Encoder Based Lifelong Learning - Supplementary materials

Amal Rannen* Rahaf Aljundi* Mathew B. Blaschko Tinne Tuytelaars
KU Leuven
KU Leuven, ESAT-PSI, IMEC, Belgium
firstname.lastname@esat.kuleuven.be

1. Analysis of the main method

In this section, we present a detailed analysis of the main contribution of the paper and demonstrate its theoretical grounding.

The following derivations are based on the hypothesis that all the functions involved in the model training are Lipschitz continuous. The most commonly used functions such as sigmoid, ReLU or linear functions satisfy this condition. It is also the case for the most commonly used loss functions, e.g. softmax, logistic, or hinge losses. Note that squared and exponential losses are not Lipschitz continuous, but this is not a significant limitation as such losses are less frequently applied in practice due to their sensitivity to label noise.

Definition 1. *We say that a function f is Lipschitz continuous if and only if there exist a constant K such that:*

$$\forall (x,y), \|f(x) - f(y)\| \leq K\|x - y\|$$

1.1. Relation between Encoder based Lifelong Learning and joint training

In the sequel, we use the same notation as in the main paper. In a two-task scenario, we propose to use the following objective to train a network using only the data from the second task:

$$\begin{aligned}\mathcal{R} = \mathbb{E}[&\ell(T_2 \circ T \circ F(\mathcal{X}^{(2)}), \mathcal{Y}^{(2)})) \\ &+ \ell_{dist}(T_1 \circ T \circ F(\mathcal{X}^{(2)}), T_1^* \circ T^* \circ F^*(\mathcal{X}^{(2)})) \\ &+ \frac{\alpha}{2}\|\sigma(W_{enc}F(\mathcal{X}^{(2)})) - \sigma(W_{enc}F^*(\mathcal{X}^{(2)}))\|_2^2]. \quad (1)\end{aligned}$$

In the ideal case where we can keep in memory data from the first task, the best solution is to train the network jointly by minimizing the following objective:

$$\mathbb{E}[\ell(T_2 \circ T \circ F(\mathcal{X}^{(2)}), \mathcal{Y}^{(2)}))] + \mathbb{E}[\ell(T_1 \circ T \circ F(\mathcal{X}^{(1)}), \mathcal{Y}^{(1)}))] \quad (2)$$

*Authors with equal contribution

Proposition 1. *The difference between* (2) *and* $\mathbb{E}[\ell(T_2 \circ T \circ F(\mathcal{X}^{(2)}), \mathcal{Y}^{(2)})) + \ell(T_1 \circ T \circ F(\mathcal{X}^{(2)}), T_1^* \circ T^* \circ F^*(\mathcal{X}^{(2)}))]$ *can be controlled by the independent minimization of the knowledge distillation loss and five terms:* (7)*,* (8)*,* (9)*,* (10)*, and* (11)*.*

Proof. From Lipschitz continuity, we deduce that the difference between (2) and $\mathbb{E}[\ell(T_2 \circ T \circ F(\mathcal{X}^{(2)}), \mathcal{Y}^{(2)})) + \ell(T_1 \circ T \circ F(\mathcal{X}^{(2)}), T_1^* \circ T^* \circ F^*(\mathcal{X}^{(2)}))]$ is bounded, and we can write for any vector norm $\|.\|$:

$$\begin{aligned}|\ell(&T_1 \circ T \circ F(\mathcal{X}^{(1)}), \mathcal{Y}^{(1)}) \\ &- \ell(T_1 \circ T \circ F(\mathcal{X}^{(2)}), T_1^* \circ T^* \circ F^*(\mathcal{X}^{(2)}))| \\ &\leq K_1\|F(\mathcal{X}^{(1)}) - F(\mathcal{X}^{(2)})\| \quad (3) \\ &+ K_2\|\mathcal{Y}^{(1)} - T_1^* \circ T^* \circ F^*(\mathcal{X}^{(2)})\|. \quad (4)\end{aligned}$$

(4) is related to the classification error on the first task. Indeed, using the triangle inequality, we can write:

$$\begin{aligned}\|\mathcal{Y}^{(1)} &- T_1^* \circ T^* \circ F^*(\mathcal{X}^{(2)})\| \\ &\leq \|\mathcal{Y}^{(1)} - T_1^* \circ T^* \circ F^*(\mathcal{X}^{(1)})\| \quad (5) \\ &+ \|T_1^* \circ T^* \circ F^*(\mathcal{X}^{(1)}) - T_1^* \circ T^* \circ F^*(\mathcal{X}^{(2)})\| \quad (6)\end{aligned}$$

Note that all the terms on the right hand side of the inequality do not change during training of task 2, and thus cannot be controlled during the second training phase. Moreover, (5) depends only on the capacity of the network and is therefore not influenced by the encoder based lifelong learning scheme. (6) is the result of using $\mathcal{X}^{(2)}$ instead of $\mathcal{X}^{(1)}$. In order to reduce the effect of this shift, we use the knowledge distillation loss [1] as in the Learning without Forgetting (LwF) method [3].

On the other hand, expression (3) is bounded as well (us-

ing again the triangle inequality):

$$\|F(\mathcal{X}^{(1)}) - F(\mathcal{X}^{(2)})\| \leq \|F(\mathcal{X}^{(1)}) - F^*(\mathcal{X}^{(1)})\| \quad (7)$$
$$+ \|F^*(\mathcal{X}^{(1)}) - r \circ F^*(\mathcal{X}^{(1)})\| \quad (8)$$
$$+ \|r \circ F^*(\mathcal{X}^{(1)}) - r \circ F^*(\mathcal{X}^{(2)})\| \quad (9)$$
$$+ \|r \circ F^*(\mathcal{X}^{(2)}) - r \circ F(\mathcal{X}^{(2)})\| \quad (10)$$
$$+ \|r \circ F(\mathcal{X}^{(2)}) - F(\mathcal{X}^{(2)})\|. \quad (11)$$

This bound generalizes trivially to the expected value of the loss, which finishes the proof. □

Analyzing each of these terms individually, we see that our training strategy effectively controls the difference with the joint training risk:

- (8) is minimized through the autoencoder (AE) training and does not change during training of task 2.
- (9) does not change during training of task 2. Moreover, $\| r \circ F^*(\mathcal{X}^{(1)}) - r \circ F^*(\mathcal{X}^{(2)})\|$ measures the distance between two elements of the manifold that the AE represents. As the use of weight decay during training the task model makes the weights small, the projection of data into the feature space is contractive. Through our experiments, we observed that r is also contractive. As a result, this distance is significantly smaller than $\|\mathcal{X}^{(1)} - \mathcal{X}^{(2)}\|$. The experiment in Sec. 1.5 supports this observation.
- (10) **is controlled during the training thanks to the second constraint we propose to use**.
- (11) is the part of the features that we propose to relax in order to give space for the model to adjust to the second task as explained in Sec. 3.3 in the main text. Indeed, if we control this distance, the features are forced to stay in the manifold related to the first task. This will result in a stronger conservation of the first task performance, but the training model will face the risk of not being able to converge for the second task.
- (7) is a term that we cannot access during training. However, we observed that this distance is either decreasing or first increasing then decreasing during the training. An explanation of this behavior is that in the beginning $\ell(T_2 \circ T \circ F(\mathcal{X}^{(2)}), \mathcal{Y}^{(2)}))$ may have a bigger influence on the objective, however, after decreasing the loss for the second task, $\ell(T_1 \circ T \circ F(\mathcal{X}^{(2)}), T_1^* \circ T^* \circ F^*(\mathcal{X}^{(2)}))$ and $\|\sigma(W_{enc}F(\mathcal{X}^{(2)})) - \sigma(W_{enc}F^*(\mathcal{X}^{(2)}))\|_2$ tend to push F towards F^*. Figure 2 and Section 1.4 support this observation.

This derivation motivates the code distance that we propose to use. It also elucidates the sources of possible divergence from joint-training.

1.2. Multiple task scenario

Each time a task is added, a new source of divergence from the joint-training objective is added. The difference with (2) grows with $\mathcal{T}$. Indeed, at each step, an extra irreducible $\|r \circ F^*(\mathcal{X}^{(\mathcal{T}-1)}) - r \circ F^*(\mathcal{X}^{(\mathcal{T})})\|$ is added. Moreover, for each task, the autoencoders of the previous tasks are trained using the corresponding feature extractors. The remaining difference between the two losses that is neither directly controlled nor decreasing while training the model can be expressed as follows (for a constant K):

$$K\Bigg(\sum_{t=2}^{\mathcal{T}} \|r_{t-1} \circ F^{(*,t-1)}(\mathcal{X}^{(t-1)}) - r_{t-1} \circ F^{(*,t-1)}(\mathcal{X}^{(t)})\| \quad (12)$$
$$+ \sum_{t=2}^{\mathcal{T}} \|r_{t-1} \circ F(\mathcal{X}^{(\mathcal{T})}) - F(\mathcal{X}^{(\mathcal{T})})\| \quad (13)$$
$$+ \sum_{t=1}^{\mathcal{T}-2} \|r_t \circ F^{(*,\mathcal{T}-1)}(\mathcal{X}^{(t)}) - r_t \circ F^{(*,t)}(\mathcal{X}^{(t)})\|\Bigg), \quad (14)$$

where $F^{(*,t)}$ is the feature extraction operator after training the model on the task t data. As observed for (9), expressions (12) and (14) remain small and the conclusion of the experiment in Sec. 1.5 holds also for these distances. Therefore, their effect on the growth of the difference with joint-training is minimal. In contrast to the other terms, as for (11), controlling (13) may prevent the network from converging for the new task. Thus, relaxing these distances is an important degree of freedom for our method.

1.3. Autoencoder training: choice of λ

As stated in the main text, the autoencoder training aims to solve the following minimization problem:

$$\arg\min_r \mathbb{E}_{(\mathcal{X}^{(1)},\mathcal{Y}^{(1)})}[\lambda\|r(F^*(\mathcal{X}^{(1)})) - F^*(\mathcal{X}^{(1)})\|_2 \quad (15)$$
$$+ \ell(T_1^* \circ T^*(r(F^*(\mathcal{X}^{(1)}))), \mathcal{Y}^{(1)})],$$

where ℓ is the loss function used to train the model on the first task data, and λ is a hyper-parameter that controls the compromise between the two terms in this loss. In our experiments, λ is tuned manually in order to allow the convergence of the code loss and the classification loss on the training data. Figure 1 shows the evolution of these losses

for the training and validation samples of ImageNet during the training of an autoencoder based on the $conv5$ features extracted with AlexNet and VGG-verydeep-16. λ is set to 10^{-6} in all cases, as this value makes both the code loss and the classification loss decrease.

1.4. Behavior Analysis

To examine the effect of our representation control over the learning process, we perform an analysis on the distance between the representation obtained over the learning procedure and the one that is optimal for the first task. We use Flower $\rightarrow$ Scenes$\rightarrow$ Birds as a test case and compute the distance for each epoch between the current features of the Flowers dataset $F(\mathcal{X}^1)$ and that obtained by the initial Flowers network $F^*(\mathcal{X}^1)$, as shown in Figure 2. In practice, we aimto minimize:

$$R_N = \frac{1}{N}\sum_{i=1}^{N}\Bigg(\ell(T_T \circ T \circ F(X_i^{(\mathcal{T})}), Y_i^{(\mathcal{T})}) + \sum_{t=1}^{\mathcal{T}-1}\ell_{dist}(T_t \circ T \circ F(X_i^{(\mathcal{T})}), T_t^* \circ T^* \circ F^*(X_i^{(\mathcal{T})})) + \sum_{t=1}^{\mathcal{T}-1}\frac{\alpha_t}{2}\|\sigma(W_{enc,t}F(X_i^{(\mathcal{T})})) - \sigma(W_{enc,t}F^*(X_i^{(\mathcal{T})}))\|_2^2\Bigg). \tag{16}$$

In the beginning of the training, the leading term in Eq. (16) is the loss related to the new task $\mathcal{T}$. Thus, in the first stage, the model is driven towards optimizing the performance of the most recent task. This results in a quick loss of performance for the previous tasks, and an increase in the other loss terms of the objective. Then, the second stage kicks in. In this stage, all the terms of Eq. (16) contribute and the model is pushed towards recovering its performance for the previous tasks while continuing improving for the most recent one. Gradually, $F(\mathcal{X}^1)$ gets again closer to $F^*(\mathcal{X}^1)$, until a new equilibrium is reached.

1.5. Empirical study: F and r are contractive

This experiment aims to show empirically that $\|\mathcal{X}^{(1)} - \mathcal{X}^{(2)}\|$ is significantly larger than $\|r \circ F^*(\mathcal{X}^{(1)}) - r \circ F^*(\mathcal{X}^{(2)})\|$ using the ℓ_2 norm. To have empirical evidence for this observation, we conducted the following experiment:

1. First, we generate random inputs for the model from two different distributions. We use normal distributions with uniformly distributed means and variances.
2. Then, we compute the features of these inputs (output of F).
3. These features are fed to the AE to compute the reconstructions.
4. The mean squared error (MSE) between the samples, the features and the reconstructions are stored.
5. This procedure is repeated for several trials. Each time a different pair of distributions is used. Finally, the mean of the obtained MSE is computed.

	Samples	Features	Reconstructions
Exp. 1	229.98	33.05	0.21
	$\pm$0.065	$\pm$0.066	$\pm$0.0024
Exp. 2	10176.54	106.60	0.21
	$\pm$41.17	$\pm$0.32	$\pm$0.0027

Table 1. Showing that F and r are contractive: Mean MSE of 50 samples from random Gaussian distribution, of their corresponding features and reconstructions over 500 trails - The main model and the AE are trained on Flowers dataset. As we move from left to right in the table, the entries in the columns decrease significantly verifying that the mappings are contractive.

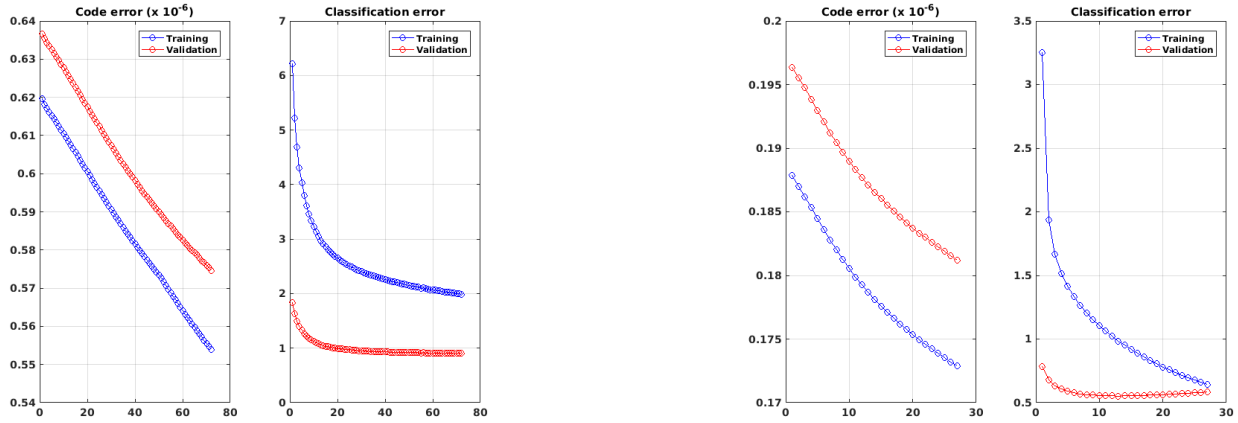
	Samples	Features	Reconstructions
Exp. 1	230.02	9.48	1.55
	$\pm$0.063	$\pm$0.022	$\pm$0.01
Exp. 2	10173.11	61.29	2.12
	$\pm$41.35	$\pm$0.18	$\pm$0.013

Table 2. Showing that F and r are contractive: Mean MSE of 50 samples from random Gaussian distribution, of their corresponding features and reconstructions over 500 trails - The main model and the AE are trained on Imagenet dataset. As we move from left to right in the table, the entries in the columns decrease significantly verifying that the mappings are contractive.

We repeat this experiment twice. In the first instance, the mean and variance of the Gaussian distributions are generated in a way to have relatively small distances between the samples, and in the second we force the samples to have bigger distance. Tables 1 and 2 show the results of this experiment. The numbers in Table 1 are computed using AlexNet fine-tuned on Flowers dataset, and the AE trained as explained in Sec. 3.3 (in the main text) on the features of Flowers extracted using AlexNet convolutional layers after convergence. We report in this table the mean MSE and the error bars obtained with 50 generated samples from 500 different pairs of Gaussian distributions. The numbers in Table 2 are computed similarly but from the AlexNet and the AE related to ImageNet. In all cases, the difference between the samples is many orders of magnitude larger than the difference between the reconstructions indicating that the mapping is indeed contractive, and the residual error from this step is minimal.

1.6. Choice of F and T

In this work, we select F to be the convolutional layers and T to be the fully connected (fc) layers. Here, we present



(a) Alexnet (b) VGG

Figure 1. Training of Alexnet and VGG-verydeep-16 based autoencoders for ImageNet - The objective makes the code loss *and* the classification loss decrease. The training is stopped when we observe a convergence of the classification loss.

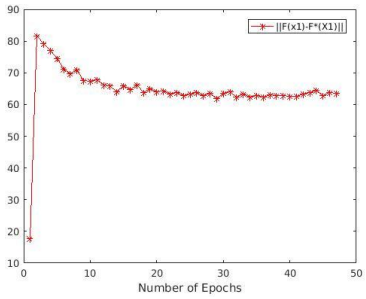


Figure 2. Distance between the representation obtained during training of the Birds task given Flowers samples and the original representation for Flowers network. Starting from the Scenes network trained using our method after Flowers.

the motivation behind our choice. The fc-layers project the output of convolutional blocks to decreasing dimensions. Therefore, the later we apply the autoencoder, the less informative features it captures. Consequently, when we apply our loss after the shared task operator T, we expect more forgetting. We run experimnets with our loss placed after each of the fully connection layers. However, unfortunately, in our deep networks there is a dropout layer between the fc-layers. For AlexNet, learning the autoencoder at the end of the shared task operator (fc7) means that when training the new task, the projection of the samples on the submanifold may change drastically between epochs because the operator itself is changing. As a result the code loss will give an inaccurate estimation of the distance which in turn will not prevent the forgetting. Therefore, in our experiments, using our loss after fc7 could not help in reducing the forgetting over LwF. When placing the autoencoder after fc6 before the dropout, we get more forgetting than with the autoencoder after conv5, in the case of ImageNet $\rightarrow$ CUB : 55.12% vs 55.3%, significantly different (N-1 Chi-squared test, p = 0.3215).

2. Additional experiments

The experiments in this section aim to show that the success of the proposed method is independent of the chosen model. For this purpose, we train VGG-verydeep-16 [6] in a two-task image classification scenario and compare our method against the state-of-the-art (LwF) [3].

Tested scenario We test our method and LwF on a two-task scenario starting from ImageNet (LSVRC 2012 subset) [5] (more than 1 million training images) then training on MIT *Scenes* [4] for indoor scene classification (5,360 samples). The showed results are obtained on the test data of Scenes and the validation data of ImageNet.

Architecture We experiment with VGG-verydeep-16 [6] due to its popularity and success in many image classification tasks. The feature extraction block F corresponds to the convolutional layers. Note that this architecture has feature extractor twice as deep as AlexNet [2] (used in the main text). As for the experiments conducted using AlexNet, the shared task operator T corresponds to all but the last fully connected layers (i.e., $fc6$ and $fc7$), while the task-specific part T_i contains the last classification layer ($fc8$). The used hyperparameters are the same as in the main text: same α (10^{-3}) for the training of our method, and same λ (10^{-6}) for the autoencoder training. The used architecture for the autoencoder is also similar to the main text (2-layers with a sigmoid non-linearity in between) with a code of 300 entries. Figure 1(b) shows the evolution of the code error and the classification error during the training of the

Model	Size	Feature size	Autoencoder size
AlexNet	449 MB	9216	10 MB (2.2%)
VGG	1.1 GB	25088	28 MB (2.5%)

Table 3. Size of the used autoencoders compared to the size of the models during training. The model size corresponds to the required memory during training. The feature size corresponds to the length of the feature extractor output.

autoencoder on ImageNet features extracted using VGG-verydeep-16.

Autoencoder size To illustrate the growth of the size of the autoencoder with the size of the model, we show in Table 3 the memory required by AlexNet and VGG-verydeep-16 while training for Scenes after ImageNet, along with the autoencoder input length (Feature size) and the memory required by the autoencoder during training with our method. Naturally, the autoencoder size grows with the feature length, but remains very small comparing with the size of the global model.

Results: Method behavior comparison Figure 3 shows the evolution of the model performance for both tasks, ImageNet and Scenes, when trained with our method (red curves) and with LwF (blue curves).

Our method shows a better preservation of the performance on ImageNet. Even if the classification error grows for both methods, it increases slower in our case. After 20 epochs, the performance on ImageNet is 1% higher using our method.

The test and train errors on Scenes highlight an interesting characteristic of our method. The use of the code loss on top of LwF appears to act as a regularizer for the training on Scenes. Our method shows a slightly higher training error, and a better generalization. VGG-verydeep-16 is a large model, and the risk of overfitting while training on a small dataset like Scenes is higher than for AlexNet. A stronger regularization (using a higher value of α) may thus result in an improvement of the behavior of our method.

Conclusion From this experiment, we observe that the convergence of the autoencoder training and the improvement observed over LwF are not dependent on the used architecture. Moreover, the additional memory required by our method remains small with respect to the size of the global model.

References

[1] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *stat*, page 9, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Z. Li and D. Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016.

[4] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420, 2009.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 211–252, 2015.

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

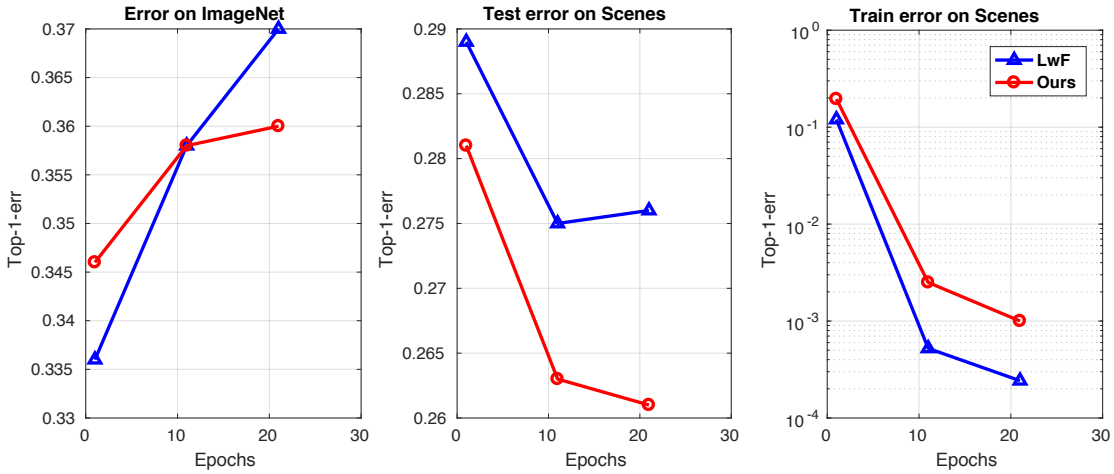


Figure 3. Comparison between our method (red) and LwF (blue). Left : Evolution of the error on ImageNet validation set; it shows a slower loss of performance on ImageNet for our method - Center: Evolution of the error on Scenes test set - Right: Evolution of the error on Scenes training set. Center and Right suggest that our method benefits from a better regularization than LwF.