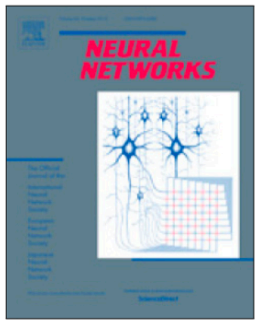
Neural Networks 116 (2019) 56–73

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Continuous learning in single-incremental-task scenarios

Davide Maltoni *, Vincenzo Lomonaco

Department of Computer Science and Engineering, University of Bologna, Italy

Renato Santos Aranha
Dez/2020

Up to now: architectural, regularization and rehearsal strategies to train deep models sequentially on a number of disjoint tasks

Up to now: architectural, regularization and rehearsal strategies to train deep models sequentially on a number of disjoint tasks

tasks are not disjoint (constitute a single incremental task, e.g. class-incremental learning.) → these strategies are unsatisfactory

Up to now: architectural, regularization and rehearsal strategies to train deep models sequentially on a number of disjoint tasks

tasks are not disjoint (constitute a single incremental task, e.g. class-incremental learning.) → these strategies are unsatisfactory

main contributions of this paper:

- point some differences between MT and SIT scenarios under NC update type
- review of popular CL approaches (LWF, EWC, SI)
- propose a new CL strategy (AR1) by extending CWR architectural strategy (introduced in Lomonaco and Maltoni (2017a) and combining it with a light regularization approach such as SI.
- compare AR1 with other common CL strategies on CORe50 and iCIFAR-100 benchmarks
- discuss diagnostic techniques to simplify hyperparameters tuning in complex CL settings and detect misbehaviors.
- additional material section: implementation details of all the techniques reviewed/proposed and the experiment carried out using the Caffe framework

Up to now: architectural, regularization and rehearsal strategies to train deep models sequentially on a number of disjoint tasks

tasks are not disjoint (constitute a single incremental task, e.g. class-incremental learning.) → these strategies are unsatisfactory

main contributions of this paper:

- **point some differences between MT and SIT scenarios under NC update type**
- **review of popular CL approaches (LWF, EWC, SI)**
- **propose a new CL strategy (AR1) by extending CWR architectural strategy (introduced in Lomonaco and Maltoni (2017a) and combining it with a light regularization approach such as SI.**
- **compare AR1 with other common CL strategies on CORe50 and iCIFAR-100 benchmarks**
- **discuss diagnostic techniques to simplify hyperparameters tuning in complex CL settings and detect misbehaviors.**
- **additional material section: implementation details of all the techniques reviewed/proposed and the experiment carried out using the Caffe framework**

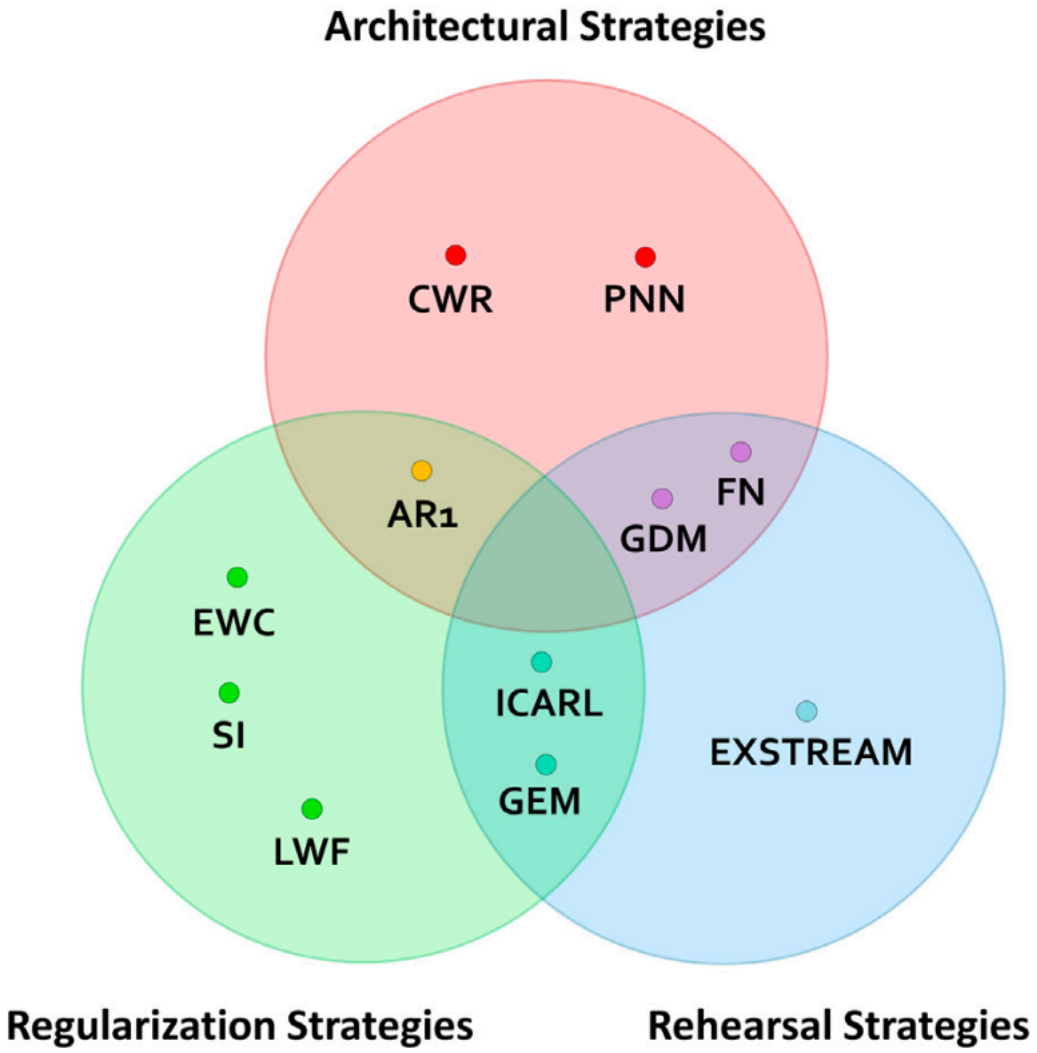
Up to now: architectural, regularization and rehearsal strategies to train deep models sequentially on a number of disjoint tasks

tasks are not disjoint (constitute a single incremental task, e.g. class-incremental learning.) → these strategies are unsatisfactory

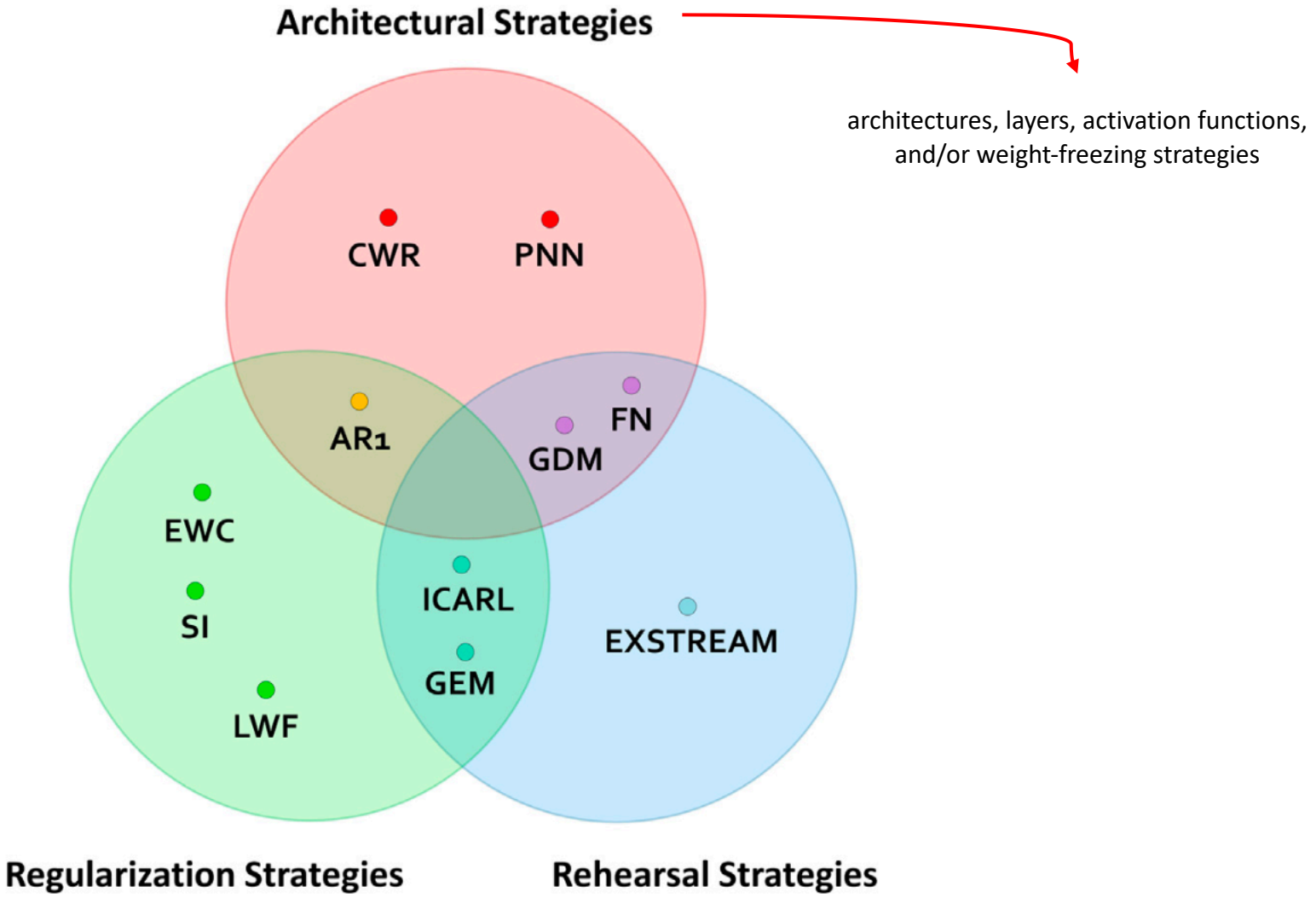
main contributions of this paper:

- **point some differences between MT and SIT scenarios under NC update type**
- **review of popular CL approaches (LWF, EWC, SI)**
- **propose a new CL strategy (AR1) by extending CWR architectural strategy (introduced in Lomonaco and Maltoni (2017a) and combining it with a light regularization approach such as SI.**
- **compare AR1 with other common CL strategies on CORe50 and iCIFAR-100 benchmarks**
- **discuss diagnostic techniques to simplify hyperparameters tuning in complex CL settings and detect misbehaviors.**
- **additional material section: implementation details of all the techniques reviewed/proposed and the experiment carried out using the Caffe framework**

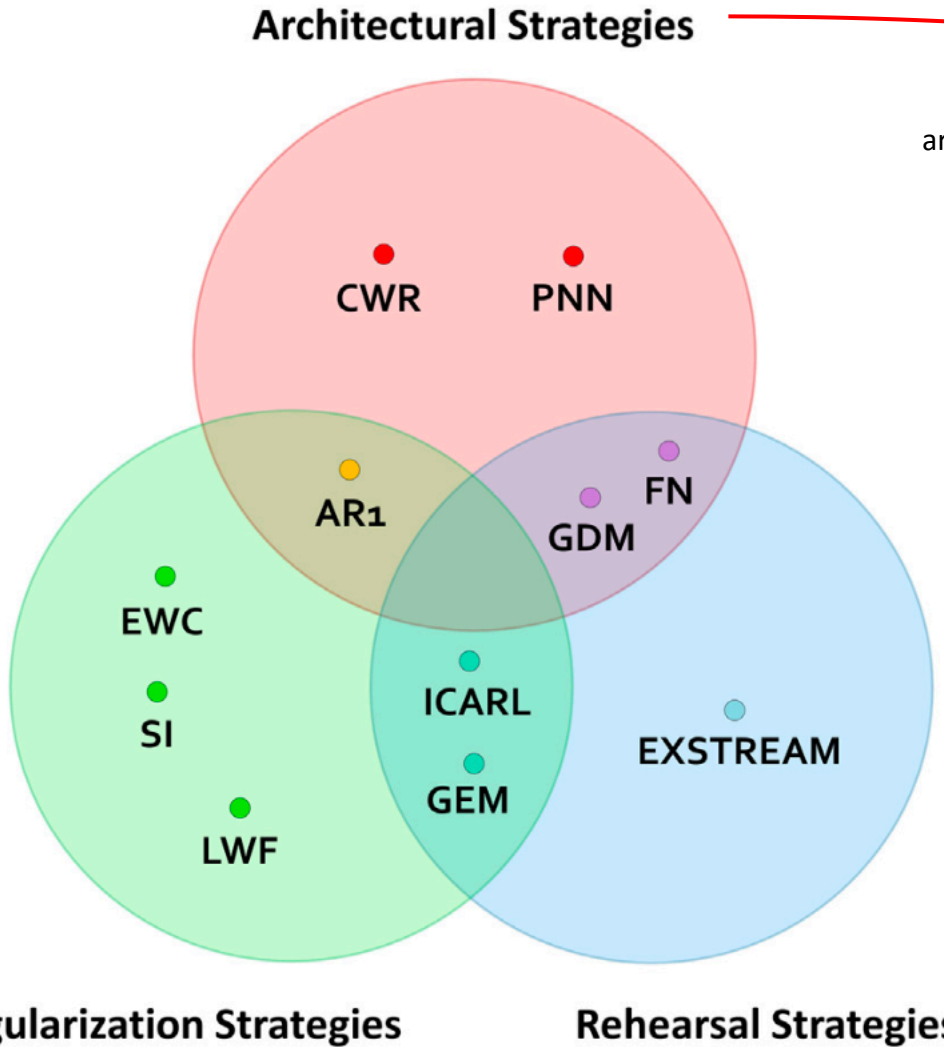
categorization of the most common CL strategies



categorization of the most common CL strategies



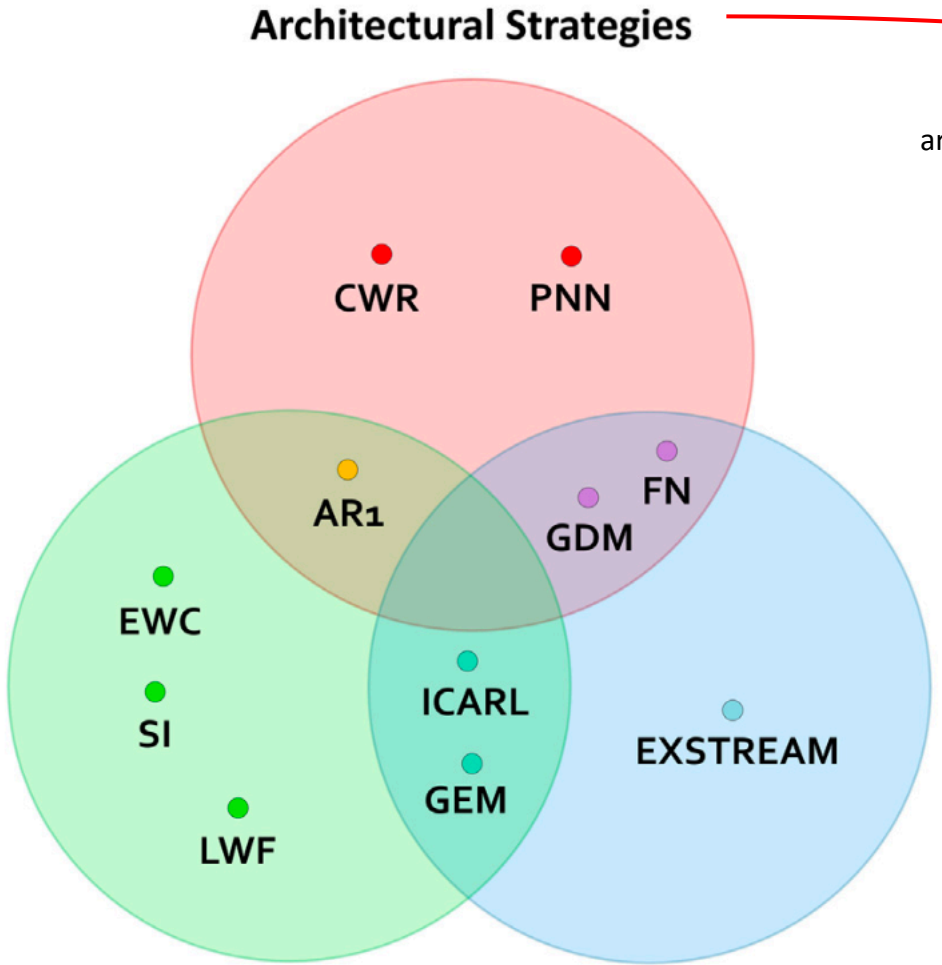
categorization of the most common CL strategies



architectures, layers, activation functions, and/or weight-freezing strategies

loss function is extended with terms promoting selective consolidation of weights important to retain past “memories”

categorization of the most common CL strategies

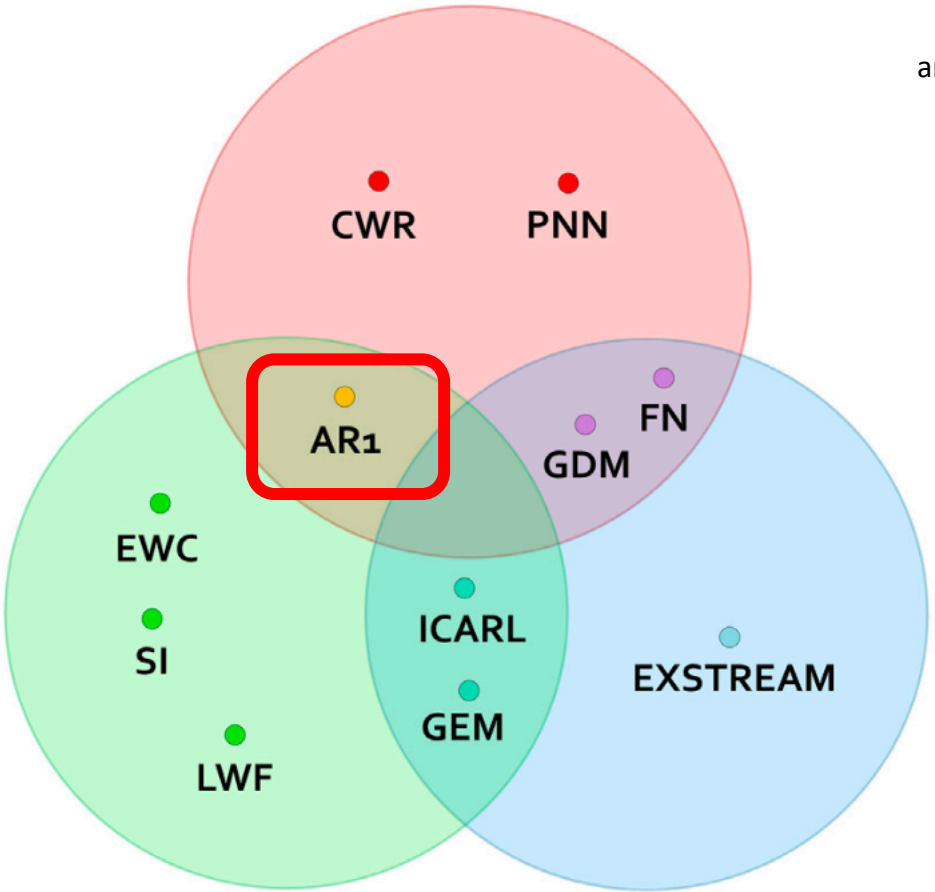


architectures, layers, activation functions, and/or weight-freezing strategies

loss function is extended with terms promoting selective consolidation of weights important to retain past “memories”

past information periodically replayed to the model to strengthen connections for memories it has already learned

categorization of the most common CL strategies



architectures, layers, activation functions, and/or weight-freezing strategies

loss function is extended with terms promoting selective consolidation of weights important to retain past “memories”

past information periodically replayed to the model to strengthen connections for memories it has already learned

Multi-Task (MT) scenario

Single-Incremental-Task (SIT) scenario

Multi-Task (MT) scenario

- most CL studies

Single-Incremental-Task (SIT) scenario

Multi-Task (MT) scenario

- most CL studies

- same model is required to learn incrementally a number of disjoint tasks (no class overlapping)

Single-Incremental-Task (SIT) scenario

Multi-Task (MT) scenario

- most CL studies

- same model is required to learn incrementally a number of disjoint tasks (no class overlapping)

- accuracy is computed separately for each task

Single-Incremental-Task (SIT) scenario

Multi-Task (MT) scenario

- most CL studies

- same model is required to learn incrementally a number of disjoint tasks (no class overlapping)

- accuracy is computed separately for each task

- not appropriate for addressing "class incremental" problems…

Single-Incremental-Task (SIT) scenario

Multi-Task (MT) scenario

- most CL studies

- same model is required to learn incrementally a number of disjoint tasks (no class overlapping)

- accuracy is computed separately for each task

- not appropriate for addressing "class incremental" problems…

Single-Incremental-Task (SIT) scenario

- still largely unexplored scenario

Multi-Task (MT) scenario

- most CL studies

- same model is required to learn incrementally a number of disjoint tasks (no class overlapping)

- accuracy is computed separately for each task

- not appropriate for addressing "class incremental" problems…

Single-Incremental-Task (SIT) scenario

- still largely unexplored scenario

- considers a single task which is incremental in nature (e.g. class-incremental learning)

Multi-Task (MT) scenario

- most CL studies

- same model is required to learn incrementally a number of disjoint tasks (no class overlapping)

- accuracy is computed separately for each task

- not appropriate for addressing "class incremental" problems…

Single-Incremental-Task (SIT) scenario

- still largely unexplored scenario

- considers a single task which is incremental in nature (e.g. class-incremental learning)

- when computing accuracy, we need to distinguish among classes encountered so far

Multi-Task (MT) scenario

- most CL studies

- same model is required to learn incrementally a number of disjoint tasks (no class overlapping)

- accuracy is computed separately for each task

- not appropriate for addressing "class incremental" problems…

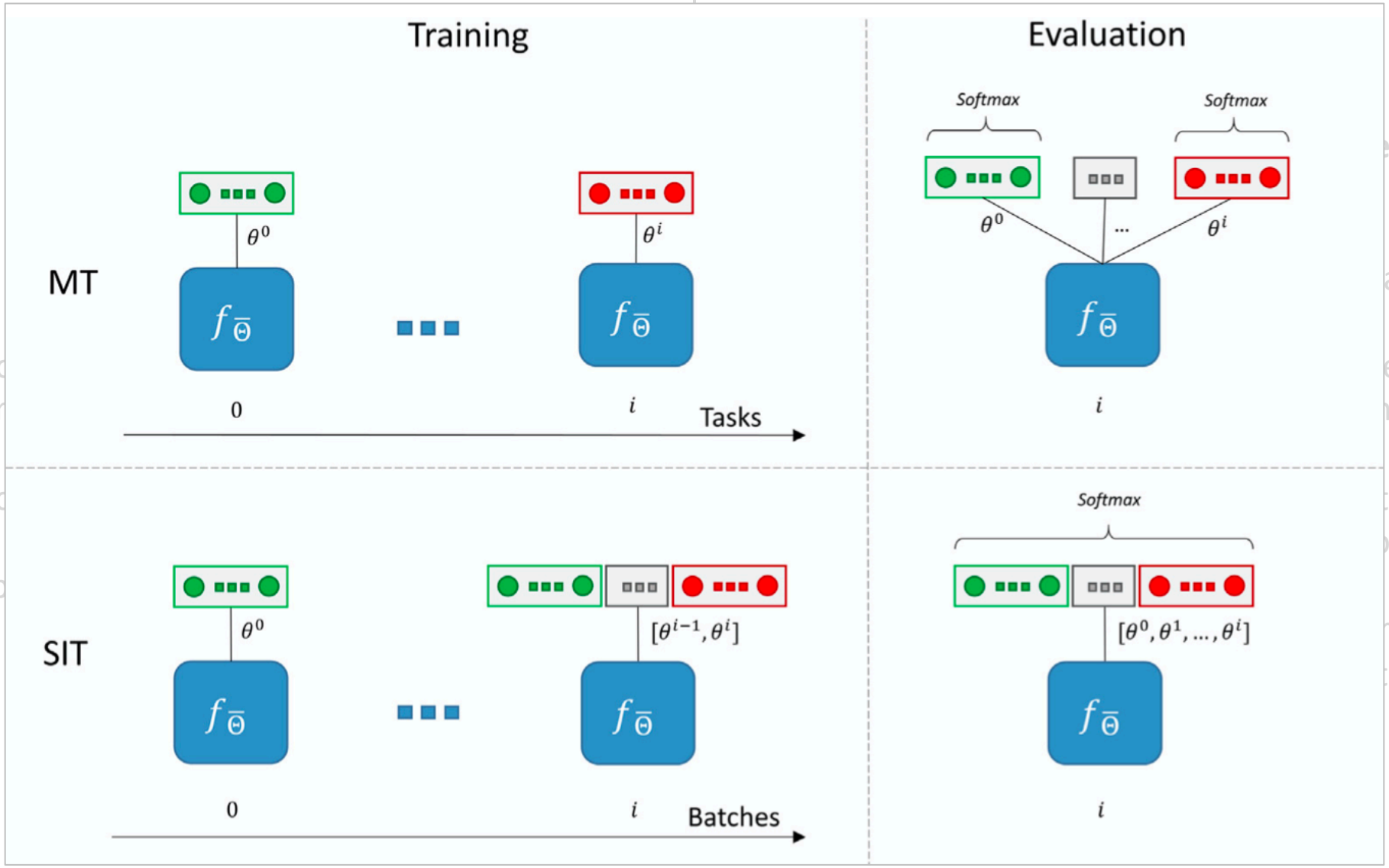
Single-Incremental-Task (SIT) scenario

- still largely unexplored scenario

- considers a single task which is incremental in nature (e.g. class-incremental learning)

- when computing accuracy, we need to distinguish among classes encountered so far

- quite common in natural learning (a child recognizing new objects…)


Training
Evaluation
MT
SIT
Softmax
Softmax
Softmax
θ^0
θ^i
$[\theta^{i-1}, \theta^i]$
$[\theta^0, \theta^1, \ldots, \theta^i]$
$f_{\bar{\Theta}}$
0
i
Tasks
Batches

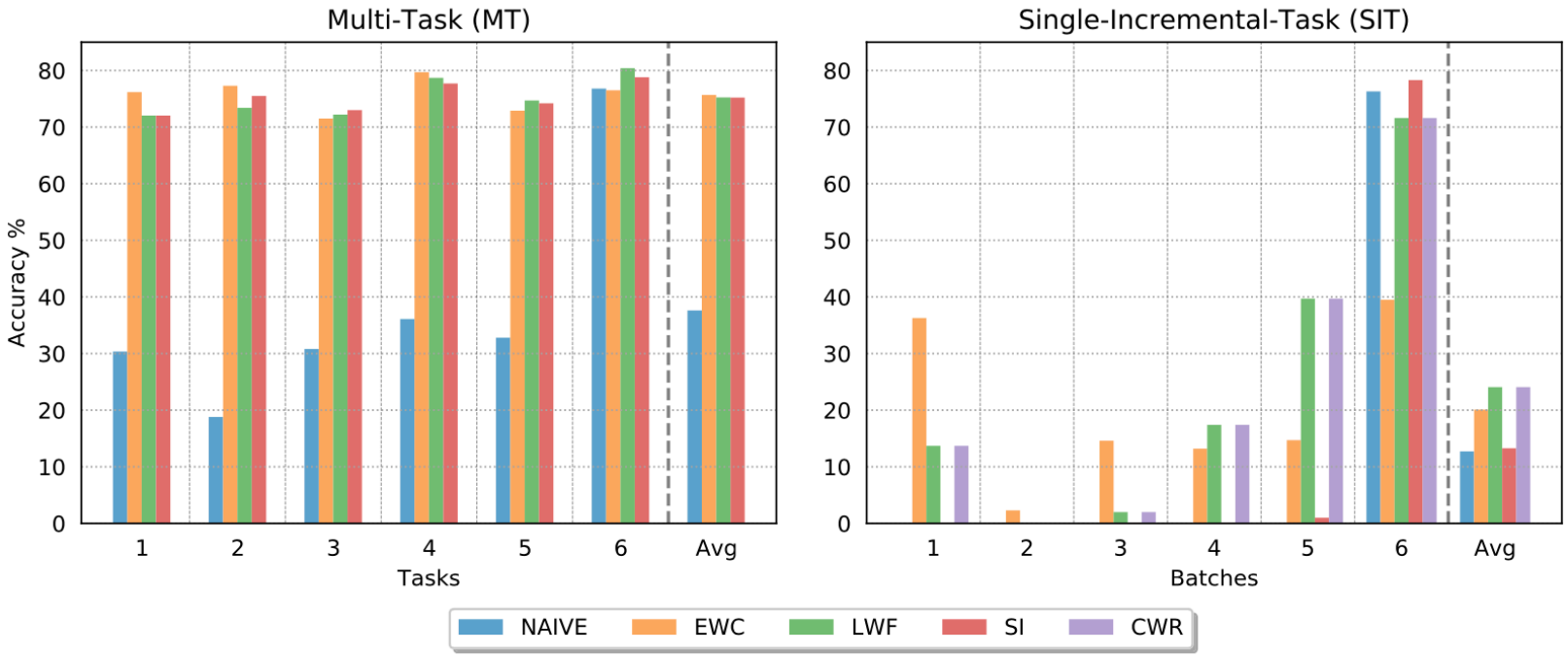


Fig. 3. Accuracy in MT and SIT scenarios for 5 CL strategies (NAÏVE, EWC, LWF, SI, CWR) after the last training batch. With NAÏVE we denote a simple incremental fine-tuning where early stopping is the only option available to limit forgetting. Analogously to Zenke et al. (2017) this experiment was performed on the first 6 tasks of CIFAR-10/100 split. For both MT and SIT we report the accuracies on the classes of each batch (1, 2, …, 5) and their average (Avg). CWR is specifically designed for SIT and was not tested under MT. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

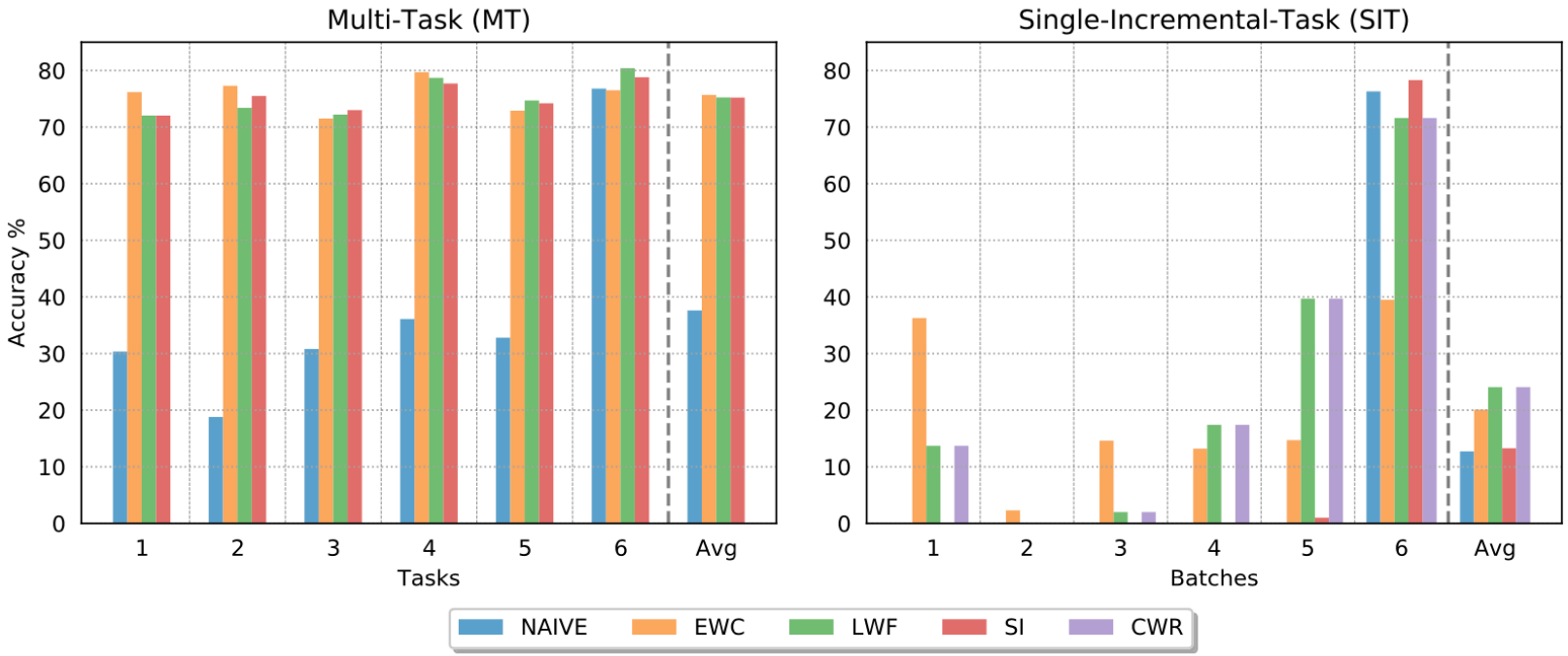


Fig. 3. Accuracy in MT and SIT scenarios for 5 CL strategies (NAÏVE, EWC, LWF, SI, CWR) after the last training batch. With NAÏVE we denote a simple incremental fine-tuning where early stopping is the only option available to limit forgetting. Analogously to Zenke et al. (2017) this experiment was performed on the first 6 tasks of CIFAR-10/100 split. For both MT and SIT we report the accuracies on the classes of each batch (1, 2, …, 5) and their average (Avg). CWR is specifically designed for SIT and was not tested under MT. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- unbalanced nature of CIFAR Split benchmark (50% of all the train and test set examples belong to the first batch) makes SIT strategies harder to parametrize;

- however, as argued by other researches (Kemker & Kanan, 2018; Kemker et al., 2018), most of the existing CL approaches perform well on MT but fail on SIT scenarios;

Up to now: architectural, regularization and rehearsal strategies to train deep models sequentially on a number of disjoint tasks

tasks are not disjoint (constitute a single incremental task, e.g. class-incremental learning.) → these strategies are unsatisfactory

main contributions of this paper:

- **point some differences between MT and SIT scenarios under NC update type**
- **review of popular CL approaches (LWF, EWC, SI)**
- **propose a new CL strategy (AR1) by extending CWR architectural strategy (introduced in Lomonaco and Maltoni (2017a) and combining it with a light regularization approach such as SI.**
- **compare AR1 with other common CL strategies on CORe50 and iCIFAR-100 benchmarks**
- **discuss diagnostic techniques to simplify hyperparameters tuning in complex CL settings and detect misbehaviors.**
- **additional material section: implementation details of all the techniques reviewed/proposed and the experiment carried out using the Caffe framework**

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

CWR → two sets of weights in output classification layer

cw: consolidated weights used for inference
tw: temporary weights used for training

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

CWR → two sets of weights in output classification layer

cw: consolidated weights used for inference

tw: temporary weights used for training

initialized to 0 before the first batch

randomly reinitialized (e.g., Gaussian initialization with std = 0.01, mean = 0) before each training batch

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

CWR → two sets of weights in output classification layer

cw: consolidated weights used for inference

tw: temporary weights used for training

initialized to 0 before the first batch

randomly reinitialized (e.g., Gaussian initialization with std = 0.01, mean = 0) before each training batch

at the end of each batch training → weights in tw corresponding to the classes in the current batch are scaled and copied in cw

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

CWR → two sets of weights in output classification layer

cw: consolidated weights used for inference

tw: temporary weights used for training

initialized to 0 before the first batch

randomly reinitialized (e.g., Gaussian initialization with std = 0.01, mean = 0) before each training batch

at the end of each batch training → weights in tw corresponding to the classes in the current batch are scaled and copied in cw

to avoid forgetting in the lower levels → after the first batch B1, all lower level weights Θ are frozen

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

CWR → two sets of weights in output classification layer

cw: consolidated weights used for inference
tw: temporary weights used for training

initialized to 0 before the first batch

randomly reinitialized (e.g., Gaussian initialization with std = 0.01, mean = 0) before each training batch

at the end of each batch training → weights in tw corresponding to the classes in the current batch are **scaled** and copied in cw

to avoid forgetting in the lower levels → after the first batch B1, all lower level weights Θ are frozen

Obs.: weight scaling (with batch specific weights wi) is necessary in case of unbalanced batches with respect to the number of classes or examples per class.

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

$cw = 0$
init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)
for each training batch B_i:
    expand output layer with s_i neuros for the new classes in B_i
    random re-init tw (for all neurons in the output layer)
    Train the model with SGD on the s_i classes of B_i:
        if $B_i = B_1$ learn both $\bar{\Theta}$ and tw
        else learn tw while keeping $\bar{\Theta}$ fixed
    for each class j among the s_i classes in B_i:
        $cw[j] = w_i \cdot tw[j]$
    Test the model by using $\bar{\Theta}$ and cw

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

$cw = 0$
init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)
for each training batch B_i:
    expand output layer with s_i neuros for the new classes in B_i
    random re-init tw (for all neurons in the output layer)
    Train the model with SGD on the s_i classes of B_i:
        if $B_i = B_1$ learn both $\bar{\Theta}$ and tw
        else learn tw while keeping $\bar{\Theta}$ fixed
    for each class j among the s_i classes in B_i:
        $cw[j] = w_i \cdot tw[j]$
    Test the model by using $\bar{\Theta}$ and cw

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

$cw = 0$
init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)
for each training batch B_i:
    expand output layer with s_i neuros for the new classes in B_i
    random re-init tw (for all neurons in the output layer)
    Train the model with SGD on the s_i classes of B_i:
        if $B_i = B_1$ learn both $\bar{\Theta}$ and tw
        else learn tw while keeping $\bar{\Theta}$ fixed
    for each class j among the s_i classes in B_i:
        $cw[j] = w_i \cdot tw[j]$
    Test the model by using $\bar{\Theta}$ and cw

CWR modification (CWR+): mean-shift and zero initialization

$cw = 0$
init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)
for each training batch B_i:
    expand output layer with s_i neuros for the new classes in B_i
    $\boldsymbol{tw = 0}$ (for all neurons in the output layer)
    Train the model with SGD on the s_i classes of B_i:
        if $B_i = B_1$ learn both $\bar{\Theta}$ and tw
        else learn tw while keeping $\bar{\Theta}$ fixed
    for each class j among the s_i classes in B_i:
        $\boldsymbol{cw[j] = tw[j] - avg(tw)}$
    Test the model by using $\bar{\Theta}$ and cw

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

$cw = 0$
init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)
for each training batch B_i:
    expand output layer with s_i neuros for the new classes in B_i
    random re-init tw (for all neurons in the output layer)
    Train the model with SGD on the s_i classes of B_i:
        if $B_i = B_1$ learn both $\bar{\Theta}$ and tw
        else learn tw while keeping $\bar{\Theta}$ fixed
    for each class j among the s_i classes in B_i:
        $cw[j] = w_i \cdot tw[j]$
    Test the model by using $\bar{\Theta}$ and cw

CWR modification (CWR+): mean-shift and zero initialization

$cw = 0$
init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)
for each training batch B_i:
    expand output layer with s_i neuros for the new classes in B_i
    $\boldsymbol{tw = 0}$ (for all neurons in the output layer)
    Train the model with SGD on the s_i classes of B_i:
        if $B_i = B_1$ learn both $\bar{\Theta}$ and tw
        else learn tw while keeping $\bar{\Theta}$ fixed
    for each class j among the s_i classes in B_i:
        $\boldsymbol{cw[j] = tw[j] - avg(tw)}$
    Test the model by using $\bar{\Theta}$ and cw

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

$cw = 0$
init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)
for each training batch B_i:
    expand output layer with s_i neuros for the new classes in B_i
    random re-init tw (for all neurons in the output layer)
    Train the model with SGD on the s_i classes of B_i:
        if $B_i = B_1$ learn both $\bar{\Theta}$ and tw
        else learn tw while keeping $\bar{\Theta}$ fixed
    for each class j among the s_i classes in B_i:
        $cw[j] = w_i \cdot tw[j]$
    Test the model by using $\bar{\Theta}$ and cw

CWR modification (CWR+): mean-shift and zero initialization

$cw = 0$
init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)
for each training batch B_i:
    expand output layer with s_i neuros for the new classes in B_i
    $\boldsymbol{tw = 0}$ (for all neurons in the output layer)
    Train the model with SGD on the s_i classes of B_i:
        if $B_i = B_1$ learn both $\bar{\Theta}$ and tw
        else learn tw while keeping $\bar{\Theta}$ fixed
    for each class j among the s_i classes in B_i:
        $\boldsymbol{cw[j] = tw[j] - avg(tw)}$
    Test the model by using $\bar{\Theta}$ and cw

weights Θ are tuned during the first batch and then frozen…
one more CWR and CWR+ drawback!

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

Synaptic intelligence

Zenke et al. 2017

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

Synaptic intelligence

Zenke et al. 2017

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

ELASTIC WEIGHT CONSOLIDATION

Kirkpatrick et al. [2017]

The idea:
inspired by human brain in which synaptic consolidation enables continual learning by reducing the plasticity of synapses related to previous learned tasks

plasticity of weights closely related to previous tasks → more prone to cause catastrophic forgetting

EWC:

- quantify weights importance in terms of impact on previous tasks
- constraint parameters to low-error task regions
- selectively decrease the plasticity of those important weights to previous tasks

importance modeled as posterior distr. P(θ|D)
(prob. of parameter θ given a task's training data D)

Using Bayes rule:

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

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

all the information related to task A intractable...

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

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

Synaptic intelligence

Zenke et al. 2017

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

ELASTIC WEIGHT CONSOLIDATION

Kirkpatrick et al. [2017]

The idea:
inspired by human brain in which synaptic consolidation enables continual learning by reducing the plasticity of synapses related to previous learned tasks

plasticity of weights closely related to previous tasks → more prone to cause catastrophic forgetting

EWC:

- quantify weights importance in terms of impact on previous tasks
- constraint parameters to low-error task regions
- selectively decrease the plasticity of those important weights to previous tasks

importance modeled as posterior distr. P(θ|D)
(prob. of parameter θ given a task's training data D)

Using Bayes rule:

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

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

all the information related to task A intractable...

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

Fisher information computation is expensive for CL…
weight importance online, during SGD!

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

Synaptic intelligence

Zenke et al. 2017

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

ELASTIC WEIGHT CONSOLIDATION

Kirkpatrick et al. [2017]

The idea:
inspired by human brain in which synaptic consolidation enables continual learning by reducing the plasticity of synapses related to previous learned tasks

plasticity of weights closely related to previous tasks → more prone to cause catastrophic forgetting

EWC:

- quantify weights importance in terms of impact on previous tasks
- constraint parameters to low-error task regions
- selectively decrease the plasticity of those important weights to previous tasks

importance modeled as posterior distr. P(θ|D)
(prob. of parameter θ given a task's training data D)

Using Bayes rule:

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

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

all the information related to task A intractable...

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

Fisher information computation is expensive for CL…
weight importance online, during SGD!

loss change of a weight update step during SGD: $\Delta\mathcal{L}_k = \Delta\theta_k \cdot \frac{\partial\mathcal{L}}{\partial\theta_k}$

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

Synaptic intelligence
Zenke et al. 2017

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

ELASTIC WEIGHT CONSOLIDATION

Kirkpatrick et al. [2017]

The idea:
inspired by human brain in which synaptic consolidation enables continual learning by reducing the plasticity of synapses related to previous learned tasks

plasticity of weights closely related to previous tasks → more prone to cause catastrophic forgetting

EWC:

- quantify weights importance in terms of impact on previous tasks
- constraint parameters to low-error task regions
- selectively decrease the plasticity of those important weights to previous tasks

importance modeled as posterior distr. P(θ|D)
(prob. of parameter θ given a task's training data D)

Using Bayes rule:

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

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

all the information related to task A intractable...

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

Fisher information computation is expensive for CL…
weight importance online, during SGD!

loss change of a weight update step during SGD: $\Delta\mathcal{L}_k = \Delta\theta_k \cdot \frac{\partial\mathcal{L}}{\partial\theta_k}$

total loss change of a parameter: $\sum \Delta\mathcal{L}_k$

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

Synaptic intelligence
Zenke et al. 2017

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

ELASTIC WEIGHT CONSOLIDATION

Kirkpatrick et al. [2017]

The idea:
inspired by human brain in which synaptic consolidation enables continual learning by reducing the plasticity of synapses related to previous learned tasks

plasticity of weights closely related to previous tasks → more prone to cause catastrophic forgetting

EWC:

- quantify weights importance in terms of impact on previous tasks
- constraint parameters to low-error task regions
- selectively decrease the plasticity of those important weights to previous tasks

importance modeled as posterior distr. P(θ|D)
(prob. of parameter θ given a task's training data D)

Using Bayes rule:

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

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

all the information related to task A intractable...

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

Fisher information computation is expensive for CL…
weight importance online, during SGD!

loss change of a weight update step during SGD: $\Delta\mathcal{L}_k = \Delta\theta_k \cdot \frac{\partial\mathcal{L}}{\partial\theta_k}$

total loss change of a parameter: $\sum \Delta\mathcal{L}_k$

weight importance: $F_k = \frac{\sum \Delta\mathcal{L}_k}{T_k^2 + \xi}$

Where:
T_k is the total movement of weight θ_k during a batch training
ξ is a small constant to avoid division by 0

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

Synaptic intelligence

Zenke et al. 2017

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

AR1 model

$cw = 0$
init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)
$\Theta = 0$ (Θ are the optimal shared weights resulting from the last training, see Section 2.3)
$\hat{F} = 0$ ($\hat{F}$ is the weight importance matrix, see Section 3.3).
for each training batch B_i:
    expand output layer with s_i neuros for the new classes in B_i
    $tw = 0$ (for all neurons in the output layer)
    Train the model with SGD on the s_i classes of B_i by simultaneously:
        learn tw with no regularization
        learn $\bar{\Theta}$ subject to SI regularization according to $\hat{F}$ and Θ
    for each class j among the s_i classes in B_i:
        $cw[j] = tw[j] - avg(tw)$
    $\Theta = \bar{\Theta}$
    Update $\hat{F}$ according to trajectories computed on B_i (see eq. 7 and 8)
    Test the model by using $\bar{\Theta}$ and cw

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

Synaptic intelligence

Zenke et al. 2017

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

AR1 model

$cw = 0$

init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)

$\Theta = 0$ (Θ are the optimal shared weights resulting from the last training, see Section 2.3)

$\hat{F} = 0$ ($\hat{F}$ is the weight importance matrix, see Section 3.3).

for each training batch B_i:

  expand output layer with s_i neuros for the new classes in B_i

zero init → $tw = 0$ (for all neurons in the output layer)

  Train the model with SGD on the s_i classes of B_i by simultaneously:

    learn tw with no regularization

    learn $\bar{\Theta}$ subject to SI regularization according to $\hat{F}$ and Θ

  for each class j among the s_i classes in B_i:

    $cw[j] = tw[j] - avg(tw)$

  $\Theta = \bar{\Theta}$

  Update $\hat{F}$ according to trajectories computed on B_i (see eq. 7 and 8)

  Test the model by using $\bar{\Theta}$ and cw

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

Synaptic intelligence

Zenke et al. 2017

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

AR1 model

$cw = 0$

init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)

$\Theta = 0$ (Θ are the optimal shared weights resulting from the last training, see Section 2.3)

$\hat{F} = 0$ ($\hat{F}$ is the weight importance matrix, see Section 3.3).

for each training batch B_i:

  expand output layer with s_i neuros for the new classes in B_i

  $tw = 0$ (for all neurons in the output layer)

zero init

  Train the model with SGD on the s_i classes of B_i by simultaneously:

    learn tw with no regularization

    learn $\bar{\Theta}$ subject to SI regularization according to $\hat{F}$ and Θ

  for each class j among the s_i classes in B_i:

    $cw[j] = tw[j] - avg(tw)$

mean-shift

  $\Theta = \bar{\Theta}$

  Update $\hat{F}$ according to trajectories computed on B_i (see eq. 7 and 8)

  Test the model by using $\bar{\Theta}$ and cw

AR1: combining architectural and regularization strategies

Lomonaco and Maltoni (2017a) → *Copy weight with reinit (CWR)* model, architectural, effective in SIT scenario

Synaptic intelligence

Zenke et al. 2017

drawback of CWR and CWR+ → weights Θ are tuned during the first batch and then frozen

AR1 → CWR+ extend, allowing Θ to be tuned across batches subject to a regularization constraint (as per LWF, ECW or SI).

AR1 model

$cw = 0$

init $\bar{\Theta}$ random or from a pre-trained model (e.g. ImageNet)

$\Theta = 0$ (Θ are the optimal shared weights resulting from the last training, see Section 2.3)

$\hat{F} = 0$ ($\hat{F}$ is the weight importance matrix, see Section 3.3).

for each training batch B_i:

  expand output layer with s_i neuros for the new classes in B_i

  $tw = 0$ (for all neurons in the output layer) — zero init

  Train the model with SGD on the s_i classes of B_i by simultaneously:

    learn tw with no regularization

    learn $\bar{\Theta}$ subject to SI regularization according to $\hat{F}$ and Θ

  for each class j among the s_i classes in B_i:

    $cw[j] = tw[j] - avg(tw)$ — mean-shift

  $\Theta = \bar{\Theta}$ — Θ tuned

  Update $\hat{F}$ according to trajectories computed on B_i (see eq. 7 and 8)

  Test the model by using $\bar{\Theta}$ and cw

Up to now: architectural, regularization and rehearsal strategies to train deep models sequentially on a number of disjoint tasks

tasks are not disjoint (constitute a single incremental task, e.g. class-incremental learning.) → these strategies are unsatisfactory

main contributions of this paper:

- **point some differences between MT and SIT scenarios under NC update type**
- **review of popular CL approaches (LWF, EWC, SI)**
- **propose a new CL strategy (AR1) by extending CWR architectural strategy (introduced in Lomonaco and Maltoni (2017a) and combining it with a light regularization approach such as SI.**
- **compare AR1 with other common CL strategies on CORe50 and iCIFAR-100 benchmarks**
- **discuss diagnostic techniques to simplify hyperparameters tuning in complex CL settings and detect misbehaviors.**
- **additional material section: implementation details of all the techniques reviewed/proposed and the experiment carried out using the Caffe framework**

Experiments

two datasets: CORe50 and iCIFAR-100

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)

CORe50 dataset (Lomonaco & Maltoni, 2017a) was specifically designed
as a benchmark for continuous object recognition

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)

CORe50 dataset (Lomonaco & Maltoni, 2017a) was specifically designed
as a benchmark for continuous object recognition

NC content update type (a.k.a. incremental-class learning)

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)

CORe50 dataset (Lomonaco & Maltoni, 2017a) was specifically designed
as a benchmark for continuous object recognition

NC content update type (a.k.a. incremental-class learning)

50 classes partitioned in 9 batches provided sequentially:

B1 → 10 classes
B2 . . . B8 → 5 classes each

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)

CORe50 dataset (Lomonaco & Maltoni, 2017a) was specifically designed
as a benchmark for continuous object recognition

NC content update type (a.k.a. incremental-class learning)

50 classes partitioned in 9 batches provided sequentially:

B1 → 10 classes
B2 . . . B8 → 5 classes each

each class → 2.400 patterns (300 frames × 8 sessions) in training batches and 900 patterns (300 frames × 3 sessions) in test set

test set is fixed and accuracy evaluated after each batch on the whole test set

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)

CORe50 dataset (Lomonaco & Maltoni, 2017a) was specifically designed as a benchmark for continuous object recognition

Fig. 6. Example images of the 50 objects in CORe50. Each column denotes one of the 10 categories, but for the experiments reported in this paper we associate each object to a distinct class.

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)

approaches compared with two baselines:

- **Naïve:** tunes the model without specific mechanism to control forgetting;

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)

approaches compared with two baselines:

- **Naïve:** tunes the model without specific mechanism to control forgetting;

- **Cumulative:** model is retrained with patterns from the current and all the previous batches;

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)

approaches compared with two baselines:

- **Naïve:** tunes the model without specific mechanism to control forgetting;
- **Cumulative:** model is retrained with patterns from the current and all the previous batches;

considered as ideal performance that CL approaches should try to reach!

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)

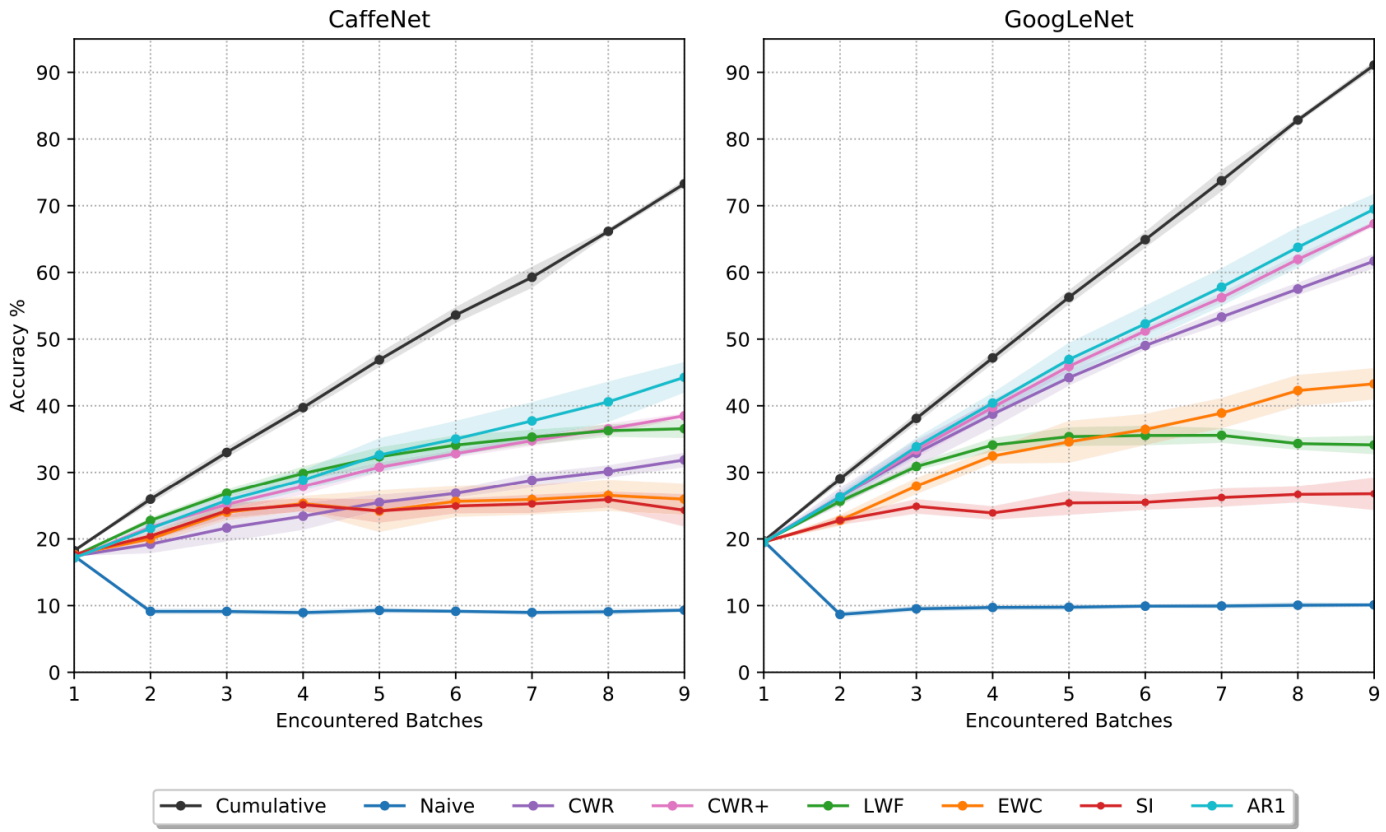
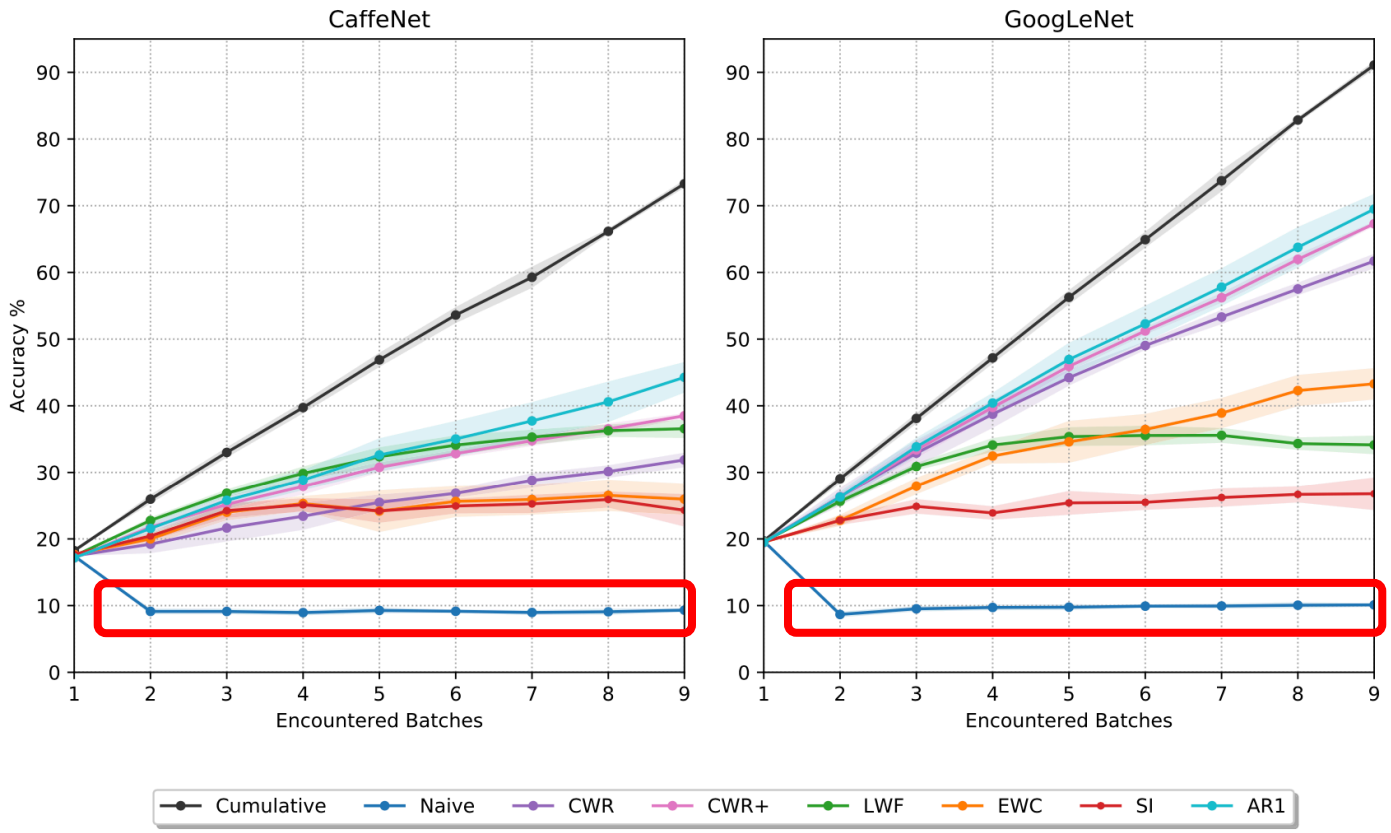


Fig. 7. The graphs show CaffeNet and GoogLeNet accuracy on CORe50 after each training batch (average on 10 runs, with different class ordering). Colored areas represent the standard deviation of each curve. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)



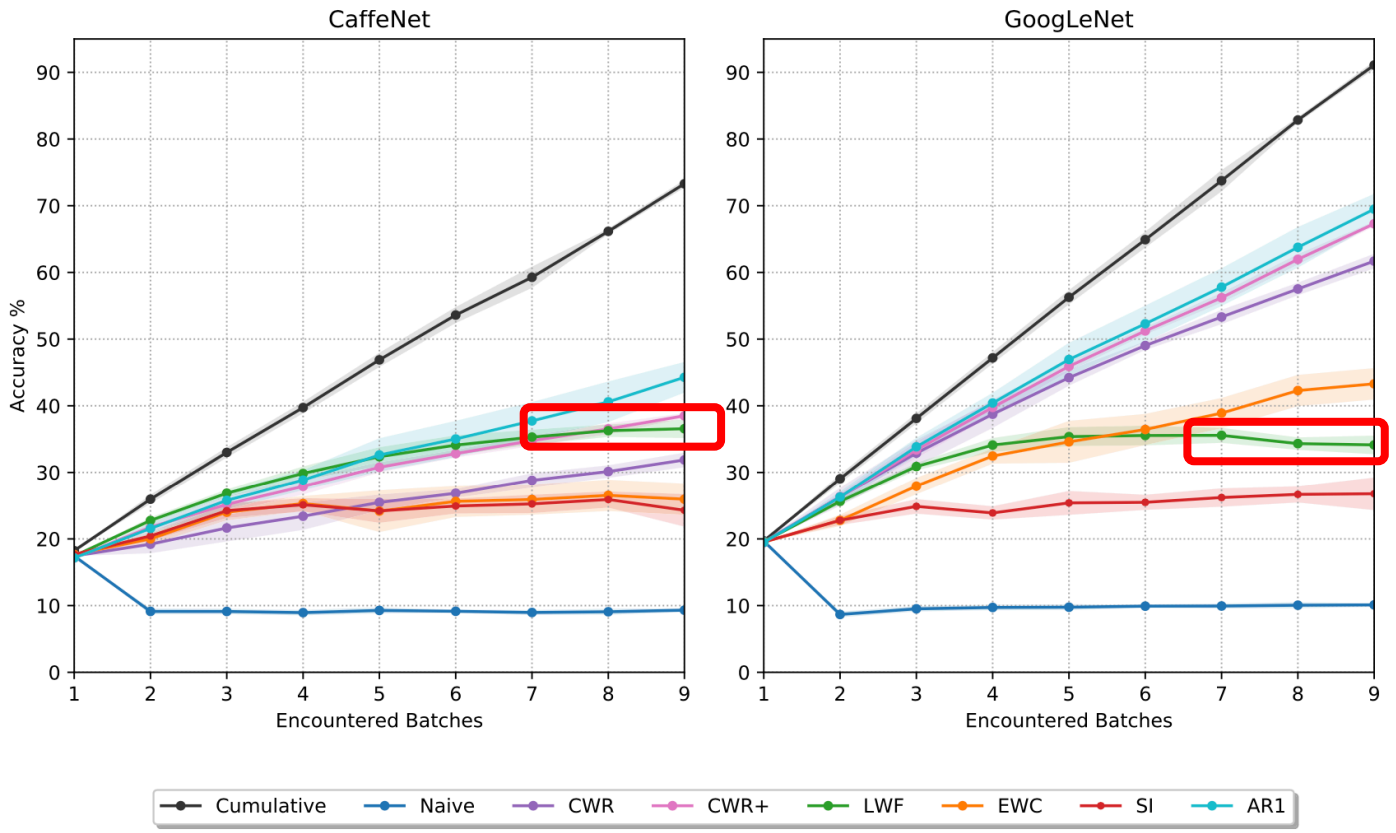
catastrophic forgetting is well evident for Naïve approach

Fig. 7. The graphs show CaffeNet and GoogLeNet accuracy on CORe50 after each training batch (average on 10 runs, with different class ordering). Colored areas represent the standard deviation of each curve. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)



catastrophic forgetting is well evident for Naïve approach

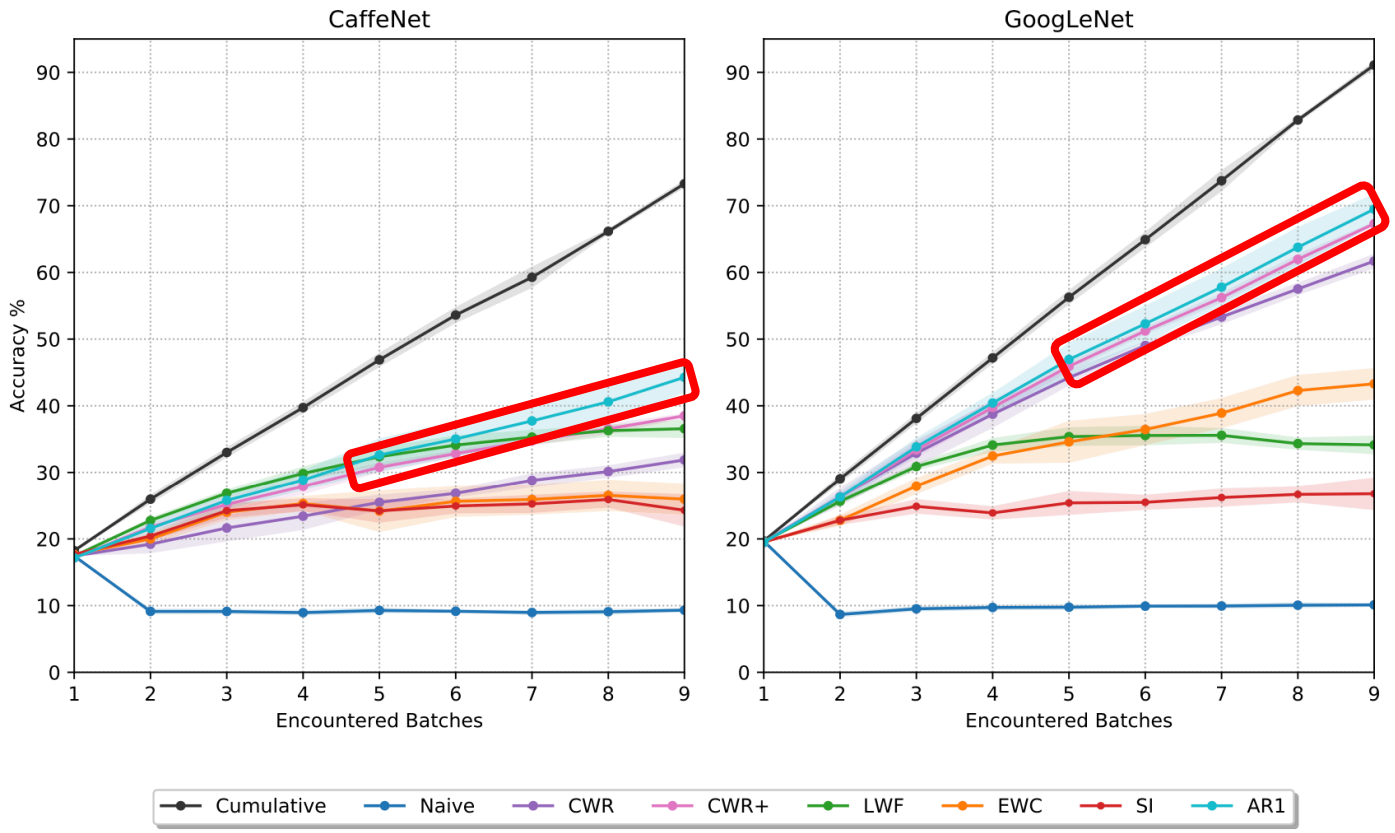
LWF well for CaffeNet and moderately well for GoogLeNet: CL is evident and accuracy better than Naive

Fig. 7. The graphs show CaffeNet and GoogLeNet accuracy on CORe50 after each training batch (average on 10 runs, with different class ordering). Colored areas represent the standard deviation of each curve. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Experiments

two datasets: **CORe50** and iCIFAR-100

two CNN architectures: CaffeNet (Jia et al., 2014) and GoogLeNet (Szegedy et al., 2015)



catastrophic forgetting is well evident for Naïve approach

LWF well for CaffeNet and moderately well for GoogLeNet: CL is evident and accuracy better than Naive

both models are able to learn incrementally with EWC, but for CaffeNet, after a few batches it tends to stabilize and the final accuracy is lower

Fig. 7. The graphs show CaffeNet and GoogLeNet accuracy on CORe50 after each training batch (average on 10 runs, with different class ordering). Colored areas represent the standard deviation of each curve. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Experiments

two datasets: CORe50 and **iCIFAR-100**

CNN architecture: 4 convolutional + 2 fully connected layers, Zenke et al. (2017)

CIFAR-100 (Krizhevsky, 2009): well-known and largely used dataset for small (32×32) natural image classification

Experiments

two datasets: CORe50 and **iCIFAR-100**

CNN architecture: 4 convolutional + 2 fully connected layers, Zenke et al. (2017)

CIFAR-100 (Krizhevsky, 2009): well-known and largely used dataset for small (32×32) natural image classification

100 classes containing 600 images each (500 training + 100 test)

default classification benchmark translated into a SIT-NC scenario (denoted as iCIFAR-100 by Rebuffi et al., 2017) by splitting the 100 classes in groups

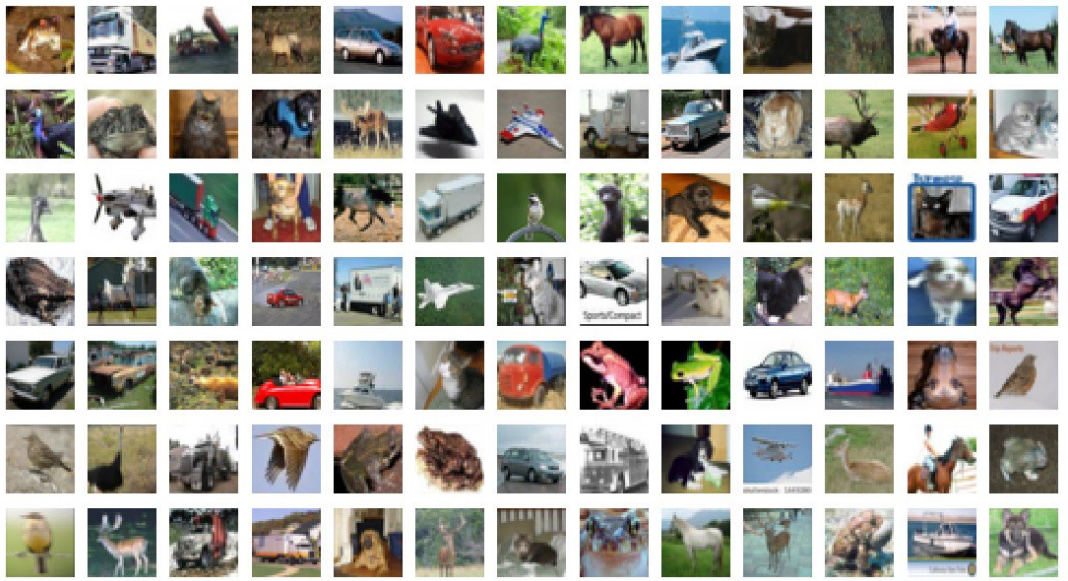
this paper → groups of 10 classes, thus obtaining 10 incremental batches

Experiments

two datasets: CORe50 and **iCIFAR-100**

CNN architecture: 4 convolutional + 2 fully connected layers, Zenke et al. (2017)

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor



Experiments

two datasets: CORe50 and **iCIFAR-100**

CNN architecture: 4 convolutional + 2 fully connected layers, Zenke et al. (2017)

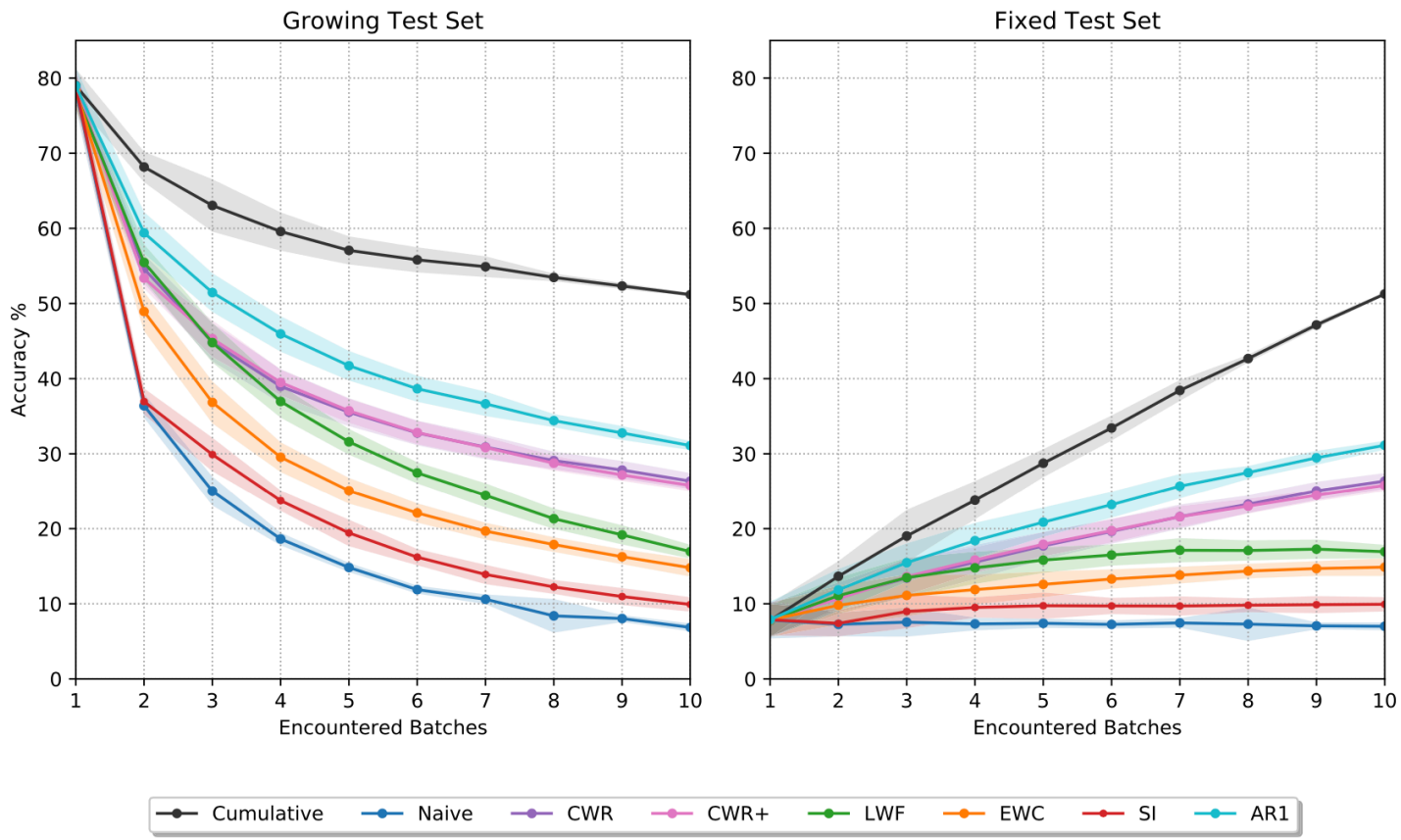


Fig. 8. Accuracy on iCIFAR-100 with 10 batches (10 classes per batch). Results are averaged on 10 runs: for all the strategies hyperparameters have been tuned on run 1 and kept fixed in the other runs. The experiment on the right, consistently with CORe50 test protocol, considers a fixed test set including all the 100 classes, while on the left we include in the test set only the classes encountered so far (analogously to results reported in Rebuffi et al., n.d.). Colored areas represent the standard deviation of each curve. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Experiments

two datasets: CORe50 and **iCIFAR-100**

CNN architecture: 4 convolutional + 2 fully connected layers, Zenke et al. (2017)

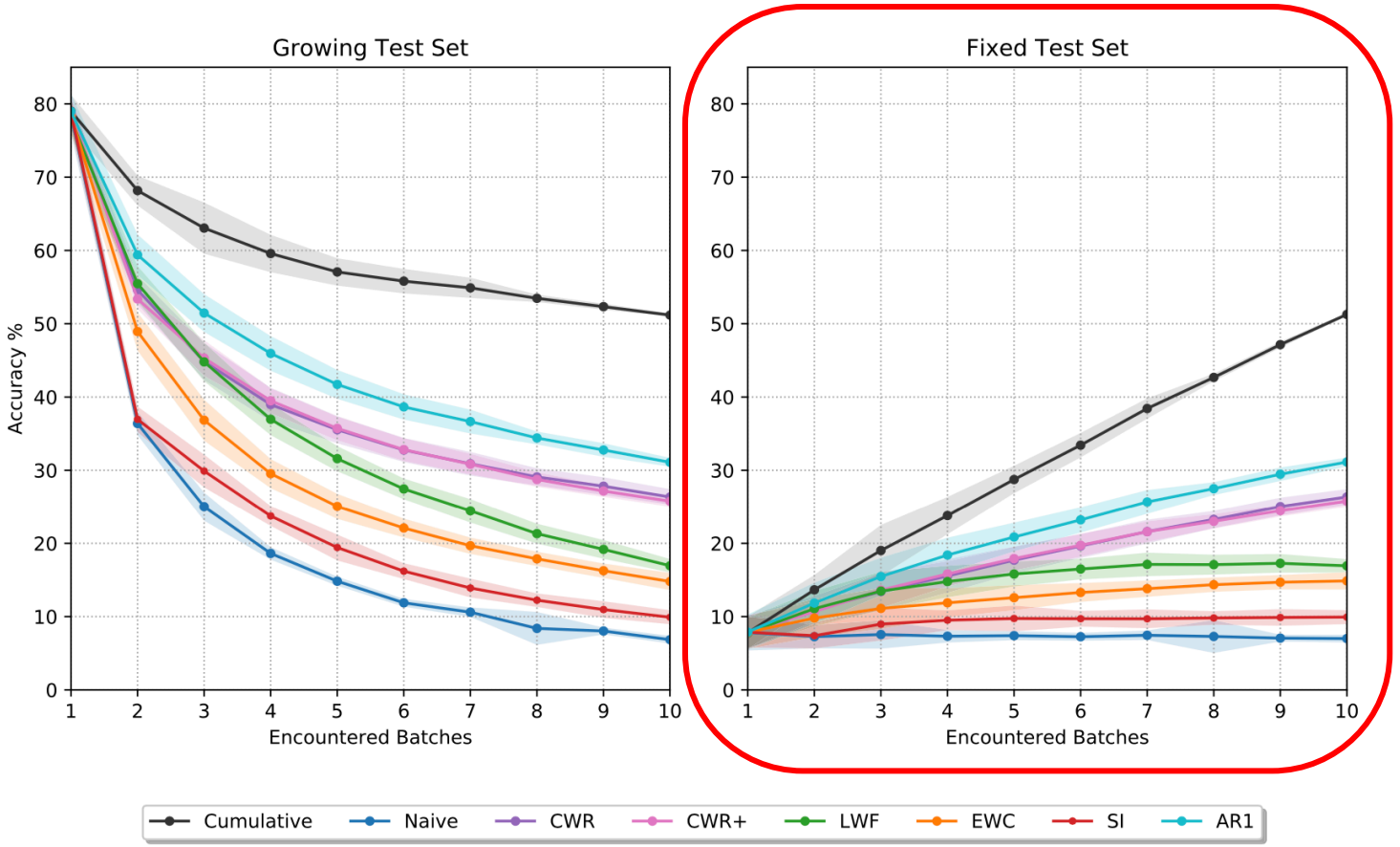


Fig. 8. Accuracy on iCIFAR-100 with 10 batches (10 classes per batch). Results are averaged on 10 runs: for all the strategies hyperparameters have been tuned on run 1 and kept fixed in the other runs. The experiment on the right, consistently with CORe50 test protocol, considers a fixed test set including all the 100 classes, while on the left we include in the test set only the classes encountered so far (analogously to results reported in Rebuffi et al., n.d.). Colored areas represent the standard deviation of each curve. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Up to now: architectural, regularization and rehearsal strategies to train deep models sequentially on a number of disjoint tasks

tasks are not disjoint (constitute a single incremental task, e.g. class-incremental learning.) → these strategies are unsatisfactory

main contributions of this paper:

- **point some differences between MT and SIT scenarios under NC update type**
- **review of popular CL approaches (LWF, EWC, SI)**
- **propose a new CL strategy (AR1) by extending CWR architectural strategy (introduced in Lomonaco and Maltoni (2017a) and combining it with a light regularization approach such as SI.**
- **compare AR1 with other common CL strategies on CORe50 and iCIFAR-100 benchmarks**
- **discuss diagnostic techniques to simplify hyperparameters tuning in complex CL settings and detect misbehaviors.**
- **additional material section: implementation details of all the techniques reviewed/proposed and the experiment carried out using the Caffe framework**

diagnostic techniques

simply looking at accuracy trend → not enough to understand if an approach is ok…

poor accuracy → insufficient learning of the new classes or forgetting the old ones?

diagnostic techniques

simply looking at accuracy trend → not enough to understand if an approach is ok…

poor accuracy → insufficient learning of the new classes or forgetting the old ones?

sequence of confusion matrices must be considered!

diagnostic techniques

simply looking at accuracy trend → not enough to understand if an approach is ok…

poor accuracy → insufficient learning of the new classes or forgetting the old ones?

sequence of confusion matrices must be considered!

Naïve approach on CaffeNet

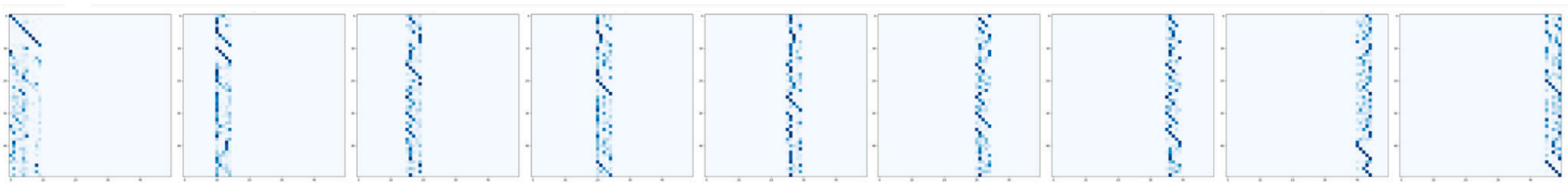


Fig. 10. Sequence of confusion matrices computed after each training batch for the Naïve approach on CaffeNet. On the vertical axis the true class and on the abscissa the predicted class.

diagnostic techniques

simply looking at accuracy trend → not enough to understand if an approach is ok…

poor accuracy → insufficient learning of the new classes or forgetting the old ones?

sequence of confusion matrices must be considered!

LWF and CaffeNet

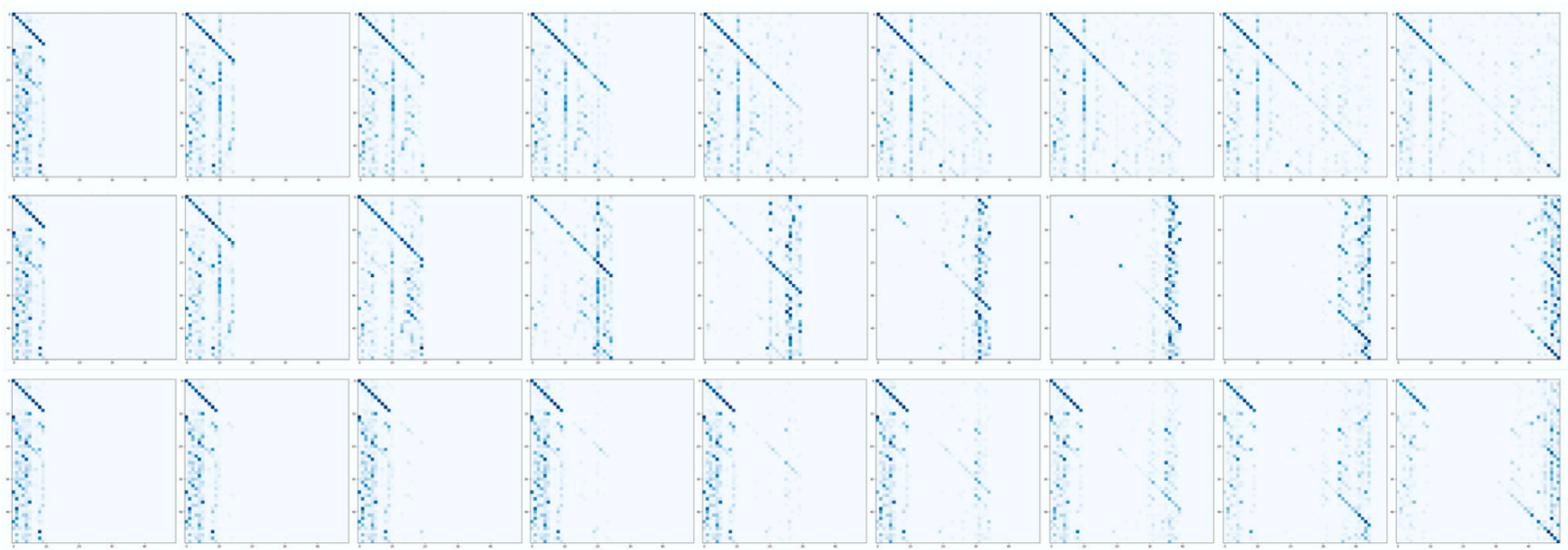


Fig. 11. Sequence of confusion matrices computed after each training batch (1…9) for the LWF approach and CaffeNet. In the first row the approach is properly parametrized (variable λ and *map* function, see Table 4) and the model is able to continuously learn new classes without forgetting the old ones. In the second we used the same $\lambda_i = 0.5$ for all the batches: for $B_2 \ldots B_3$ the regularization is appropriate but for successive batches is too light leading to excessive forgetting. In the third row we used the same $\lambda_i = 0.8$ for all the batches: for $B_2 \ldots B_6$ the regularization is too strong and learning of corresponding classes is poor.

diagnostic techniques

another useful diagnostic technique:

average amount of **weight changes** in each layer at the end of **each batch!**

diagnostic techniques

another useful diagnostic technique:

average amount of **weight changes** in each layer at the end of **each batch!**

p.s.: average of absolute values, to avoid cancellation.

CaffeNet

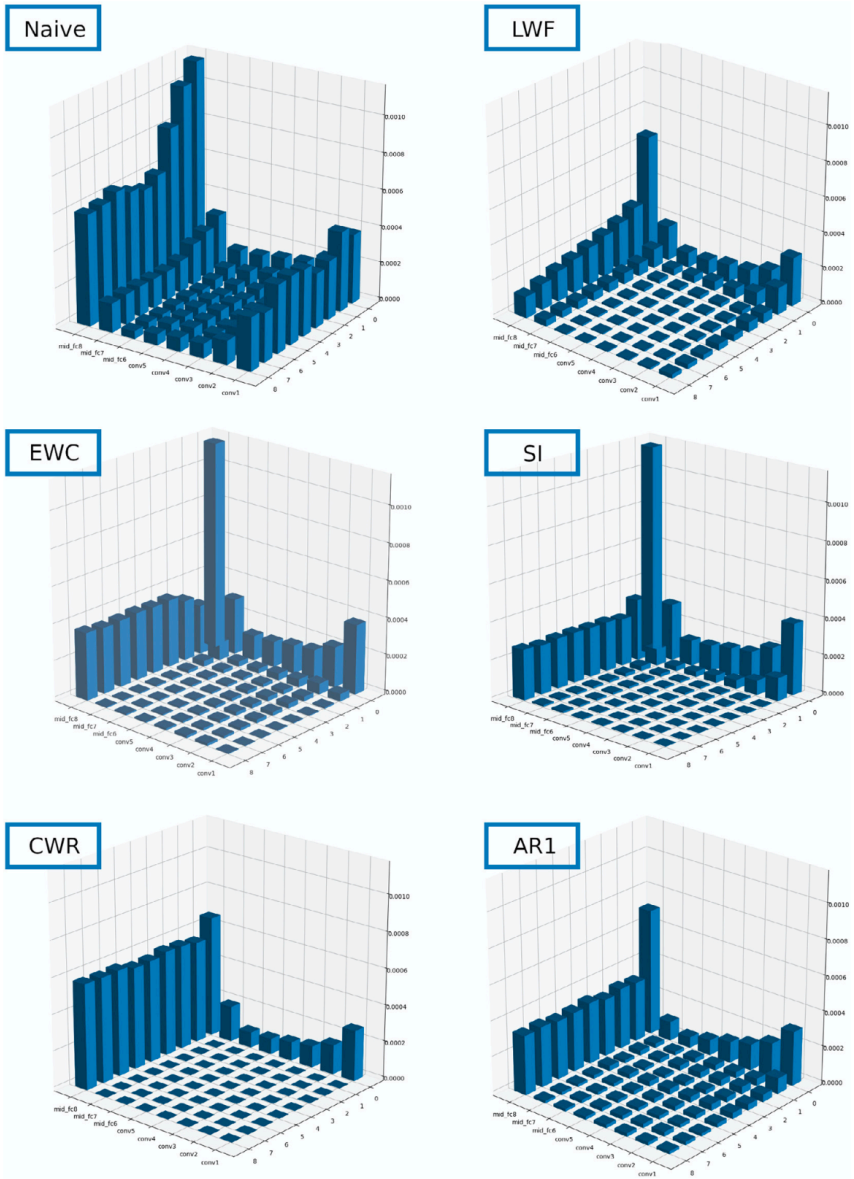


Fig. 13. Amount of weight changes by layer and training batch in CaffeNet for different approaches.

main conclusions

- introduced a novel approach (AR1) for SIT scenario combining architectural and regularization strategies;
- AR1 accuracy higher than existing regularization approaches such as LWF and EWC;
- diagnostic techniques suggestion;