incremental Learning

Figures from [Li and Hoiem 2016]
Naïve Solution 1

- Finetune only last layer using new data only
  - Leads to suboptimal results
Naïve Solution 2

Original model

Fine tuning

KA: Incremental Learning

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
How well does network B perform?

<table>
<thead>
<tr>
<th>method</th>
<th>Training with the initial set of classes</th>
<th>old</th>
<th>new</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(1-10)</td>
<td></td>
<td>65.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+B(11-20)</td>
<td></td>
<td>12.8</td>
<td>64.5</td>
<td>38.7</td>
</tr>
<tr>
<td>A(1-20)</td>
<td>Training with the new set of classes</td>
<td>68.4</td>
<td>71.3</td>
<td>69.8</td>
</tr>
</tbody>
</table>

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

No guidance for retaining the old classes

[Catastrophic forgetting: McCloskey and Cohen 1989, Ratcliff 1990]
Incremental Learning: The Rules!

- 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
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

KA: Incremental Learning
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
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

\[
\mathcal{L}(\theta) = \mathcal{L}_B(\theta^n) + \alpha \sum_{k} \lambda_k (\theta_k^n - \theta_k^{n-1})^2
\]

- Loss on new task(s)
- 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 \( \lambda \)

Figure from paper
Model-focused Regularization

- Elastic weight consolidation [Kirkpatrick et al., 2017]
  - Indiv. penalty for each previous task
    \[ \sum_k \sum_{i < n} \lambda^n_{k-i} (\theta^n_k - \theta^n_{k-i})^2 \]
  - Fisher information matrix for \( \lambda \)

**Thumbs up:** Agnostic to architecture; Good results empirically

**Thumbs down:** Only valid locally

**Question mark:** 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 \( \lambda \)

Figure from paper

KA: Incremental Learning
Model-focused Regularization

• Memory aware synapses [Aljundi et al., 2018]
  – Considers only the previous task \( \sum_k \lambda_k (\theta_k^m - \theta_{k-1}^m)^2 \)
  – Change in gradients for \( \lambda \)

  ![Green Thumbs Up]
  Agnostic to architecture; Leverages data & output

  ![Red Thumbs Down]
  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

KA: Incremental Learning
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

[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
$y^* \leftarrow \operatorname{arg\min}_{y=1,\ldots,t} \|\varphi(x) - \mu_y\|$ // nearest prototype
output class label $y^*$
```

[Rebuffi et al. 2017]
iCaRL: Incremental classifier and representation learning [Rebuffi et al.’17]

\[
\text{Algorithm } \text{iCaRL UPDATE REPRESENTATION} \\
\text{input } X^s, \ldots, X^t \quad \text{// training images of classes } s, \ldots, t \\
\text{require } \mathcal{P} = (P_1, \ldots, P_{s-1}) \quad \text{// exemplar sets} \\
\text{require } \Theta \quad \text{// current model parameters} \\
\text{// form combined training set:} \\
\mathcal{D} \leftarrow \bigcup_{y=s, \ldots, t} \{ (x, y) : x \in X^y \} \cup \bigcup_{y=1, \ldots, s-1} \{ (x, y) : x \in P_y \} \\
\text{// store network outputs with pre-update parameters:} \\
\text{for } y = 1, \ldots, s-1 \text{ do} \\
q^y_i \leftarrow g_y(x_i) \quad \text{for all } (x_i, \cdot) \in \mathcal{D} \\
\text{end for} \\
\text{run network training (e.g. BackProp) with loss function} \\
\ell(\Theta) = \sum_{(x_i, y_i) \in \mathcal{D}} \left[ \sum_{y=s}^t \delta_{y=y_i} \log g_y(x_i) + \delta_{y \neq y_i} \log(1-g_y(x_i)) \right. \\
\left. + \sum_{y=1}^{s-1} q^y_i \log g_y(x_i) + (1-q^y_i) \log(1-g_y(x_i)) \right] \\
\text{that consists of classification and distillation terms.}
\]

[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]

KA: Incremental Learning
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
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

KA: Incremental Learning
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
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
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

<table>
<thead>
<tr>
<th></th>
<th>Tiny Imagenet</th>
<th>iNaturalist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Classes per task</td>
<td>20</td>
<td>5 to 314</td>
</tr>
<tr>
<td>Training data per task</td>
<td>8k</td>
<td>0.6k to 66k</td>
</tr>
<tr>
<td>Validation data per task</td>
<td>1k</td>
<td>0.1k to 9k</td>
</tr>
<tr>
<td>Task Constitution</td>
<td>random class selection</td>
<td>supercategory</td>
</tr>
</tbody>
</table>

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

[Lange et al., 2020]
Comparative Evaluation (TinyImagenet)

Evaluation on Task

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>finetuning</td>
<td>21.30 (26.90)</td>
<td></td>
</tr>
<tr>
<td>joint*</td>
<td>55.70 (n/a)</td>
<td></td>
</tr>
<tr>
<td>PackNet</td>
<td>49.13 (0.00)</td>
<td></td>
</tr>
<tr>
<td>SI</td>
<td>33.93 (15.77)</td>
<td></td>
</tr>
<tr>
<td>EWC</td>
<td>42.43 (7.51)</td>
<td></td>
</tr>
<tr>
<td>MAS</td>
<td>46.90 (1.58)</td>
<td></td>
</tr>
<tr>
<td>LwF</td>
<td>41.91 (3.08)</td>
<td></td>
</tr>
<tr>
<td>mode-IMM</td>
<td>36.89 (0.98)</td>
<td></td>
</tr>
<tr>
<td>EBLI</td>
<td>45.34 (1.44)</td>
<td></td>
</tr>
</tbody>
</table>

Image credit: [Lange et al., 2020]
Comparative Evaluation (TinyImagenet)

Image credit: [Lange et al., 2020]

KA: Incremental Learning

Rehearsal/Replay based
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.
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
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
Mitigate Catastrophic Forgetting

• Learning without forgetting [Li and Hoiem 2016]
  - Tasks defined on a new dataset
  - Focus on image classification (rare co-occurrence of old and new)

• iCaRL [Rebuffi et al. 2017]
  - Decouple classifier and feature learning
  - Rely on a subset of the old data
Mitigate Catastrophic Forgetting

• Elastic weight consolidation [Kirkpatrick et al., 2017]
  – Selectively slowing down learning on weights
    Limited to specific settings
    Focus on image classification
    (rare co-occurrence of old and new)

• Other attempts, e.g., [Aljundi et al., 2018, Jung et al., 2016, Mallya and Lazebnik, 2017, Risin et al., 2014, Rusu et al., 2016]
Mitigate Catastrophic Forgetting

• Elastic weight consolidation [Kirkpatrick et al., 2017]
  – Selectively slowing down learning on weights
    Limited to specific settings

Lack of methods for
incremental learning of object detectors

• Other attempts, e.g., [Jung et al., 2016, Mallya and Lazebnik, 2017, Risin et al., 2014, Rusu et al., 2016]
An approach

• Incremental Learning of Object Detectors without Catastrophic Forgetting [Shmelkov et al., 2017]
Our Approach

• Builds on knowledge distillation [Hinton et al. ’14]

• Knowledge distillation originally for:
  – learning a simpler net equivalent to a complex one, and
  – produces a differently structured network

• We distill the “old” net when learning new classes

[Shmelkov, Schmid, Alahari, ICCV 2017]
Our Approach

• Copy of network (A) to evaluate proposals & memorize outputs

• New component (B) to learn the new classes

• Learn with a combination of losses

KA: Incremental Learning

[Fast RCNN: Girshick 2015]
Our Approach

• Minimize

\[ \mathcal{L} = \mathcal{L}_{rcnn} + \lambda \mathcal{L}_{dist} \]

\[ \mathcal{L}_{rcnn}(p, k^*, t, t^*) = -\log p_{k^*} + [k^* \geq 1] R(t - t^*) \]

Loss over classes

Localization loss

\[ \mathcal{L}_{dist}(y_A, t_A, y_B, t_B) = \frac{1}{N|C_A|} \sum \left[ (\bar{y}_A - \bar{y}_B)^2 + (t_A - t_B)^2 \right] \]

Comparing the two net components

KA: Incremental Learning
Our Approach

• Minimize

\[ \mathcal{L} = \mathcal{L}_{\text{rcnn}} + \lambda \mathcal{L}_{\text{dist}} \]

\[ \mathcal{L}_{\text{rcnn}}(p, k^*, t, t^*) = -\log p_{k^*} + [k^* \geq 1]R(t - t^*) \]

- Loss over classes
- Localization loss

\[ \mathcal{L}_{\text{dist}}(y_A, t_A, y_B, t_B) = \frac{1}{N|C_A|} \sum \left[ (\bar{y}_A - \bar{y}_B)^2 + (t_A - t_B)^2 \right] \]

- Logit responses
- Localization
Experimental setup

• Datasets: PASCAL VOC 2007 and MS COCO

• A(n-m): initial training on classes from n to m (old)
• B(n-m): incremental training on classes n to m (new) added all at once
• B(n)(m): incremental training for class n, then m, added one after another

• Unbiased distillation: distillation proposals are selected completely randomly
Addition of 10 classes

<table>
<thead>
<tr>
<th>method</th>
<th>old</th>
<th>new</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(1-10)</td>
<td>65.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+B(11-20), no distillation</td>
<td>12.8</td>
<td>64.5</td>
<td>38.7</td>
</tr>
<tr>
<td>+B(11-20) with distillation</td>
<td>63.2</td>
<td>63.1</td>
<td>63.1</td>
</tr>
<tr>
<td>A(1-20) (baseline)</td>
<td>68.4</td>
<td>71.3</td>
<td>69.8</td>
</tr>
</tbody>
</table>
## Addition of 10 classes

<table>
<thead>
<tr>
<th>method</th>
<th>old</th>
<th>new</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(1-10)</td>
<td>65.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+B(11-20), no distillation</td>
<td>12.8</td>
<td>64.5</td>
<td>38.7</td>
</tr>
<tr>
<td>+B(11-20) with distillation</td>
<td>63.2</td>
<td>63.1</td>
<td>63.1</td>
</tr>
<tr>
<td>+B(11-20) with EWC</td>
<td>31.6</td>
<td>61.0</td>
<td>46.3</td>
</tr>
<tr>
<td>A(1-20) (baseline)</td>
<td>68.4</td>
<td>71.3</td>
<td>69.8</td>
</tr>
</tbody>
</table>
Addition of 5 classes

- VOC 2007 validation set

<table>
<thead>
<tr>
<th>method</th>
<th>old</th>
<th>new</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong>(1-15)</td>
<td>70.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>+B</strong>(16-20) with distillation</td>
<td><strong>68.4</strong></td>
<td>58.4</td>
<td>65.9</td>
</tr>
<tr>
<td><strong>+B</strong>(16)(17)...(20) with distillation</td>
<td><strong>66.0</strong></td>
<td>51.6</td>
<td>62.4</td>
</tr>
<tr>
<td><strong>A</strong>(1-20) (baseline)</td>
<td>70.9</td>
<td>66.7</td>
<td>69.8</td>
</tr>
</tbody>
</table>

KA: Incremental Learning
Addition of 5 classes

- VOC 2007 validation set

<table>
<thead>
<tr>
<th>method</th>
<th>old</th>
<th>new</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(1-15)</td>
<td>70.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+B(16-20) with distillation</td>
<td>68.4</td>
<td>58.4</td>
<td>65.9</td>
</tr>
<tr>
<td>+B(16)(17)...(20) with distillation</td>
<td>66.0</td>
<td>51.6</td>
<td>62.4</td>
</tr>
<tr>
<td>+B(16)(17)...(20) w unbiased distillation</td>
<td>45.8</td>
<td>46.5</td>
<td>46.0</td>
</tr>
<tr>
<td>A(1-20) (baseline)</td>
<td>70.9</td>
<td>66.7</td>
<td>69.8</td>
</tr>
</tbody>
</table>

KA: Incremental Learning
Addition of 40 classes

- COCO minival (first 5000 validation images)

<table>
<thead>
<tr>
<th>method</th>
<th>mAP@.5</th>
<th>mAP@[.5, .95]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(1-40)+B(41-80)</td>
<td>37.4</td>
<td>21.3</td>
</tr>
<tr>
<td>A(1-80) (baseline)</td>
<td>38.1</td>
<td>22.6</td>
</tr>
</tbody>
</table>

KA: Incremental Learning
Intermediate Summary

- No catastrophic forgetting with
  - distillation loss
  - in particular, biased distillation

- At ECCV 2018: Balanced fine-tuning
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
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
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

See also: Workshop on 14th June
Continual Learning in Computer Vision