Evaluation of Continuous Image Features Learned by ODE Nets

Fabio Carrara(✉), Giuseppe Amato, Fabrizio Falchi, and Claudio Gennaro

Institute of Information Science and Technologies (ISTI), Italian National Research Council (CNR), Via G. Moruzzi 1, 56124 Pisa, Italy
{fabio.carrara,giuseppe.amato,fabrizio.falchi, claudio.gennaro}@isti.cnr.it

Abstract. Deep-learning approaches in data-driven modeling relies on learning a finite number of transformations (and representations) of the data that are structured in a hierarchy and are often instantiated as deep neural networks (and their internal activations). State-of-the-art models for visual data usually implement deep residual learning: the network learns to predict a finite number of discrete updates that are applied to the internal network state to enrich it. Pushing the residual learning idea to the limit, ODE Net—a novel network formulation involving continuously evolving internal representations that gained the best paper award at NeurIPS 2018—has been recently proposed. Differently from traditional neural networks, in this model the dynamics of the internal states are defined by an ordinary differential equation with learnable parameters that defines a continuous transformation of the input representation. These representations can be computed using standard ODE solvers, and their dynamics can be steered to learn the input-output mapping by adjusting the ODE parameters via standard gradient-based optimization. In this work, we investigate the image representation learned in the continuous hidden states of ODE Nets. In particular, we train image classifiers including ODE-defined continuous layers and perform preliminary experiments to assess the quality, in terms of transferability and generality, of the learned image representations and compare them to standard representation extracted from residual networks. Experiments on CIFAR-10 and Tiny-ImageNet-200 datasets show that representations extracted from ODE Nets are more transferable and suggest an improved robustness to overfit.

Keywords: Transfer learning · Image representations · Continuous neural networks · Ordinary differential equations

This work was partially supported by "Automatic Data and documents Analysis to enhance human-based processes" (ADA), CUP CIPE D55F17000290009, and by the AI4EU project, funded by the EC (H2020 - Contract n. 825619). We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Tesla K40 GPU used for this research.

E. Ricci et al. (Eds.): ICIAP 2019, LNCS 11751, pp. 432–442, 2019.
https://doi.org/10.1007/978-3-030-30642-7_39

1 Introduction

The last decade witnessed the renaissance of neural networks and deep differentiable models for multi-level representation learning known as Deep Learning, that highly improved Artificial Intelligence (AI) and Machine Perception with a special emphasis on Computer Vision. The AI renaissance started in 2012 when a deep neural network, built by Hinton's team, won the ImageNet Large Scale Visual Recognition Challenge [18], and from that, the astonishing results obtained by deep-learning approaches for data-driven modeling produced an exponential-growing research activity on this field. Deep Learning methods have been, and still are, the driving force behind this renaissance, and impressive results have been obtained through the adoption of deep learning in tasks such as image classification [14, 18], object detection [26, 27], cross-media retrieval [6], image sentiment analysis [31], recognition [1], etc. Being a representation learning approach, the rationale behind deep-learning methods is to automatically discover a set of multi-level representations from raw data that are specialized for the specific task to be solved, such as object detection or classification [19]. Starting from raw data, each level of representation captures features of the input at increasing level of abstraction that are useful for building successive representations. Following this definition, we understand how relevant representations learned in intermediate layers of deep learning architectures are. In the context of visual data modeling, the architectures of models, mostly based on convolutional neural networks, rapidly evolved from simple feed-forward networks to very deep models with complex interactions between intermediate representations, such as residual [15] or densely connected networks [16].

Recently, in the NeurIPS 2018 best paper [9], Chen et al. proposed *ODE Nets*—a novel model formulation with continuous intermediate representations defined by parametric ordinary differential equations (ODEs). This models can be used as a generic building block for neural modeling: the evolution of the activations and the gradients with respect to parameters can be computed calling a generic ODE solver. This formulation provides several benefits, including natural continuous-time modeling, O(1)-memory cost, adaptive computation, and tunable trade-off between speed and accuracy at inference time. The authors demonstrated ODE blocks in image classifiers trained on the MNIST dataset, actually creating a continuous and evolving activation space of image representations.

In this work, we analyze the continuous feature hierarchy created by ODE Nets when classifying natural images in terms of generality and transferability, and we compare them to representations extracted with standard neural networks. We investigate multiple architectures in which a different amount of processing is delegated to ODE blocks: we analyze standard residual networks, mixed residual-ODE networks, and finally we also consider ODE-only architectures. Preliminary experiments on CIFAR-10 and Tiny-ImageNet-200 datasets show promising results for continuous representations extracted by ODE Nets outperforming similar-sized standard residual networks on a transfer learning benchmark.

2 Related Work

Neural Image Representations. Ever since the recent breakthroughs in the deep learning field, extracting image representations from deep models, specially convolutional neural networks, has led to unprecedented accuracy in many vision tasks. Early studies explored features extracted from generic object classifiers trained on ImageNet: activations of late fully-connected layers played the role of global descriptors and provided a strong baseline as robust image representations [5,29]. With the definition of more complex networks, the attention shifted to feature maps obtained from convolutional layers. Effective representations can be extracted from convolutional feature maps via spatial max-pooling [3,25,30] or sum-pooling [4,17], or more complex aggregation methods [2,21,24]. Better representation can be obtained by fine-tuning the pretrained networks to the retrieval task via siamese [23] or triplet [2,12] learning approaches. To the best of our knowledge, we are the first to investigate ODE-derived continuous image representations.

ODE-inspired Neural Architectures. Most of current state-of-the art models implements some sort of residual learning [14,15], in which each layer or block computes an update to be added to its input to obtain its output instead of directly predict it. Recently, several works showed a strong parallelism between residual networks and discretized ODE solutions, specifically demonstrating that residual networks can be seen as the discretization of the Euler solution [22,33]. This interpretation sprouted novel residual networks architectures inspired by advanced discretizations of differential equations. [22] and [35] derived residual architectures justified by approximating respectively the Linear Multi-step and Runge–Kutta methods. Comparisons with dynamical systems inspired works on reversibility and stability of residual networks [7,8,13,28]. [9] propose to directly adopt ODE solvers to implement continuous dynamics inside neural networks. Traditional variable-step ODE solvers enable sample-wise adaptive computations in a natural way, while previously proposed methods for adaptive computation on classical networks [8,32] require additional parameters to be trained.

3 ODE Nets

In this section, we review the main concepts about ODE Nets, including their formulation and training approach. For a full detailed description, see [9].

An ODE Net is a neural network that include one or more blocks whose internal states are defined by a parametric ordinary differential equation (ODE). Let $\mathbf{z}(t)$ the vector of activations at a specific time t of its evolution. We define its dynamics by a first-order ODE parametrized by θ

$$\frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} = f(\mathbf{z}(t), t, \theta)\,. \tag{1}$$

Given the initial value of the state $\mathbf{z}(t_0)$—the input of the ODE block—we can compute the value of the state at a future time $\mathbf{z}(t_1)$—that we consider the output of the ODE block—via integration of Eq. 1

$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} \frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t}\mathrm{d}t = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)\mathrm{d}t\,. \tag{2}$$

This computation can be efficiently performed by modern ODE solvers, such as the ones belonging to the Runge-Kutta family. Thus, the forward pass of an ODE block is implemented as a call to a generic ODE solver

$$\mathbf{z}(t_1) = \mathrm{ODESolver}(f, \mathbf{z}(t_0), t_0, t_1, \theta)\,, \tag{3}$$

where f can be an arbitrary function parametrized by θ which is implemented as a standard neural network.

In order to be able to train ODE Nets, we need to adjust the parameters θ in order to implement the correct dynamics of the continuous internal state for our specific task. Thus, given a loss function $\mathcal{L}$, we need to compute its gradient with respect to parameters $\mathrm{d}\mathcal{L}/\mathrm{d}\theta$ to perform a gradient descent step. Although we can keep track of all the internal operations of the specific ODE solver used and use backpropagation, this leads to a huge memory overhead, specially when the dynamics of the internal state are complex, and the ODE solver requires many steps to find the solution. Instead, Chen et al. [9] proposed to adopt the *adjoint sensitivity method*. The adjoint state $\mathbf{a}(t)$ is defined as the derivative of the loss with respect to the internal state $\mathbf{z}(t)$

$$\mathbf{a}(t) = \frac{\partial \mathcal{L}}{\partial \mathbf{z}(t)}, \tag{4}$$

and its dynamics can be described by the following ODE

$$\frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t} = -\mathbf{a}(t)\frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}(t)}. \tag{5}$$

The quantity we are interest in—the derivative of the loss with respect to parameters $\mathrm{d}\mathcal{L}/\mathrm{d}\theta$—can be expressed in function of the adjoint $\mathbf{a}(t)$

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} = \int_{t_0}^{t_1} \mathbf{a}(t)\frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta}\mathrm{d}t, \tag{6}$$

where $\partial f(\mathbf{z}(t), t, \theta)/\partial\theta$ is known and defined by the structure of f. To compute $\mathbf{a}(t)$ and thus $\mathrm{d}\mathcal{L}/\mathrm{d}\theta$, we need to know the entire trajectory of $\mathbf{z}(t)$, but this can be recovered starting from the last state $\mathbf{z}(t_1)$ and by solving its ODE (Eq. 1) backward in time. With a clever formulation, Chen et al. [9] also showed that it is possible to combine the process for finding $\mathbf{z}(t)$, $\mathbf{a}(t)$, and $\mathrm{d}\mathcal{L}/\mathrm{d}\theta$ in a unique additional call to the ODE solver.

Among the properties of ODE Nets, noteworthy benefits are (a) *O(1)-memory cost*, since no intermediate activations are needed to be stored for both forward and backward operations, (b) *adaptive computation*, as modern adaptive ODE solvers automatically adjust the step size required to find the solution depending on the complexity of the dynamics induced by a specific input, (c) *inference-time speed-accuracy trade-off tuning*, as the tolerance of adaptive solvers can be lowered at inference time to obtain less accurate solutions faster or viceversa.

4 Tested Architectures

In this section, we describe the architectures of the image classifiers implemented with ODE Nets that we are going to analyze. We test three architectures in total. The first two are the ones defined by Chen et al. [9], i.e. a standard residual network with 8 residual blocks, and a mixed architecture with two residual blocks and an ODE block. In addition, we analyze an architecture defined by the minimum amount of standard layers, that is thus composed by a single convolutional layer and an ODE block. A detailed description of the architectures follows.

Residual Net. We choose a standard residual network (ResNet) as a baseline image classifier with the same architecture chosen by Chen et al. [9]. Starting from the input, the ResNet is composed by two residual blocks each with a downsample factor of 2, and then by six additional residual blocks. The output of the last residual block is average-pooled and followed by a fully-connected layer with softmax activation that produces the final classification. The formulation of the residual block is the standard one proposed in [15], but the batch normalization operation is replaced with group normalization [34]. Thus, the structure of the residual block is composed by two 3×3 256-filters convolutions preceded by a 32-group normalization and ReLU activation, and a last group normalization: GroupNorm-ReLU-Conv-GroupNorm-ReLU-Conv-GroupNorm. For the first two blocks, we used 64-filters convolutions, and we employ 1×1 convolutions with stride 2 in the shortcut connections to downsample its input.

Res-ODE Net. The first ODE-defined architecture tested is the one proposed by Chen et al. [9]. They proposed to keep the first part of the architecture as the previously described ResNet and substitute the last six residual blocks by an ODE block that evolves a continuous state $\mathbf{z}(t)$ in a normalized time interval $[0, 1]$. The ODE function f defining its dynamics is implemented using the same network used in the residual blocks. In addition, this module takes the value of the current time t as input to convolutional layers as a constant feature maps concatenated to the other input maps. Similarly to ResNets, the output of the ODE block $\mathbf{z}(1)$ is average-pooled and fed to a fully-connected layer with softmax activation.

ODE-only Net. To fully exploit the ODE block and analyze its internal evolution, we explore an additional architecture only composed by a single convolutional layer and an ODE block. The convolutional layer has 256 4×4 filters slided with stride 2 which is not followed by any non-linear activation. The ODE block, defined as in the Res-ODE architecture, takes the output of the convolution as the initial state of the ODE block $\mathbf{z}(0)$. As in the other architectures, the final state $\mathbf{z}(1)$ is taken as output and fed to the classification layer.

5 Experimental Evaluation

Following [29], we evaluate the effectiveness and generality of learned image representation by measuring its effectiveness in a transfer learning scenario [11]. We learn features extractors for a particular image classification task (source), and we evaluate them by using the learned representations as high-level features for another image classification task with similar domain (target).

For our investigation, we used two low-resolution datasets, that is CIFAR-10 for the source task, and Tiny-ImageNet-200 for the target task. CIFAR-10 [20] is a small-resolution 10-class image classification datasets with 50k training images and 10k test images. Tiny-ImageNet-200[1] is a 200-class classification dataset with 64×64 images extracted from the famous ImageNet subset used for the ILSVRC challenge. Each class has 500 training images, 50 validation images, and 50 test images, for a total of 100k, 10k, and 10k images respectively for training, validation, and test sets.

We train all the models (Residual Net, Res-ODE Net, ODE-only Net) for 200 epochs on the CIFAR-10 dataset, adopting the SGD optimizer with momentum of 0.9, a batch size of 128, a learning rate of 0.1 decreased by a factor 10 when the loss plateaus, and a L2 weight decay of 10^{-4}. We employ commonly used data augmentation techniques for CIFAR-10, that is random cropping, color jittering, and horizontal flipping, and we apply dropout with a .5 drop probability on the layer preceeding the classifier. As ODE solver in ODE Nets, we employ a GPU implementation[2] of the adaptive-step fourth order Runge-Kutta method [10], that performs six function evaluation per step plus the initial and final timestep evalution, i.e. number of function evaluation $= 6 \times \text{steps} + 2$.

Table 1 reports for each model the best test classification error obtained and the complexity in both terms of number of parameters and ODE solver steps. The introduction of ODE blocks in the image classification pipeline drastically reduces the number of parameters of the model but also introduced a slight performance degradation of the overall classification performance. Also note that for ODE Nets, the number of steps required by the ODE solver to compute a forward pass of the network depends on the complexity of the dynamics of internal state induced by a specific input. For Res-ODE models, the ODE solver requires 3 to 4 steps to process an image, indicating that the learned dynamics of hidden state are quite simple, and most of the information extraction process is due to preceding standard layers. On the other hand, in ODE-only networks the ODE block is responsible to model the entire feature extraction process and

[1] https://tiny-imagenet.herokuapp.com/.
[2] https://github.com/rtqichen/torchdiffeq.

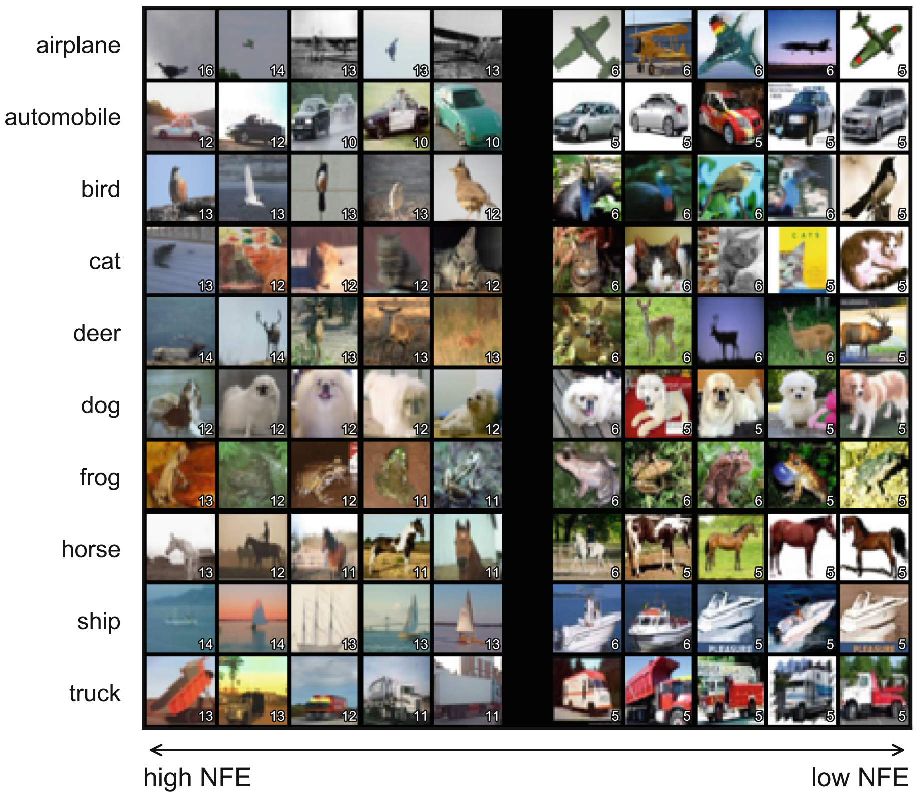

Fig. 1. The most (left) and least (right) demanding images of CIFAR-10 test set in terms of the number of solver steps required by the ODE solver (that is reported near each image).

Table 1. Classification performance on CIFAR-10.

	Test error	Params	Solver steps
Residual Net	7.28%	7.92M	-
Res-ODE Net	7.80%	2.02M	3.8 ± 0.4
ODE-only Net	9.17%	1.20M	7.8 ± 1.5

thus requires to learn more complex dynamics of the hidden state; as a consequence, the mean number of solver step required is higher, but it is more variable depending on the input image. Figure 1 show the top-5 and bottom-5 images of the CIFAR-10 test set in terms of number of solver steps required to make a prediction; we can notice that the more prototypical and easily recognizable images require fewer steps, while additional processing is adaptively employed by the ODE solver when more challenging images are presented.

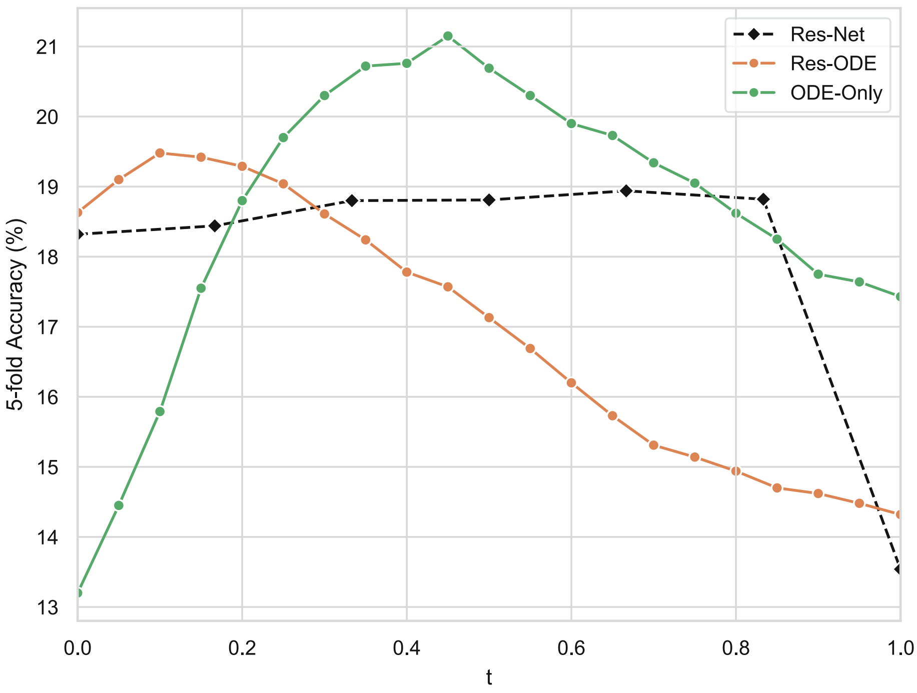

Fig. 2. Accuracy (%) on the Tiny-ImageNet-200 validation set of a linear SVM trained on $\mathbf{z}(t)$. Results obtained using the 7 intermediate layers of the Residual Net are evenly placed between 0 and 1 on the x-axis.

We extract intermediate activations from all the trained models as image representations for the target task (Tiny-ImageNet-200). For Residual Nets, we test the output of the last 7 residual modules before the classifier. For both ODE Nets, there are an infinite amount of intermediate states $\mathbf{z}(t), t \in [0, 1]$ that we can extract; we sample $z(t)$ between 0 and 1 with a sample rate of 0.05 and test every sample as image representation for the target task. For all the extracted representations, we apply global average pooling to obtain a spatial-agnostic feature vector.

We train a linear SVM classifier that rely on the extracted features on the validation set of Tiny-ImageNet-200 (for which labels are provided): we perform a grid search of the penalty parameter $C \in \{0.01, 0.1, 1, 10, 100\}$, keeping track of the configuration that obtained the best 5-fold cross-validated accuracy. We then retrain this configuration on the whole set and report its accuracy. In Fig. 2, we report the accuracies obtained by all the SVMs trained on different internal activations of all the tested models. The x-axis indicate the time stamp t used to extract the internal representation of ODE Nets $\mathbf{z}(t)$, while the y-axis indicate the obtained accuracy. For convenience, we place the 7 points obtained from the 7 intermediate layers of the Residual Net evenly spaced in the x-axis between 0 and 1.

In both ODE Nets, we observe a concave trend of the accuracy when using later activations, with a maximum accuracy obtained using intermediate features extracted from the early or mid evolution of the continuous hidden states ($\sim$21% at t = .45 for ODE-only and $\sim$19.5% at t = .1 for Res-ODE). As already suggested by findings in other works [3,5], mid-features seem to be more transferable. Mid-features in Res-ODE are already extracted by preceding standard layers, thus they occur early in the evolution of the continuous hidden state. ODE Nets provide a more general and transferable image representation with respect to Residual Nets that instead provide a lower and practically constant performance on the target task, suggesting a higher degree of overfit to the source task.

Notwithstanding that, the CIFAR-10 dataset is not able to provide enough information about all the classes of the target dataset to obtain competitive accuracies, and a larger and more complex dataset should be used as a source task. Unfortunately, training ODE Nets has currently a high computational cost, as also suggested by the evaluation of their proposers that was limited to the MNIST dataset for image classification. This limits our ability to perform a larger-scale experimentation, that are left for future work.

6 Conclusions

In this paper, we investigated the representations learned by ODE Nets, a promising and potentially revolutionary deep-learning approach in which hidden states are defined by an ordinary differential equation with learnable parameters. We conducted our experiments in a transfer learning scenario: we trained three deep-learning architectures (ODE-only Net, Res-ODE Net and Residual Net) on a particular image classification task (CIFAR-10), and we evaluate them by using the learned representations as high-level features for another image classification task (Tiny-ImageNet-200). The results show that ODE Nets provide a more transferable, and thus more general, image representation with respect to standard residual networks. Considering also other intrinsic advantages of ODE Nets, such as O(1)-memory cost, and adaptive and adjustable inference-time computational cost, this preliminary analysis justifies and encourages additional research on the optimization of this kind of networks and its adoption in image representation learning.

References

1. Amato, G., Falchi, F., Vadicamo, L.: Visual recognition of ancient inscriptions using convolutional neural network and fisher vector. J. Comput. Cult. Heritage (JOCCH) **9**(4), 21 (2016)
2. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5297–5307 (2016)

3. Azizpour, H., Sharif Razavian, A., Sullivan, J., Maki, A., Carlsson, S.: From generic to specific deep representations for visual recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 36–45 (2015)
4. Babenko, A., Lempitsky, V.: Aggregating local deep features for image retrieval. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1269–1277 (2015)
5. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 584–599. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_38
6. Carrara, F., Esuli, A., Fagni, T., Falchi, F., Moreo Fernández, A.: Picture it in your mind: generating high level visual representations from textual descriptions. Inform. Retrieval J. **21**(2), 208–229 (2018). https://doi.org/10.1007/s10791-017-9318-6
7. Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., Holtham, E.: Reversible architectures for arbitrarily deep residual neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
8. Chang, B., Meng, L., Haber, E., Tung, F., Begert, D.: Multi-level residual networks from dynamical systems view. In: International Conference on Learning Representations (2018). https://openreview.net/forum?id=SyJS-OgR-
9. Chen, T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. In: Advances in Neural Information Processing Systems, pp. 6572–6583 (2018)
10. Dormand, J.R., Prince, P.J.: A family of embedded Runge-Kutta formulae. J. Comput. Appl. Math. **6**(1), 19–26 (1980)
11. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
12. Gordo, A., Almazan, J., Revaud, J., Larlus, D.: End-to-end learning of deep visual representations for image retrieval. Int. J. Comput. Vis. **124**(2), 237–254 (2017)
13. Haber, E., Ruthotto, L.: Stable architectures for deep neural networks. Inverse Probl. **34**(1), 014004 (2017)
14. 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)
15. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
16. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
17. Kalantidis, Y., Mellina, C., Osindero, S.: Cross-dimensional weighting for aggregated deep convolutional features. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9913, pp. 685–701. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46604-0_48
18. 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)
19. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
20. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)

21. Li, Y., Xu, Y., Wang, J., Miao, Z., Zhang, Y.: MS-RMAC: multiscale regional maximum activation of convolutions for image retrieval. IEEE Signal Process. Lett. **24**(5), 609–613 (2017)
22. Lu, Y., Zhong, A., Li, Q., Dong, B.: Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. arXiv preprint arXiv:1710.10121 (2017)
23. Radenović, F., Tolias, G., Chum, O.: CNN image retrieval learns from BoW: unsupervised fine-tuning with hard examples. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 3–20. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_1
24. Radenović, F., Tolias, G., Chum, O.: Fine-tuning cnn image retrieval with no human annotation. IEEE Trans. Pattern Anal. Mach. Intell. **41**, 1655–1668 (2018)
25. Razavian, A.S., Sullivan, J., Carlsson, S., Maki, A.: Visual instance retrieval with deep convolutional networks. ITE Trans. Media Technol. Appl. **4**(3), 251–258 (2016)
26. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition, pp. 779–788 (2016)
27. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
28. Ruthotto, L., Haber, E.: Deep neural networks motivated by partial differential equations. arXiv preprint arXiv:1804.04272 (2018)
29. Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 806–813 (2014)
30. Tolias, G., Sicre, R., Jégou, H.: Particular object retrieval with integral max-pooling of CNN activations. arXiv preprint arXiv:1511.05879 (2015)
31. Vadicamo, L., et al.: Cross-media learning for image sentiment analysis in the wild. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 308–317 (Oct 2017). https://doi.org/10.1109/ICCVW.2017.45
32. Veit, A., Belongie, S.: Convolutional networks with adaptive inference graphs. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–18 (2018)
33. Weinan, E.: A proposal on machine learning via dynamical systems. Commun. Math. Stat. **5**(1), 1–11 (2017)
34. Wu, Y., He, K.: Group normalization. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–19 (2018)
35. Zhu, M., Chang, B., Fu, C.: Convolutional neural networks combined with Runge-Kutta methods. arXiv preprint arXiv:1802.08831 (2018)