Adding New Tasks to a Single Network with Weight Transformations using Binary Masks

Massimiliano Mancini[1,2], Elisa Ricci[2,3], Barbara Caputo[4], Samuel Rota Bulò[5]

[1]Sapienza University of Rome, [2]Fondazione Bruno Kessler,[3]University of Trento, [4]Italian Institute of Technology, [5]Mapillary Research
{mancini,caputo}@diag.uniroma1.it,eliricci@fbk.eu,samuel@mapillary.com

Abstract. Visual recognition algorithms are required today to exhibit adaptive abilities. Given a deep model trained on a specific, given task, it would be highly desirable to be able to adapt incrementally to new tasks, preserving scalability as the number of new tasks increases, while at the same time avoiding catastrophic forgetting issues. Recent work has shown that masking the internal weights of a given original conv-net through learned binary variables is a promising strategy. We build upon this intuition and take into account more elaborated affine transformations of the convolutional weights that include learned binary masks. We show that with our generalization it is possible to achieve significantly higher levels of adaptation to new tasks, enabling the approach to compete with fine tuning strategies by requiring slightly more than 1 bit per network parameter per additional task. Experiments on two popular benchmarks showcase the power of our approach, that achieves the new state of the art on the Visual Decathlon Challenge.

Keywords: Incremental Learning, Multi-task Learning

1 Introduction

A long-standing goal of AI is the ability to adapt an initial, pre-trained model to novel, unseen scenarios. This is crucial for increasing the knowledge of an intelligent system and developing effective life-long learning [41, 42, 38] algorithms. While fascinating, achieving this goal requires facing multiple challenges. First, learning a new task should not negatively affect the performance on old tasks, avoiding the catastrophic forgetting phenomenon [6, 8]. Second, it should be avoided adding multiple parameters to the model for each new task learned, as it would lead to poor scalability of the framework [31]. In this context, while deep learning algorithms have achieved impressive results on many computer vision benchmarks [17, 11, 7, 22], mainstream approaches for adapting deep models to novel tasks tend to suffer from the problems mentioned above.

Different works addressed these problems by either considering regularization techniques [21, 14] or task-specific network parameters [36, 31, 34, 24, 25]. Interestingly, in [25] the authors effectively addressed sequential multi-task learning by creating a binary mask for each task. This mask is then multiplied by the main network weights, determining which of them are useful for addressing the new task and requiring just one bit for each parameter per task.

Our paper takes inspiration from this last work. We formulate sequential multi-task learning as the problem of learning a perturbation of a *baseline*, pre-trained network, maximizing the performance on a new task. As opposed to [25], we apply an affine transformation to each convolutional weight of the baseline network, involving both a learned binary mask and few additional parameters. Our solution allows to: 1) boosting the performance of each task-specific network, by leveraging the higher degree of freedom in perturbing the baseline network; 2) keeping a low per-task overhead in terms of additional parameters (slightly more than 1 bit per parameter per task). We assess the validity of our method on standard benchmarks, achieving performances comparable with fine-tuning separate networks for each task.

2 Related works

The keen interest on incremental and life-long learning methods dates back to the pre-convnet era, with shallow learning approaches ranging from large margin classifiers [18, 19] to non-parametric methods [27, 33].

Recently, various works have addressed these problems within the framework of deep architectures [31, 10, 1]. A major risk when training a neural network on a novel task is to deteriorate its performances on old tasks, discarding previous knowledge, a phenomenon called *catastrophic forgetting* [26, 6, 8]. To alleviate this issue, various works designed constrained optimization procedures taking into account the initial network weights, trained on previous tasks. In [21], the authors exploit knowledge distillation [13] to obtain target objectives for previous tasks, while training for novel ones. In [14] the authors design an update of the network parameters, based on their importance for previously seen tasks.

Recent methods achieved higher performances with the cost of adding task specific parameters for each newly learned task, keeping untouched the initial network parameters. The extreme case is [36], where a parallel network is added each time a new task is presented. In [31, 32], task-specific residual components are added in standard residual blocks. In [34] the authors use controller modules where the parameters of the base architecture are recombined channel-wise. In [24] a different subset of network parameters is considered for each task. A more compact and effective solution is [25], where separate binary masks are learned for each novel task and multiplied to the original network weights. The binary masks determine which parameters are useful for the new task and which are not. We take inspiration from this last work but we use the binary masks to design task specific affine transformations through. This allows us to use a comparable number of parameters per task with increased flexibility, further reducing the gap with the individual end-to-end trained architectures.

3 Method

We address the problem of sequential multi-task learning, as in [25], *i.e.* we modify a *baseline* network such as, *e.g.* ResNet-50 pretrained on the ImageNet classification task, so to maximize its performance on a new task, while limiting the amount of additional parameters needed. The solution we propose exploits the key idea from Piggyback [25] of learning task-specific masks, but instead of pursuing the simple multiplicative transformation of the parameters of the

baseline network, we define a parametrized, affine transformation mixing a binary mask and real parameters. This choice keeps a low per-task overhead while significantly increases the expressiveness of the approach, leading to a rich and nuanced ability to adapt the old parameters to the needs of the new tasks.

3.1 Overview

Let us assume to be given a pre-trained, *baseline* network $f_0(\cdot;\Theta,\Omega_0):\mathcal{X}\to\mathcal{Y}_0$ assigning a class label in $\mathcal{Y}_0$ to elements of an input space $\mathcal{X}$ (*e.g.* images).[1] The parameters of the baseline network are partitioned into two sets: Θ comprises parameters that will be shared for other tasks, whereas Ω_0 entails the rest of the parameters (*e.g.* the classifier). Our goal is to learn for each task $i\in\{1,\dots,\mathsf{m}\}$, with a possibly different output space $\mathcal{Y}_i$, a classifier $f_i(\cdot;\Theta,\Omega_i):\mathcal{X}\to\mathcal{Y}_i$. Here, Ω_i entails the parameters specific for the ith task, while Θ holds the shareable parameters of the baseline network mentioned above. Before delving into the details of our method, we review the Piggyback solution presented in [25].

Each task-specific network f_i shares the same structure of the baseline network f_0, except for having a possibly, differently sized classification layer. All parameters of f_0, excepting the classifier, are shared across all the tasks. For each convolutional layer[2] of f_0 with parameters $\mathtt{W}$, the task-specific network f_i holds a binary mask $\mathtt{M}$ that is used to mask $\mathtt{W}$ obtaining

$$\hat{\mathtt{W}}=\mathtt{W}\circ\mathtt{M}\,, \tag{1}$$

where $\circ$ is the Hadamard product. The transformed parameters $\hat{\mathtt{W}}$ are then used in the convolutional layer of f_i. By doing so, the task-specific parameters that are stored in Ω_i amount to just a single bit per parameter in each convolutional layer, yielding a low overhead per additional task, while retaining a sufficient degree of freedom to build new convolutional weights.

Proposed. Similarly to [25], we consider task-specific networks f_i that are shaped as the baseline network f_0 and we store in Ω_i a binary mask $\mathtt{M}$ for each convolutional kernel $\mathtt{W}$ in the shared set Θ. However, we depart from the simple multiplicative transformation of $\mathtt{W}$ used in (1), and consider instead an affine transformation of the base convolutional kernel $\mathtt{W}$ that depends on a binary mask $\mathtt{M}$ as well as additional parameters. Specifically, we transform $\mathtt{W}$ into

$$\check{\mathtt{W}}=k_0\mathtt{W}+k_1\mathtt{1}+k_2\mathtt{M}\,, \tag{2}$$

where $k_j\in\mathbb{R}$ are additional task-specific parameters in Ω_i that we learn along with the binary mask $\mathtt{M}$, and $\mathtt{1}$ is an opportunely sized tensor of 1s. We can consider either a scale (k_2) and bias (k_1) parameter per convolutional kernel, or distinct values for each feature channel.

Besides learning the binary masks and the parameters k_j, we opt also for task-specific batch-normalization (BN) parameters (*i.e.* mean, variance, scale and bias), which will be part of Ω_i, and thus optimized for each task, rather than being fixed in Θ. In the cases where we have a convolutional layer followed by BN, we keep the corresponding parameter k_0 fixed to 1, because the output of batch normalization is invariant to the scale of the convolutional weights.

[1] We focus on classification tasks, but the proposed method applies also to other tasks.
[2] Fully-connected layers are a special case.

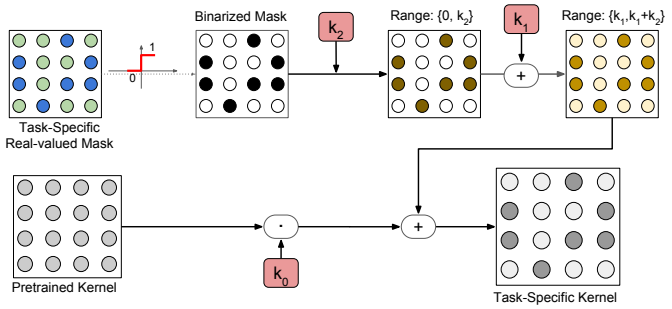

Fig. 1: Proposed model. An affine transformation scale and translate the binary masks through the parameters k_2 and k_1 respectively. The obtained mask is summed to the pretrained kernel in order to obtain the final task-specific weights.

The additional parameters introduced with our method bring a negligible per-task overhead compared to Piggyback, which is nevertheless generously balanced out by a significant boost of the performance of the task-specific classifiers.

3.2 Learning Binary Masks

We learn the parameters Ω_i of each task-specific network f_i by minimizing the classification log-loss, given a training set, using standard, stochastic optimization methods. However, special care should be taken for the optimization of the binary masks. Instead of optimizing the binary masks directly, which would turn the learning into a combinatorial problem, we apply the solution adopted in [25], *i.e.* we replace each binary mask `M` with a thresholded real matrix `R`. By doing so, we shift from optimizing discrete variables in `M` to continuous ones in `R`. However, the gradient of the hard threshold function $h(r) = 1_{r \geq 0}$ is zero almost everywhere, which makes this solution apparently incompatible with gradient-based optimization approaches. To sidestep this issue we consider a strictly increasing, surrogate function $\tilde{h}$ that will be used in place of h *only* for the gradient computation, *i.e.* if h' denotes the derivative of h with respect to its argument, we use $h'(r) \approx \tilde{h}'(r)$. The gradient obtained via the surrogate function has the property that it always points in the right down hill direction in the error surface.

By taking $\tilde{h}(x) = x$, *i.e.* the identity function, we recover the workaround suggested in [12], employed also in [25]. By taking $\tilde{h}(x) = (1 + e^{-x})^{-1}$, *i.e.* the sigmoid function, we obtain a better approximation, as suggested in [9, 2].

4 Experiments

Datasets. In the following we test our method on two different benchmarks. For the first benchmark we follow [25], and we use 6 datasets: ImageNet [35], VGG-Flowers [30], Stanford Cars [15], Caltech-UCSD Birds (CUBS) [43], Sketches [5] and WikiArt [37]. These datasets contain a lot of variations both from the category addressed (*i.e.* cars [15] vs birds [43]) and the appearance of their instances (*i.e.* from natural images [35] to art paintings [37] and sketches [5]).

The second benchmark is the Visual Decathlon Challenge [31]. The goal of this challenge is to use a single algorithm tackle 10 different classification tasks: ImageNet [35], CIFAR-100 [16], Aircraft [23], Daimler pedestrian (DPed) [28], Describable textures (DTD) [4], German traffic signs (GTSR) [40], Omniglot

[20], SVHN [29], UCF101 Dynamic Images [3, 39] and VGG-Flowers [30]. A more detailed description of the challenge can be found in [31]. For this challenge, an independent scoring function is defined: the S-score [31]. This score takes into account the performances of a model on all 10 tasks, preferring models with good performances on all tasks to ones with peaked performances in few of them.

Networks and training protocols. For the first benchmark, we use a ResNet-50, comparing our model with Piggyback [25], PackNet [24] and two baselines considering the network only as feature extractor (training only the task-specific classifier) and individual networks separately fine-tuned on each task. Since [24] is dependent on the order of the task, we report the performances for two different orderings [25]: starting from the model pre-trained on ImageNet, the first ($\rightarrow$) is CUBS-Cars-Flowers-WikiArt-Sketch while the second ($\leftarrow$) is reversed. For training, we followed the preprocessing, hyper-parameters and schedule of [25].

For the Visual Decathlon we employ the Wide ResNet-28 [44] adopted by previous methods [31, 34, 25], using the same data preprocessing. For training we choose the same hyper-parameters of [25], *keeping the same values for all the tasks* except the ImageNet pretraining, for which we followed [31]. For both benchmarks we employ $\tilde{h}(x) = x$ as surrogate, initializing the real-valued masks with uniform random values drawn between 0.0001 and 0.0002.

4.1 Results

ImageNet-to-Sketch. In the following we discuss the results obtained by our model on the ImageNet-to-Sketch scenario. For fairness, since our model includes task-specific BN layers, we report also the results of [25] with separate BN layers.

Results are shown in Table 1. Our model is able to fill the gap between the classifier only baseline and the individual fine-tuned architectures, almost entirely in all settings. For larger and more diverse datasets such as Sketch and WikiArt, the gap is not completely covered, but the distance between our model and the individual architectures is always less than 1%. These results are remarkable given the simplicity of our method, not involving any assumption of the optimal weights per task [24, 21], and the small overhead in terms of parameters that we report in the row "# Params" (*i.e.* 1.17), which represents the total number of parameters (counting all tasks and excluding the classifiers) relative to the ones in the baseline network. Comparing with the other algorithms, our model consistently outperforms both the basic version of Piggyback and PackNet in all settings. Introducing task-specific BN also for Piggyback reduces the performance gap, which still remains large in some settings (*i.e.* Flowers, Cars): this show how the advantages of our model are not only due to the additional BN parameters, but also to the more flexible affine transformation introduced.

Both Piggyback and our model outperform PackNet and, as opposed to the latter, do not suffer from the heavily dependence on the ordering of the tasks. This advantage stems from having a learning strategy that is task independent, with the base network not affected by the new tasks that are learned.

Visual Decathlon Challenge. In this section we report the results for the Visual Decathlon Challenge. We compare our model with other sequential multi-task learning methods: Piggyback [25] (PB), the improved version of the winner

Table 1: Accuracy of ResNet-50 architectures in the ImageNet-to-Sketch setting.

Model	ImageNet	CUBS	Cars	Flowers	WikiArt	Sketch	# Params
Classifier Only [25]	76.2	70.7	52.8	86.0	55.6	50.9	1
PackNet $\rightarrow$ [24]	75.7	80.4	86.1	93.0	69.4	76.2	1.10
PackNet $\leftarrow$ [24]	75.7	71.4	80.0	90.6	70.3	78.7	1.10
Piggyback [25]	**76.2**	80.4	88.1	93.5	73.4	79.4	1.16
Piggyback+BN [25]	**76.2**	82.1	90.6	95.2	74.1	79.4	1.17
Ours	**76.2**	**82.6**	**91.5**	**96.5**	**74.8**	**80.2**	1.17
Individual Networks [25]	76.2	82.8	91.8	96.6	75.6	80.8	6

Table 2: Results in terms of accuracy and S-Score, for the Visual Decathlon Challenge. Best model in bold, second best underlined.

Method	#Params	ImNet	Airc.	C100	DPed	DTD	GTSR	Flwr.	Oglt.	SVHN	UCF	Mean	S-Score	Score/Params
Feature [31]	1	59.7	23.3	63.1	80.3	45.4	68.2	73.7	58.8	43.5	26.8	54.3	544	544
Finetune [31]	10	59.9	60.3	82.1	92.8	55.5	97.5	81.4	87.7	96.6	51.2	76.5	2500	250
RA[31]	2	59.7	56.7	<u>81.2</u>	93.9	50.9	97.1	66.2	<u>89.6</u>	96.1	47.5	73.9	2118	1059
DAN [34]	2.17	57.7	64.1	80.1	91.3	56.5	98.5	86.1	**89.7**	<u>96.8</u>	<u>49.4</u>	<u>77.0</u>	2852	1314
PA [32]	2	<u>60.3</u>	<u>64.2</u>	**81.9**	94.7	58.8	**99.4**	84.7	89.2	96.5	**50.9**	**78.1**	**3412**	1706
PB [25]	1.28	57.7	**65.3**	79.9	**97.0**	57.5	97.3	79.1	87.6	**97.2**	47.5	76.6	2838	<u>2217</u>
PB ours	1.28	**60.8**	52.3	80.0	<u>95.1</u>	**59.6**	98.7	82.9	85.1	96.7	46.9	75.8	2805	2191
Ours	1.29	**60.8**	51.3	**81.9**	94.7	<u>59.0</u>	<u>99.1</u>	<u>88.0</u>	89.3	96.5	48.7	76.9	<u>3263</u>	**2529**

entry of the 2017 edition of the challenge [34] (DAN), the network with task-specific residual [31] (RA) and parallel [32] (PA) adapters. We additionally report the baselines of [31]: the pre-trained network used as feature extractor (Feature) and 10 different models fine-tuned on each task (Finetune). Moreover, we add the results of our implementation of [25] with the same pre-trained model and training schedule adopted for our method (PB ours).

The results are reported in Table 2. We can see that our simple model achieves close to state-of-the-art performances on this competition. The only model outperforming ours is [32]: however, we employ a much lower parameters overhead and a single training schedule for all ten tasks. This produces a gain of more than 800 points with respect to [32] in the ratio between the S-Score and the number of parameters adopted. Remarkably, we obtain a gain on the previous winning entry [34] and Piggyback of more than 400 points.

From the partial results, excluding the ImageNet baseline, our model achieves the top-1 or top-2 scores in 4 out of 9 tasks, with comparable performances in the others. The only exceptions are UCF-101 and Aircraft, where our model suffers a high accuracy drop. Tuning the hyper-parameters could cover this gap, but this is out of the scope of this work. Interestingly, while our model achieves comparable (*e.g.* PB, DAN) average accuracy with respect to other approaches, it obtains a much higher decathlon score. This highlights its capabilities of tackling all 10 tasks with good results, without peaked accuracies on just few of them.

5 Conclusions

We presented a simple yet powerful method for learning incrementally new tasks, given a fixed, pre-trained deep architecture. We build on the intuition of [25], generalizing the idea of masking the original weights of the network with learned binary masks. By introducing an affine transformation that acts upon such weights, we allow for a richer set of possible modifications of the original network,allowing to better capture the characteristics of the new tasks. Experiments on two public benchmarks confirm the effectiveness of our approach.

References

1. Bendale, A., Boult, T.E.: Towards open set deep networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 1563–1572 (2016)
2. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
3. Bilen, H., Fernando, B., Gavves, E., Vedaldi, A., Gould, S.: Dynamic image networks for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3034–3042 (2016)
4. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. pp. 3606–3613. IEEE (2014)
5. Eitz, M., Hays, J., Alexa, M.: How do humans sketch objects? ACM Trans. Graph. **31**(4), 44–1 (2012)
6. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
7. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
8. Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A., Bengio, Y.: An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211 (2013)
9. Goodman, R.M., Zeng, Z.: A learning algorithm for multi-layer perceptrons with hard-limiting threshold units. In: Neural Networks for Signal Processing [1994] IV. Proceedings of the 1994 IEEE Workshop. pp. 219–228. IEEE (1994)
10. Guerriero, S., Caputo, B., Mensink, T.: Deep nearest class mean classifiers. In: International Conference on Learning Representations, Worskhop Track (2018)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
12. Hinton, G.: Neural networks for machine learning (2012), coursera, video lectures.
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
14. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (2017)
15. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on. pp. 554–561. IEEE (2013)
16. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
18. Kuzborskij, I., Orabona, F., Caputo, B.: From N to N+1: multiclass transfer incremental learning. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013. pp. 3358–3365 (2013)

19. Kuzborskij, I., Orabona, F., Caputo, B.: Scalable greedy algorithms for transfer learning. Computer Vision and Image Understanding **156**, 174–185 (2017)
20. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. Science **350**(6266), 1332–1338 (2015)
21. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence (2017)
22. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)
23. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013)
24. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. arXiv preprint arXiv:1711.05769 (2017)
25. Mallya, A., Lazebnik, S.: Piggyback: Adding multiple tasks to a single, fixed network by learning to mask. arXiv preprint arXiv:1801.06519 (2018)
26. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24, pp. 109–165. Elsevier (1989)
27. Mensink, T., Verbeek, J.J., Perronnin, F., Csurka, G.: Distance-based image classification: Generalizing to new classes at near-zero cost. IEEE Trans. Pattern Anal. Mach. Intell. **35**(11), 2624–2637 (2013)
28. Munder, S., Gavrila, D.M.: An experimental study on pedestrian classification. IEEE transactions on pattern analysis and machine intelligence **28**(11), 1863–1868 (2006)
29. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning. vol. 2011, p. 5 (2011)
30. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on. pp. 722–729. IEEE (2008)
31. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Learning multiple visual domains with residual adapters. In: Advances in Neural Information Processing Systems. pp. 506–516 (2017)
32. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Efficient parametrization of multi-domain deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8119–8127 (2018)
33. Ristin, M., Guillaumin, M., Gall, J., Gool, L.J.V.: Incremental learning of random forests for large-scale image classification. IEEE Trans. Pattern Anal. Mach. Intell. **38**(3), 490–503 (2016)
34. Rosenfeld, A., Tsotsos, J.K.: Incremental learning through deep adaptation. arXiv preprint arXiv:1705.04228 (2017)
35. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
36. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)
37. Saleh, B., Elgammal, A.: Large-scale classification of fine-art paintings: Learning the right metric on the right feature. arXiv preprint arXiv:1505.00855 (2015)

38. Silver, D.L., Yang, Q., Li, L.: Lifelong machine learning systems: Beyond learning algorithms. In: AAAI Spring Symposium: Lifelong Machine Learning. vol. 13, p. 05 (2013)
39. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
40. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural networks **32**, 323–332 (2012)
41. Thrun, S., Mitchell, T.M.: Lifelong robot learning. Robotics and autonomous systems **15**(1-2), 25–46 (1995)
42. Thrun, S., Pratt, L.: Learning to learn. Springer Science & Business Media (2012)
43. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
44. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)