Improving SeNA-CNN by Automating Task Recognition*

Abel Zacarias[1][0000−0002−0226−9682] and Luís A. Alexandre[1][0000−0002−5133−5025]

Instituto de Telecomunicações,
Universidade da Beira Interior, Rua
Marquês d'Ávila e Bolama, 6201-001
Covilhã, Portugal
{abel.zacarias, luis.alexandre}@ubi.pt

Abstract. Catastrophic forgetting arises when a neural network is not capable of preserving the past learned task when learning a new task. There are already some methods proposed to mitigate this problem in artificial neural networks. In this paper we propose to improve upon our previous state-of-the-art method, SeNA-CNN, such as to enable the automatic recognition in test time of the task to be solved and we experimentally show that it has excellent results. The experiments show the learning of up to 4 different tasks with a single network, without forgetting how to solve previous learned tasks.

Keywords: Supervised Learning, Lifelong learning, Catastrophic Forgetting, Convolutional Neural Networks.

1 Introduction

Deep learning has emerged as one of the most important tools to deal with the large amount of data available today. One of more widely used methods to process image data are Convolutional Neural Networks (CNNs), that have demonstrated human level ability in many computer vision tasks.

Many practical applications, for instance, in robotics, require the agent to learn new capabilities without forgetting the previous learned ones. To be able to do this using neural networks, one is confronted with the problem of catastrophic forgetting, where the network forgets previous tasks as it learns new ones. So to build lifelong learning systems, there are several proposals (a short review is provided in section 2). In this paper we improve a proposal we made recently [15] where we add layers to an existing CNN such that it can cope with new tasks, without requiring the network to train again on old tasks. This is achieved by selectively adding new layers to an existing model trained on isolated learning.

The improvement we present focus on the automatic selection of the branch of the network that should deal with a new test pattern. This is accomplished

* This work was supported by National Founding from the FCT- Fundação para a Ciência e a Tecnologia, through the UID/EEA/50008/2013 Project. The GTX Titan X used in this research was donated by the NVIDIA Corporation.

using a gate neural network. This an interesting solution given that there is biological support for an identical approach: in [8] they showed that this gate mechanism is possible in a pre-frontal cortex of a primate where there are neural representations to select the relevant inputs.

The gate network learns to distinguish between the different learned tasks and therefore, makes the SeNA-CNN a completely automatic solution to the problem of lifelong learning in CNNs. The results presented in the experiments section show that our method is able to deal with the catastrophic forgetting problem and presents better results than a state-of-the-art method [6].

2 Related Work

The problem of catastrophic forgetting is a big issue in machine learning and artificial intelligence if the goal is to build a system that learns through time, and is able to deal with more than a single problem. According to [7], without this capability we will not be able to build truly intelligent systems, we can only create models that solve isolated problems in a specific domain. There are some recent works that tried to overcome this problem, e.g., the Learning without Forgetting (LwF) algorithm proposed in [6] adds nodes to an existing network for a new task only in the fully connected layers and this approach demonstrated to preserve the performance on old tasks without re-training the old tasks after a new one is learned. We compare SeNA-CNN with LwF algorithm. The first main difference is that, instead of adding nodes in fully connected layers, we add convolutional and fully connected layers for the new tasks to an existing model, creating branches for each task; the second one, is that we use a gate to enable automatic recognition in test time. The improved method has a better capability of learning new problems than LwF because we train a series of convolutional and fully connected layers while LwF only trains the added nodes in the fully connected layer and hence, depends on the original task's learned feature extractors to represent the data from all problems to be learned. Apart from that, LwF is not able to automatically recognize the problem to be solved, as the SeNA-CNN improvement we propose in this paper.

A mixture of experts for large-scale image categorization was proposed in [3] which consists of a tree-structured network architecture. The generalist is a CNN optimized on jointly train over all classes and is used to select which expert to process the input data. We also use a CNN as our gate, also with joint train, to help choosing which branch to predict a test image. Differently from our approach, this approach was used for image categorizations only (not applied to lifelong learning), showing that using a tree-structured network architecture can yield a substantial improvement in accuracy over the base CNN trained for example on CIFAR100 and ImageNet.

Our approach, when training the gate, is to make it able to distinguish between the data that is to be processed by each different branch of the SeNA-CNN. To that end, similar to what is done in [3], we cast the definition of the gate as the problem of learning a label mapping $\ell : \mathrm{I} \mapsto \mathrm{Z}$, where I is the input image,

$Z = \{0, 1, 2, ..n-1\}$ and n is the number of branches, which is the same as the number of tasks. The gate then indicates which branch is to used for making the classification of a given pattern.

Another difference between the work in [3] and our proposal is that the tree-structured network consists of single trunk feeding all branches, one branch for each expert and the trunk is initialized with convolutional layers of the generalist. We do not follow this approach, we simply use the output of the gate to choose which model to predict at a given time.

3 Proposed Method to Overcome Catastrophic Forgetting

Our proposal is a method that is able to preserve the performance on old tasks while learning new tasks, using only training data from the old tasks for adjusting the gate. Note that other approaches, as [6], are not able to detect automatically to which of the learned tasks the input image belongs to.

A model that is capable of learning two or more tasks has several advantages against that which only learns one task. First advantage is that the previous learned task can help better and faster learning the new task. Second, the model that learns multiple tasks may result in more universal knowledge and it can be used as a key to learn new task domains [13].

The gate network is trained to separate the images from the different tasks automatically. After training the gate, a CNN is created with random initialization of its weights. The network is then trained until convergence for the first task. Figure 1(a) presents the model for the first task trained on isolated learning and Figure 1(b) is our proposed model with the gate model used to choose automatically a task to deploy at test time. In Figure 1(b) the blue colour represents the old task network and the orange corresponds to the new added nodes for the new tasks.

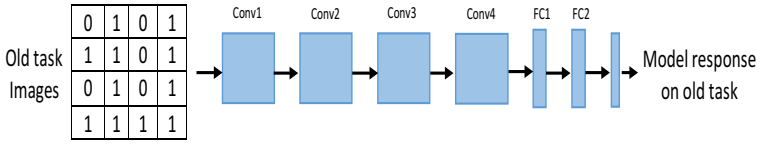
When a new tasks is going to be learned instead of adding nodes in fully connected layers as is done in [6], we add convolutional, pooling, dropout and fully connected layers for the new task. Typically the added layers contain a structure similar to the network that we trained on isolated learning. We consider the option of not adding the first two layers, because the neurons in those layers find several simple structures, such as oriented edges as demonstrated in [11]. The remaining layers seem to be devoted to more complex objects, and hence, are more specific to each problem, and that is why we choose to create these new layers instead of re-using the original ones. It also resembles the idea of mini-columns in the brain [9]. We add those layers and train them initialized with weights of an old task, keeping the other task layers frozen.

When switching to a third task and fourth task, we freeze the previous learned tasks and only train the new added layers. This process can be generalized to any number of tasks that we wish to learn.

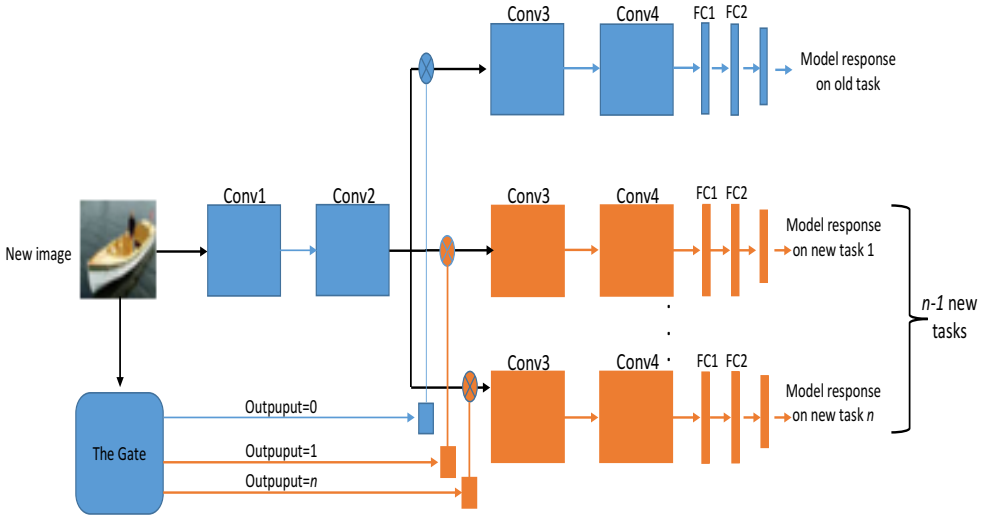
3.1 Using a Gate to Select the Correct Branch

The goal at this stage is to learn a gate neural network with a superclass of the branches that process each task. Our main goal is that the gate represent all tasks to be used at test time to choose automatically which branch to deploy. The gate is trained by joint optimization of images from all tasks. In our approach the gate network has the same architecture as each branch. The only difference is the size of the output that changed each time new task was added to our method. The gate network is initialized randomly and each task is labelled $\{0, 1, ..., n-1\}$, where n represents the total number of tasks to be deployed. This way, there is no need for all branches to process the input and produce an output and hence, only one branch is chosen to make the final decision.

Figure 1 shows this process when a new image is received by the proposed model.



(a) Original model for old task trained on isolated learning.



(b) Proposed model: the gate and the added new layers for the $n-1$ new tasks.

Fig. 1: Original and our proposal used in the experiment process to avoid the catastrophic forgetting by selective network augmentation. The blue coloured boxes correspond to the old task and the orange coloured correspond to the added layers. Adapted from [15].

3.2 Training Methodology

The network is going to learn how to solve several tasks. First the gate is trained by using images from each task, with labels that represent those tasks (and not the original task classes). Second, we train a first network branch, starting with randomly initialized weights. Then, for each new task, a new branch is added to the network, where the first two convolutional layers are reused from the first branch.

Figure 1 (a) shows the model trained on the first task. After all parameters have been optimized we use the first two convolutional layers of this network to help learn the new tasks as they come. Figure 1 (b) illustrates the overall architecture of the proposed method.

During training, we followed the same practice as [6]. The main difference is that when adding a new task we freeze all layers of the original model and only train the added nodes, freezing the first two layers that are common to all branches. During train we use back-propagation with SGD algorithm with dropout enabled. All branches had the same architecture, and the learning rate was set to 0.01, weight decay of $1e-6$ and momentum 0.9. Table 2 shows the performance of each task trained isolated after 12 training epochs. We run each experiment ten times and present results corresponding to the mean and standard deviation of these 10 repetitions. We run our experiments using a GeForce GTX TITAN X with 12 GiB.

4 Experiments

Our experiments evaluate if the proposed method can effectively avoid the catastrophic forgetting problem and if the gate can be used to automatically choose which task to deploy at test time. The experimental results presented in the following tables, contain the accuracy values of our proposed method. Is important to refer that we begin by training the gate to learn all features that are necessary to distinguish which branch to use for each input image. In all tables we present the accuracy of the gate, which is very important because the final accuracy of our method is calculate taking in count the errors that the gate made during the train and the error that each branch make when the branch is choose to predict the new task. This process was done because during the experiment the gate did not classified correctly all images and if we discards those images, the comparison would not be fair.

We conducted our experiments using four known datasets namely CIFAR10 [2], Caltech101 [5], SVHN2 [10] and Caltech-UCSD Birds-200-2011 [14]. Table 1 shows information on each dataset, and the number of images on training and test sets. CIFAR10 has 10 classes, SVHN2 corresponds to street house numbers and has 11 classes, Caltech101 has 102 classes while Birds has 200 classes. The last one has a substantially lower accuracy than the others, as shown in table 2. The first two datasets were obtained from `kerosene` that provides six datasets namely binarized-mnist, CIFAR10, CIFAR100, IRIS, MNIST and SVHN2 for

machine learning projects in `hdf5` formats. `Kerosene` datasets depend on the `fuel` library [12] which is a framework based on Python to train neural networks on large datasets.

Table 1: Number of images for train and test sets. In all tables we represent the different datasets using the following scheme: A = CIFAR10, B = Birds, C = Caltech101 and D = SVHN2.

Data set	A	D	C	B
Train	50000	73257	8229	10609
Test	10000	26032	915	1179

4.1 Network Architecture

We used a standard network architecture with an input layer followed by a convolution layer, a ReLU activation function, a convolution layer also followed by a ReLU, maxpooling and a dropout layer. As Figure 1 shows each new task added to the model has two convolution layers, two ReLU activation function layers and the last layer is the activation that corresponds to the softmax function with categorical cross-entropy loss, two dropout layers, a flatten layer and two dense layers one with 512 units and the other one corresponding to the number of classes for each task.

Input images have 32×32 pixels in all cases. The first convolution layer has filters with 32×32 while the other two convolution layers have filters with 64×64. We used the keras API [4] running on tensorflow [1].

Table 2: Network performance on isolated learning.

Train	Test	Baseline [%]	Exec.time[s]
A	A	74.80±0.70	144
D	D	93.13±0.69	204
C	C	64.04±1.00	36
B	B	14.54±1.40	48

4.2 Adding New Tasks to the Model

Table 3 presents the performance of the proposed method when adding new tasks and compares it with the baseline [6].

In the presented results, our method clearly outperforms LwF on all tasks showing that adding layers is a good approach to maintain for learning new tasks

while preserving the performance on previous learned tasks. It is interesting to verify that when we use a model trained on CIFAR10 to train a model on Caltech101 the performance for Caltech101 increases compared to isolated learning with a difference of 3.24% and once again it suggest that using one model in a task can help learn even better the other one.

Results show that by applying our method it is possible to overcome the problem of catastrophic forgetting when new tasks are added and the choice of the branch to use when predicting a specific task is done by the gate model.

Table 3: Gate and two tasks accuracy (and standard deviation) on new tasks for SeNA-CNN and LwF.Execution time for train and test.

Old	New	LwF	Gate	SeNA-CNN	Time[s]
A	D	84.69(0.84)	99.56(0.06)	**88.52**(1.50)	798
A	C	63.84(0.34)	99.83(0.12)	**67.28**(1.19)	228
A	B	7.92(1.22)	99.99(0.04)	**10.48**(1.36)	408
C	A	66.57(2.31)	99.73(0.11)	**69.29**(1.52)	306
C	D	84.78(2.31)	99.81(0.12)	**90.67**(0.78)	480
C	B	7.73(0.66)	99.56(0.10)	**8.54**(1.57)	120
B	A	56.28(1.24)	98.61(2.04)	**57.18**(3.94)	396
B	C	**57.04**(1.15)	97.94(0.55)	56.81(1.39)	132
B	D	78.61(0.54)	95.72(0.08)	**82.76**(1.68)	552
D	A	64.12(0.97)	99.98(0.01)	**70.44**(0.78)	804
D	C	53.46(1.97)	99.85(0.02)	**57.30**(0.55)	588
D	B	7.94(0.54)	99.91(0.04)	**11.36**(1.15)	580

After learning the new task is important to test if there was no catastrophic forgetting for the task previously learned. This is the main goal of our method: guarantee that the network doesn't forget what was previously learned. To demonstrate that our gate model did not forget what was previously learned we tested again on old tasks. Results are shown on table 4

Table 4: Gate and two tasks test accuracy (and standard deviation) on old tasks for SeNA-CNN and LwF. Execution time for train and test.

New	Old	LwF	Gate	SeNA-CNN	Time[s]
A	D	85.41(0.12)	99.56(0.06)	**90.08**(1.63)	798
A	C	59.84(1.16)	99.83(0.12)	**61.12**(0.34)	228
A	B	11.48(0.63)	99.99(0.04)	**13.91**(0.97)	408
C	A	69.45(1.67)	99.73(0.11)	**70.75**(0.11)	306
C	D	89.62(0.93)	99.81(0.12)	**90.27**(0.54)	480
C	B	12.00(0.64)	99.56(0.10)	**13.14**(1.81)	120
B	A	68.22(0.51)	99.61(2.04)	**70.70**(1.60)	396
B	C	57.71(0.72)	97.94(0.55)	**59.45**(1.03)	132
B	D	**91.51**(1.04)	98.93(0.08)	89.12(0.38)	552
D	A	69.36(1.16)	99.98(0.01)	**70.50**(0.62)	804
D	C	**60.80**(0.92)	99.85(0.02)	60.15(0.39)	588
D	B	11.91(0.41)	99.91(0.04)	**13.38**(1.03)	580

4.3 Three Tasks Scenario

To demonstrate that our method is able to deal with different problems, we experiment by learning three different tasks. In this case we used the three datasets previously presented and we combine them two by two for train and one for test. Table 5 presents the results from this scenario and clearly our method is able to maintain the performance and once again it shows that adding new task nodes to an existing neural network is a good option to overcome the catastrophic forgetting problem.

Analysing results in table 5 its clear that our model is effective in learning new tasks by adding the correspondent branches. In this experiment only in two cases LwF outperformed our proposed model with a slight difference of 1.32% in Birds↦SVHN2 and SVHN2↦Caltech101 with 0.50% of difference. In the remaining cases our method outperformed LwF.

Overall this situation also occurs in the other experiments when we compare the performance for two and three tasks scenarios.

Table 6 shows performance after evaluating our method for old tasks when adding two new tasks. Results show that the model did not forget what it learned before.

Table 5: Gate and three tasks test accuracy (and standard deviation) on new tasks for SeNA-CNN and LwF. Execution time for train and test.

Old	New	LwF	Gate	SeNA-CNN	Time[s]
A, D	C	62.90(0.38)	99.14(0.12)	**65.88**(0.78)	696
A, C	B	**6.84**(0.68)	98.07(1.20)	5.53(0.63)	672
A, C	D	84.48(1.16)	99.69(0.19)	**90.76**(0.34)	663
C, D	A	67.72(0.30)	99.52(0.01)	**67.24**(0.65)	660
C, A	D	82.26(0.97)	99.67(0.11)	**89.55**(0.70)	668
C, D	B	5.16(0.63)	98.99(0.19)	**6.45**(1.25)	556
B, C	D	75.86(1.00)	99.37(0.07)	**86.21**(1.10)	648
B, A	C	56.24(0.37)	99.62(0.05)	**60.50**(0.79)	612
B, D	C	57.12(1.17)	98.38(0.01)	**57.43**(0.96)	698
D, A	C	52.86(0.24	99.71(0.07)	**62.34**(0.40)	876
D, C	B	7.47(0.60)	99.63(0.05)	**8.80**(1.28)	828
D, B	A	63.83(0.63)	99.48(0.07)	**71.55**(0.48)	873

Table 6: Gate and three tasks test accuracy (and standard deviation) on old tasks for SeNA-CNN and LwF. Execution time for train and test.

New	Old	LwF	Gate	SeNA-CNN	Time[s]
A	C, D	52.28(0.79), 87.68(0.47)	99.14(0.12)	**56.48**(0.27), **88.30**(0.86)	696
A	B, C	9.93(1.55), 56.29(0.14)	98.07(1.20)	**10.72**(0.94), **58.88**(1.01)	672
A	D, B	87.31(1.14), 5.73(0.84)	99.69(0.19)	**91.38**(0.87), **9.50**(0.64)	663
C	A, D	70.81(0.19), 86.38(0.67)	99.52(0.01)	**70.61**(1.41), **90.33**(0.72)	660
C	B, A	11.73(1.11), 67.89(0.23)	99.67(0.11)	**12.36**(0.84), **68.85**(1.46)	668
C	D, A	86.60(0.93), 68.96(0.57)	98.99(0.19)	**88.00**(0.58), **70.14**(0.83)	556
B	A, C	66.71(1.32), 55.12(1.06)	99.37(0.07)	**71.98**(1.24), **58.78**(1.03)	648
B	C, D	54.72(1.01), 87.54(1.02)	99.62(0.05)	**59.47**(0.96), **91.66**(0.59)	612
B	D, C	85.96(0.89), 56.07(0.69)	98.38(0.01)	**88.47**(0.67), **59.24**(1.02)	698
D	A, C	64.13(0.76), 56.63(0.95)	99.71(0.07)	**70.38**(1.59), **59.73**(0.68)	876
D	C, A	58.03(0.80), 64.99(1.67)	99.63(0.05)	**61.60**(1.46), **70.41**(0.98)	828
D	B, C	7.06(0.70), 56.19(0.87)	99.48(0.07)	**9.13**(0.48), **61.24**(0.83)	873

4.4 Four Tasks Scenario

In this subsection we present results when we add the fourth task. In this experiment we use one task as old and then add sequentially the other three tasks and when learning the new tasks we consider the previous learned tasks as old. Results are shown in the table 7.

Table 8 presents the accuracy of LwF on old. Results in this table are compared with results on table 9 where we present the performance accuracy on old task of our model. Similarly to the others experiments, here results also show that our model outperformed LwF in combination of all tasks.

Table 7: Gate and four tasks test accuracy (and standard deviation) on new tasks for SeNA-CNN and LwF. Execution time for train and test.

Old	New	LwF	Gate	SeNA-CNN	Time[s]
A, C, B,	D	83.76(1.07)	99.53(0.13)	**90.32**(1.03)	954
C, B, D	A	67.39(1.06)	99.69(0.10)	**70.27**(1.02)	936
B, D, A	C	56.48(1.09)	99.70(0.03)	**58.98**(1.01)	948
D, A, C	B	7.96(0.98)	99.61(0.05)	**10.64**(0.90)	967

Table 8: Four tasks test accuracy (and standard deviation) on old tasks for LwF.

New	Old	LwF
A	C, B, D	51.67(0.98), 7.75(1.07), 82.98(1.03)
C	B, D, A	8.49(1.03), 83.76(1.07), 64.76(1.01)
B	D, A, C	86.44(1.02), 64.95(1.07), 51.18(0.83)
D	A, C, B	68.96(0.98), 54.81(0.97), 7.87(1.02)

Table 9: Gate and four tasks test accuracy (and standard deviation) on old tasks for SeNA-CNN. Execution time for train and test.

New	Old	Gate	SeNA-CNN	Time[s]
A	C, B, D	99.53(0.13)	**58.58**(1.03), **10.21**(1.04), **88.00**(1.03)	954
C	B, D, A	99.69(0.10)	**9.37**(1.02), **90.01**(1.04), **70.53**(1.03)	936
B	D, A, C	99.70(0.03)	**91.08**(0.97), **69.23**(0.83), **60.72**(0.98)	948
D	A, C, B	99.61(0.05)	**72.27**(0.93), **59.25**(1.07), **10.27**(1.07)	967

5 Conclusion

In this paper we presented an improvement to our previously proposed method, SeNA-CNN, to avoid the problem of catastrophic forgetting by selective network augmentation, now with automatic task recognition. The proposed method demonstrated to preserve the previous learned tasks without accessing the old task's data, except for updating the gate. It also show that it is feasible to recognize in an automatic manner, which branch should process a given input image, using a gate network. We demonstrated the effectiveness of our method to avoid catastrophic forgetting on image classification tasks by running it on four different datasets and comparing it with the baseline LwF algorithm.

As future work we consider a real application of our method in the field of robotics, for object, location and face recognition problems.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J.,

Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. Acemoglu, D., Cao, D., Acemoglu, D., Cao, D.: Mit and cifar (2010)
3. Ahmed, K., Baig, M.H., Torresani, L.: Network of experts for large-scale image categorization. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 516–532. Springer International Publishing, Cham (2016)
4. Chollet, F., et al.: Keras (2015)
5. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In: 2004 Conference on Computer Vision and Pattern Recognition Workshop. pp. 178–178 (June 2004). https://doi.org/10.1109/CVPR.2004.109
6. Li, Z., Hoiem, D.: Learning without forgetting. CoRR **abs/1606.09282** (2016), http://arxiv.org/abs/1606.09282
7. Liu, B.: Lifelong machine learning: a paradigm for continuous learning. Frontiers of Computer Science pp. 1–3 (9 2016). https://doi.org/10.1007/s11704-016-6903-6
8. Mante, V., Sussillo, D., Shenoy, K.V., Newsome, W.T.: Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature **503**(7474), 78—84 (November 2013). https://doi.org/10.1038/nature12742, http://europepmc.org/articles/PMC4121670
9. Mountcastle, V.B.: Modality and topographic properties of single neurons of cat's somatic sensory cortex. Journal of Neurophysiology **20**(4), 408–434 (1957). https://doi.org/10.1152/jn.1957.20.4.408, https://doi.org/10.1152/jn.1957.20.4.408, pMID: 13439410
10. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
11. Rafegas, I., Vanrell, M., Alexandre, L.A.: Understanding trained cnns by indexing neuron selectivity. CoRR **abs/1702.00382** (2017), http://arxiv.org/abs/1702.00382
12. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR **abs/1506.01497** (2015), http://arxiv.org/abs/1506.01497
13. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. CoRR **abs/1705.08690** (2017), http://arxiv.org/abs/1705.08690
14. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. rep.
15. Zacarias, A., Alexandre, L.: Sena-cnn: Overcoming catastrophic forgetting in convolutional neural networks by selective network augmentation. In: 8th IAPR TC3 Workshop on Artificial Neural Networks in Pattern Recognition. Springer (2018)