Continuous ODE-defined Image Features for Adaptive Retrieval

Fabio Carrara
fabio.carrara@isti.cnr.it
ISTI CNR
Pisa, Italy

Giuseppe Amato
giuseppe.amato@isti.cnr.it
ISTI CNR
Pisa, Italy

Fabrizio Falchi
fabrizio.falchi@isti.cnr.it
ISTI CNR
Pisa, Italy

Claudio Gennaro
claudio.gennaro@isti.cnr.it
ISTI CNR
Pisa, Italy

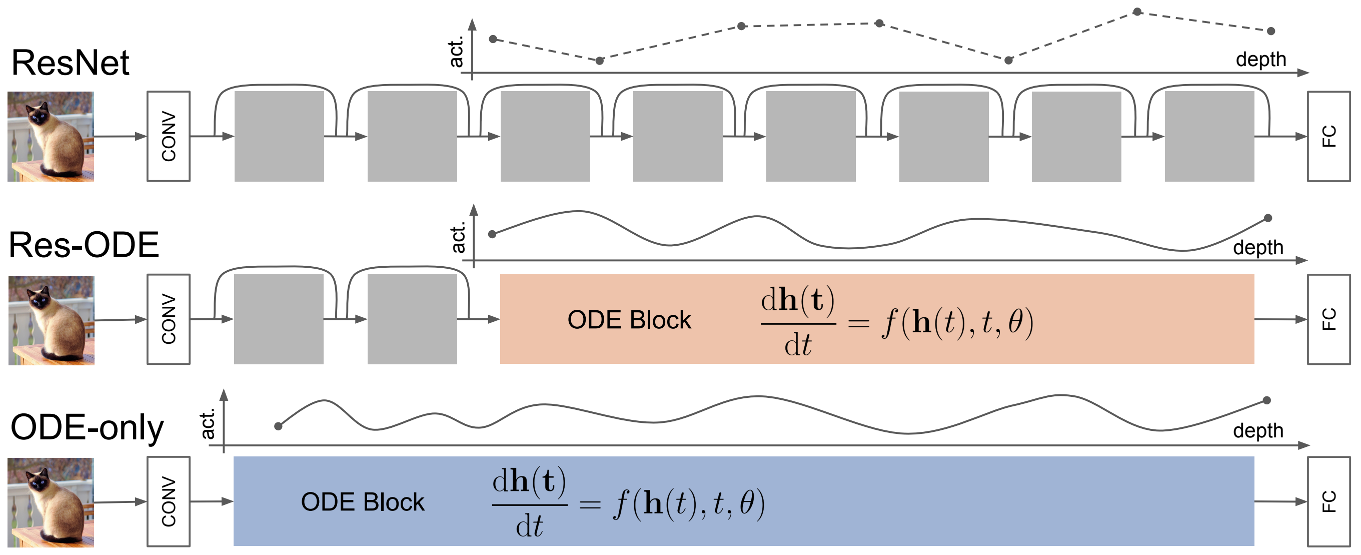

ResNet
act.
depth
CONV
FC
Res-ODE
ODE Block
$\frac{\mathrm{d}\mathbf{h}(\mathbf{t})}{\mathrm{d}t} = f(\mathbf{h}(t), t, \theta)$
ODE-only

Figure 1: Evolution of activations (act.) in standard neural networks (first row, ResNets) and of continuous ODE-defined neural networks (second row, the Res-ODE network Chen et al. [9], and, third row, the ODE-only net [6]) through depth. Gray boxes indicate residual blocks. The plots illustrate a possible evolution of a specific activation.

ABSTRACT

In the last years, content-based image retrieval largely benefited from representation extracted from deeper and more complex convolutional neural networks, which became more effective but also more computationally demanding. Despite existing hardware acceleration, query processing times may be easily saturated by deep feature extraction in high-throughput or real-time embedded scenarios, and usually, a trade-off between efficiency and effectiveness has to be accepted. In this work, we experiment with the recently proposed continuous neural networks defined by parametric ordinary differential equations, dubbed ODE-Nets, for adaptive extraction of image representations. Given the continuous evolution of the network hidden state, we propose to approximate the exact feature extraction by taking a previous "near-in-time" hidden state as features with a reduced computational cost. To understand the potential and the limits of this approach, we also evaluate an ODE-only architecture in which we minimize the number of classical layers in order to delegate most of the representation learning process — and thus the feature extraction process — to the continuous part of the model. Preliminary experiments on standard benchmarks show that we are able to dynamically control the trade-off between efficiency and effectiveness of feature extraction at inference-time by controlling the evolution of the continuous hidden state. Although ODE-only networks provide the best fine-grained control on the effectiveness-efficiency trade-off, we observed that mixed architectures perform better or comparably to standard residual nets in both the image classification and retrieval setups while using fewer parameters and retaining the controllability of the trade-off.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICMR '20, June 8–11, 2020, Dublin, Ireland
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7087-5/20/06…$15.00
https://doi.org/10.1145/3372278.3390690

CCS CONCEPTS

• **Computing methodologies** → **Image representations**; **Neural networks**; • **Mathematics of computing** → *Ordinary differential equations.*

KEYWORDS

image retrieval; feature extraction; continuous neural networks; ordinary differential equations; adaptive computation

ACM Reference Format:
Fabio Carrara, Giuseppe Amato, Fabrizio Falchi, and Claudio Gennaro. 2020. Continuous ODE-defined Image Features for Adaptive Retrieval. In *Proceedings of the 2020 International Conference on Multimedia Retrieval (ICMR*

'20), June 8–11, 2020, Dublin, Ireland. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3372278.3390690

1 INTRODUCTION

Deep neural networks and data-driven deep-learned features currently represent the state of the art for creating effective representations of perceptual data, such as images. The last years witnessed an increasing research effort on building more and more performing representations for multimedia data, that generally lead to the design of deeper and more complex models [16, 18, 36]. In the context of content-based image-retrieval and transfer learning in general, desirable properties of image representations include effectiveness — i.e. the ability to represent complex semantics conveyed by the image in a convenient vector form — and computational efficiency. With the advancement of deep neural networks, the former has been successfully increased, closing the so-called semantic gap between data and its representation, while for the latter, an increasing computational horsepower is required as networks get deeper.

Despite hardware acceleration, the significant computational budget can be problematic in high-throughput or real-time image retrieval scenarios, such as web-scale image similarity search or real-time face verification, in which the query processing time can easily dominate the system efficiency, e.g. as power or time consumption concerns. In most of the cases, a trade-off has to be accepted by decreasing processing times while accepting some performance degradation: depending on the available time budget, we could resort to less performing but simpler models and optionally perform additional computations, e.g. query augmentation or expansion.

In a NeurIPS 2018 best paper, Chen et al. [9] proposed a neural network formulation with continuously evolving hidden states that may provide a more natural way to control this trade-off in dynamic load conditions. In this novel network architecture, dubbed ODE-Net, the evolution of a hidden state is defined by a parametric ordinary differential equation (ODE) that can be numerically solved by invoking an ODE solver (see Figure 1). Chen et al. [9] also showed that we can approximate the gradients with respect to the input and the parameters of the ODE-governed hidden state with constant memory cost using the adjoint method, making possible to train the hidden dynamics for complex tasks such as image classification. Using an adaptive ODE solver, the computation needed to find the final hidden state is automatically adjusted depending on the complexity of the dynamics induced by the specific input.

In this work, we investigate the use ODE-Nets to extract image representation (to the best of our knowledge we are the first) and propose a novel approach to dynamically controlling the effectiveness-efficiency trade-off at inference time. Thanks to feature continuity, we show that by early stopping the evolution of the hidden state, we can approximate extracted features obtaining a lower computational cost while expecting only a marginal loss in accuracy. We analyze the image representation extracted and evaluate them in an image retrieval scenario. Moreover, we evaluate an ODE-dominated architecture, dubbed ODE-Only Net, in which most of the image processing, from low-level to high-level feature extraction, is defined by a single ODE. We show that ODE-Nets achieve a performance comparable with standard residual networks while offering a more fine-grained control on the effectiveness-efficiency trade-off[1].

The rest of the paper is organized as follows. Section 2 provides background on ordinary differential equations, residual network and ODE-Nets. Section 3 reviews work related to ODE-inspired neural networks and image retrieval based on neural image representations. In Section 4, we propose our approach to learn and extract ODE-defined image representations representations, and in Section 5, we describe the experimental evaluation and discuss results. Section 6 concludes the paper.

2 BACKGROUND

2.1 Ordinary Differential Equations

Given an independent variable t and a function $\mathbf{y}(t) = \mathbf{y}$, an ordinary differential equation (ODE) of order n is an equation including a function F of t, $\mathbf{y}$, and its derivatives up to the n-th one

$$F\left(t, \mathbf{y}, \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t}, \frac{\mathrm{d}^2\mathbf{y}}{\mathrm{d}t^2}, \dots, \frac{\mathrm{d}^n\mathbf{y}}{\mathrm{d}t^n}\right) = 0\,. \tag{1}$$

In the case of explicit ODEs in which the highest-order derivative of $\mathbf{y}$ can be isolated, an n-order equation can be reduced to a system of n first-order ODEs that can be written in vector format as

$$\begin{aligned} \frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} &= f(\mathbf{h}(t), t)\,, \quad \text{with} \\ \mathbf{h}(t) &= \left[\mathbf{y}, \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t}, \dots, \frac{\mathrm{d}^{n-1}\mathbf{y}}{\mathrm{d}t^{n-1}}\right]\,, \end{aligned} \tag{2}$$

where t is the independent variable (time), $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is a vector-valued function defining the dynamics, and $\mathbf{h}(t)$ is the vector solution to the equation.

Given the initial-value problem

$$\begin{cases} \dfrac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} & = f(\mathbf{h}(t), t) \\ \mathbf{h}(t_0) & = \mathbf{h}_{t_0} \end{cases}\,, \tag{3}$$

we can find the solution at time t_1 by integration

$$\mathbf{h}(t_1) = \mathbf{h}(t_0) + \int_{t_0}^{t_1} \frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t}\,\mathrm{d}t = \mathbf{h}(t_0) + \int_{t_0}^{t_1} f(\mathbf{h}(t), t)\,\mathrm{d}t\,. \tag{4}$$

The simplest algorithm to numerically solve ODEs is the Euler integrator

$$\mathbf{h}(t + \Delta t) = \mathbf{h}(t) + \Delta t f(\mathbf{h}(t), t)\,, \tag{5}$$

which is derived by taking a rectangular approximation of the integral in Equation (4) for small time steps Δt. More complex classical ODE solvers include the family of Runge-Kutta and linear multi-step methods that provide a more accurate solution for equivalent step sizes Δt and additional features, e.g. adaptive step size.

2.2 Residual Nets and ODE-Nets

A residual network [16] (ResNet in brief) is a deep neural network in which each t-th layer computes a displacement to be added to its input $\mathbf{h}_t$ to produce its output $\mathbf{h}_{t+1}$

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)\,, \tag{6}$$

[1]The code for reproducing the experiments is available at https://github.com/fabiocarrara/neural-ode-features.

where f is the function parametrized by θ_t which produces the update, e.g. usually any combination of convolutional, normalization, and activation layers in the image processing context. Comparing Equation (6) to (5), residual networks resemble the discretization of the solution of an ODE found with the Euler method for $\Delta t = 1$.

Pushing this concept to the limit, a residual network with an infinite amount of layers each producing an infinitesimal update can be formulated as the solution of an ODE defining the dynamics of its continuous hidden state $\mathbf{h}(t)$

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = f(\mathbf{h}(t), t, \theta) \,. \tag{7}$$

Thus, given an initial state $\mathbf{h}(t_0)$ and a fixed value of parameters θ, we can compute the value of the hidden state at a given time $\mathbf{h}(t_1)$ by applying a generic ODE solver

$$\mathbf{h}(t_1) = \mathrm{ODESolver}(\mathbf{h}(t_0), f, t_0, t_1, \theta) \,. \tag{8}$$

We refer to this computation as an *ODE block* and to networks including this type of computational block as *ODE-Nets*. The depth of this kind of networks can be thought as the number of steps the ODESolver takes in order to find $\mathbf{h}(t_1)$.

Training ODE-Nets requires to compute the gradient $\frac{\partial L}{\partial \theta}$ of a loss function L with respect to the parameters θ defining the dynamics of $\mathbf{h}(t)$. Although this operation can be done by backpropagating through the internals of the specific ODE solver, keeping track of the internal computations carried out and their respective gradients can lead to a prohibitive high usage of memory, especially when finding the ODE solution requires many iterations. To overcome this limitation, Chen et al. [9] proposed instead to adopt the *adjoint state method* to compute the gradient of the loss with respect to the hidden state at each time stamp, also called the adjoint $a(t) = \frac{\partial L}{\partial \mathbf{h}(t)}$, and subsequently $\frac{\partial L}{\partial \theta}$. The adjoint $a(t)$ is defined as the solution to another ODE starting from the final state $\mathbf{h}(t_1)$ and evolving backward in time until $\mathbf{h}(t_0)$, and by carefully defining the state vector, we can compute all the integrals needed for solving $a(t)$, and most importantly $\frac{\partial L}{\partial \theta}$, in a single additional call to the ODE solver, resulting in a $O(1)$ memory consumption (for more details, see [9]).

ODE-Nets can thus be adopted as building blocks for data modeling with several benefits, such as:

- multiple-layers residual networks can be substituted with a unique ODE block with $O(1)$ memory cost;
- fewer parameters are needed since the parameters θ of the function define the dynamics of the hidden state throughout all the time interval, and thus similar weights for near-time evaluations are reused;
- using adaptive ODE solvers, the number of function evaluations, and thus the computational cost, is adaptively chosen depending on the complexity of the hidden state dynamics; this may depend from the specific input instance or from the complexity of the problem to be solved (for harder tasks, the dynamics need to be more complex, and thus more steps are required to solve the ODE); moreover, the tolerance parameter of adaptive ODE solver can be adjusted at inference time to further tune the trade-off between the accuracy of the solution and number of steps required to find it;
- continuous-time sequence modeling is more natural, and there is no need to discretize the input.

Current drawbacks in the adoption of ODE-Nets are

- the increasing computational cost as the training phase proceeds, as more complex dynamics require the ODE solver to perform smaller iterations to find the solution;
- the fixed dimensionality of continuous hidden state which is not permitted to change during its evolution, in contrast with standard neural nets in which downsample operations can be added to reduce the amount of computation; downsample operations in ODE-Nets need to be placed between different ODE blocks increasing the number of calls to the solver;

3 RELATED WORK

Image Representations. In the last years, most of the literature on image representations deals with descriptors extracted from deep architectures, specifically convolutional neural networks. Descriptors extracted from image classification networks trained with large-scale and publicly available datasets (e.g. ImageNet, OpenImages, Visual Genome) proved to be good representations of image contents. Researchers adopted the output of fully-connected layers preceding the classification layer in a CNN as global image descriptors [4, 34], and more recently, the attention shifted on descriptors extracted from the outputs of intermediate convolutional layers via aggregation [20, 31, 35]. For content-based image retrieval, recent work shifted from transfer learning to adopting ad-hoc architectures and learning-to-rank techniques to train more effective image representation networks [3, 13, 27, 30]. Concerning the efficiency and the scalability of image retrieval systems, existing works often instead operate after the image representation has been extracted adopting specialized indexing techniques [1, 2, 11, 26]. For a detailed comparison of state-of-the-art methods, we refer the reader to [29].

ODE-inspired Architectures. In Section 2.2, we briefly showed the parallelism between residual networks [17] and the simple forward Euler discretization of ODE. On the same line, other work proposed architectures with novel residual blocks that can be seen as more complex discretization of differential equation solutions, such as backward/implicit [38], higher-order [22, 39], multi-step [25, 33] or multi-grid [8] methods. Exploiting this parallelism with dynamical systems, several works also proposed residual architectures with useful properties, such as reversibility [7] and stability [15] of the computational graph.

ODE-based Architectures. Differently from ODE-inspired architectures, Chen et al. [9] proposed to incorporate ODEs into differentiable networks by using ODE solvers as a computational blocks inside the architecture. Using adaptive ODE solvers and the adjoint sensitivity method, this novel methodology brings several useful properties, such as O(1)-memory cost, sample-wise adaptive computations, and continuous modeling. This properties paved the way for novel methodologies for multiple tasks, such as irregular time-series modelling [32] and continuous normalizing flows [14], which include continuous feature extraction modelling for visual tasks. In addition to Chen et al. [9], Dupont et al. [12] evaluated their augmented ODE-based architecture on image classification

ResNet	RES-ODE [9]	ODE-only [6]
$3 \times 3, 64$ $\left[\begin{matrix} 3 \times 3 \,/\, 2,\ 64 \\ 3 \times 3,\ 64 \end{matrix}\right]$ $\left[\begin{matrix} 3 \times 3 \,/\, 2,\ K \\ 3 \times 3,\ K \end{matrix}\right]$	$3 \times 3, 64$ $\left[\begin{matrix} 3 \times 3 \,/\, 2,\ 64 \\ 3 \times 3,\ 64 \end{matrix}\right]$ $\left[\begin{matrix} 3 \times 3 \,/\, 2,\ K \\ 3 \times 3,\ K \end{matrix}\right]$	$4 \times 4 \,/\, 2,\ K$
$\left[\begin{matrix} 3 \times 3,\ K \\ 3 \times 3,\ K \end{matrix}\right] \times 6$	$\left\{\begin{matrix} 3 \times 3,\ K \\ 3 \times 3,\ K \end{matrix}\right\}$	$\left\{\begin{matrix} 3 \times 3,\ K \\ 3 \times 3,\ K \end{matrix}\right\}$
global average pooling, 10-d fc, softmax		

Table 1: Tested architectures for image classifiers. Convolutional layers are written in the format *kernel width* $\times$ *kernel height [/ stride], n. filters*; padding is always set to 1. Squared and curly brackets respectively indicate residual and ODE blocks. For MNIST, $K = 64$, and for CIFAR-10, $K = 256$.

	MNIST		CIFAR-10	
	Test Error	# Params	Test Error	# Params
ResNet [9]	0.41%	0.60M	-	-
Res-ODE [9]	0.42%	0.22M	-	-
ResNet [6]	0.37%	0.60M	7.28%	7.92M
Res-ODE [6]	0.47%	0.22M	7.80%	2.02M
ODE-only [6]	0.53%	0.08M	9.17%	1.20M

Table 2: Classification performance of tested architectures on MNIST and CIFAR-10 benchmarks.

benchmarks, while Carrara et al. [6] also tested how representations extracted from ODE-defined models transfer between similar image classification tasks.

Our work follows this last line of research, but we focus on extracting image representation from ODE-based networks for image retrieval and on exploiting continuity and adaptive computation for tuning the efficiency-effectiveness trade-off at query time.

4 LEARNING IMAGE REPRESENTATIONS WITH ODE-NETS

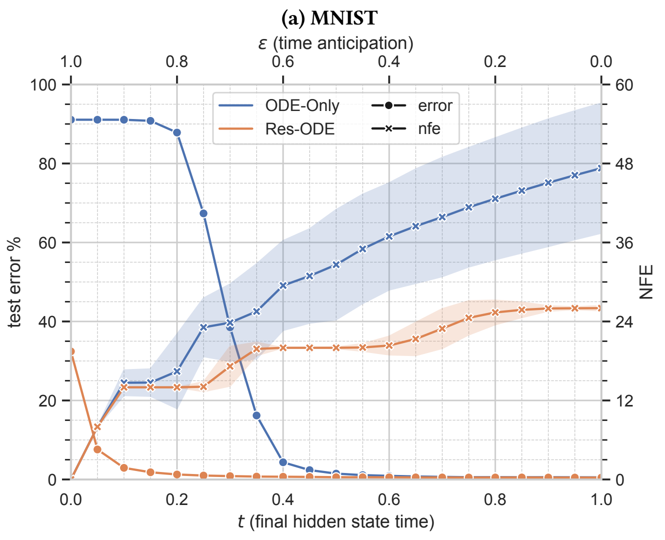
In this section, we describe the ODE-based models we used to extract image representations. Although other existing approaches may be preferable to learn better features for image retrieval (e.g. learning to rank approaches on specialized architectures [13]), in this preliminary work we resort to extracting and testing intermediate features of trained image classifiers to showcase the benefits offered by ODE-Nets in this context (as similarly done by Sharif Razavian et al. [34]), and we leave more complex approaches to future work. Thus, we tackle the image classification task as a proxy to learn good images representations for retrieval. We consider three architectures for extracting features, that is a) a standard residual network (*ResNet*) considered as the baseline, b) a mixed ODE-residual architecture (*Res-ODE*) proposed by Chen et al. [9], and c) an *ODE-only* network [6] in which most of the computation is performed by a single ODE block. The next sections will describe in more details such architectures.

4.1 *Res-ODE*: a Mixed Architecture

In [9], the authors compared ODE-Nets and ResNets on supervised image classification and showed that ODE-Nets are able to reach a comparable test error with ~3x less parameters. In their formulation for the MNIST data, they adopted two residual blocks as a common initial architecture for both models to downsample the input image size by a factor 4 while extracting low-level features. Then for the ResNet, they continued the processing pipeline with an additional 6-block residual network, while for ODE-Nets, an ODE block is adopted. The continuous hidden state $\mathbf{h}(t)$ is evolved by the ODE block in a normalized time interval $[0, 1]$, where $\mathbf{h}(0)$ represents the ODE initial state, and $\mathbf{h}(1)$ is taken as output. Both architectures adopt the standard residual block definition [17] in the residual path or in the ODE function f, with the only exception that Group Normalization [37] is adopted instead of Batch Normalization: the sequence of layers is GN-ReLU-Conv-GN-ReLU-Conv-GN, where GN stands for Group Normalization [37] with 32 groups, ReLU is the Rectified Linear Unit, and Conv is a 3×3 64-filter convolution with padding 1 and stride 1. The time input t to the function f defining the ODE is concatenated as a constant feature map to the input of both convolutional layers. Global average pooling is applied to the final feature map, and the final classifier is implemented with a 10-neuron fully connected layer followed by the softmax activation.

4.1.1 ODE Block Contribution. Aiming at extracting image representations, we first analyze the contribution to representation modeling brought by the ODE block in the Res-ODE architecture by examining the evolution of its continuous hidden state $\mathbf{h}(t)$. Given its continuity, replacing the output of the ODE block $\mathbf{h}(1)$ with $\mathbf{h}(1 - \varepsilon)$ for small $\varepsilon > 0$ is expected to produce negligible perturbations in the following computations. Figure 2 shows the test error obtained by the ODE-Net for increasing values of ε on both MNIST and CIFAR-10 datasets, together with the mean number of function evaluation (NFE) required by the ODE solver to compute $\mathbf{h}(1 - \varepsilon)$. We can notice that the test error slowly increases as ε increases, and in the extreme case of $\varepsilon = 1$ in which the ODE block is completely ignored, the test error rate of Res-ODE has a maximum increase of roughly 30% and 40%, respectively, for MNIST and CIFAR-10. Despite the significant performance drop, this suggests that a fair amount of the feature extraction workload in the Res-ODE architecture is performed by the previous classical layers in the network. This is also supported by the mean NFE, which does not vary significantly with most values of ε, suggesting that no additional iterations are needed to compute forward-in-time hidden states that instead are usually required for more complex dynamics. Moreover, the limited variability of NFE per input sample (i.e. a small standard deviation) suggests that this architecture does not fully exploit the instance-level adaptive computation property of ODE blocks.

On this basis, we explore also another ODE-based architecture, named *ODE-only* [6], in which the amount of classical convolutional layers is minimized in order to leave most of the representation learning process — from low-level to high-level feature extraction — to the dynamics of the continuous hidden state $\mathbf{h}(t)$.



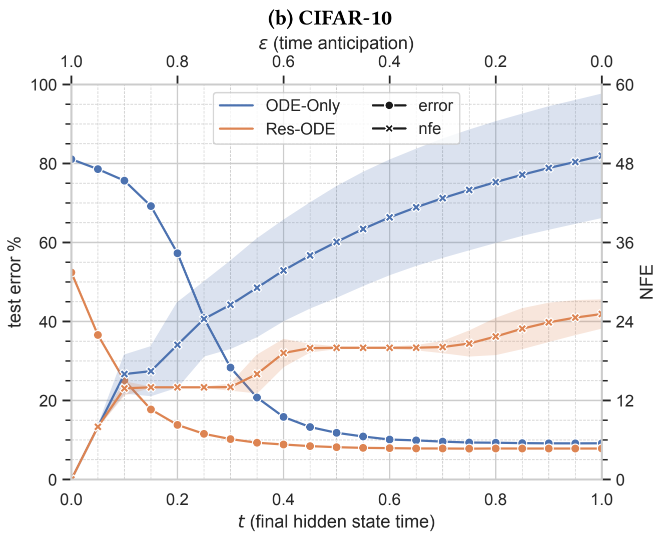


Figure 2: The test error (circles) and the mean number of function evaluations (NFE, crosses) of the ODE-Net and Full-ODE-Net image classifiers when the output of the ODE block $\mathrm{h}(1)$ is replaced by $\mathrm{h}(1-\varepsilon)$, for increasing values of ε. Shaded areas indicate intervals between one standard deviation from the mean.

4.2 *ODE-only* Architecture

The computational graph of the *ODE-only* image classifier is described as follows. First, a single convolution layer with K filters, parametrized by θ_c, is applied to the input image I to a produce a K-channel tensor $\mathbf{h}(0)$

$$\mathbf{h}(0) = \mathrm{conv}_K(I, \theta_c) \tag{9}$$

which will be used as the initial value for the continuous hidden state of the successive ODE block. The filter stride is set to 2 to reduce the input dimensionality and lower the computational cost for successive processing. No activation function is applied to the output of this operation, as it thought to only linearly map the input into a higher dimensional space. Second, the ODESolver is invoked to compute the evolution of the state $\mathbf{h}(t)$ and find its value at a fixed time $t = 1$

$$\mathbf{h}(1) = \mathrm{ODESolver}(\mathbf{h}(0), f, 0, 1, \theta)\,, \tag{10}$$

where the function describing the hidden state dynamics $f(\mathbf{h}(t), t, \theta)$ is implemented as in the Res-ODE architecture. The final state $\mathbf{h}(1)$ is further processed by applying Group Normalization, ReLU activation, and a global average pooling operation over the spatial dimensions to obtain a K-dimensional vector $\tilde{\mathbf{h}}$. Finally, a fully connected layer and the softmax function are applied to $\tilde{\mathbf{h}}$ to obtain classification predictions for the image.

We employed ODE-only nets with $K = 64$ and $K = 256$ respectively on MNIST and CIFAR-10 using the same training procedure described in Section 4.1, obtaining the results reported in Table 2. Repeating the analysis described in Section 4.1.1 on the continuous hidden state of the trained ODE-only nets, we can notice that now the ODE block describes a potentially complex hidden state dynamics that continuously evolve low-level features into high-level ones, as reflected by the significant differences in both test error and NFE when varying ε (see Figure 2, remind that the quantity ε can be thought as a way to control the trade-off between accuracy and efficiency.) This means that with this architecture, the whole output of feature extraction is under the continuity constraint, and thus the range of feature extraction controllable by ε is wider with respect to Res-ODE. On the other hand, the mean NFE to compute the output $\mathbf{h}(1)$ is higher compared to Res-ODE since the whole network is described by more complex dynamics, and thus the forward pass is in general computationally more expensive. Remind however, that the NFE takes into consideration only the evaluations inside the ODE block and does not include the additional overhead of computing the initial classical layers in Res-ODE nets.

A summary of all the architecture used is reported in Table 1.

4.3 Feature Extraction from ODE-Nets

In standard neural networks, features extracted from intermediate layers of generic object recognition models have been proven to be effective image representations to be used in multiple visual task [34], such as content-based image retrieval [35]. Different layers usually detect and encode different patterns in the image with an increasing level of abstraction. Each layer defines a common feature space in which we can compare and sort images by adopting standard similarity or distance functions between Euclidean vectors.

Following the same line of reasoning, we compute image representations by extracting features from the continuous hidden layer $\mathbf{h}(t)$ for different values of t to test their representational power for image retrieval. Setting lower values of t, we extract lower-level features from the image for a fraction of the total inference time. Differently from standard networks in which features can be usually compared only in the same common space, in ODE-Nets it is reasonable to compare features that comes from different points in time of the evolution of the hidden state: we expect $\mathbf{h}(t-\varepsilon)$ to be a better approximation of $\mathbf{h}(t)$ for smaller values of ε thanks to feature continuity and parameter sharing through time, while increasing values of ε reduces the computational cost of finding the evolution of the hidden state.

Building on this properties of ODE-Nets, we can derive a strategy for adaptive feature extraction in which we evolve $\mathbf{h}(t)$ until we exhaust the available time budget (e.g. depending on the current load of a server or the next deadline in real-time systems).

This strategy could be implemented in traditional residual networks by fixing the dimensionality of the feature spaces through depth and by computing as many layers as we can in the time budget. However, we have no guarantees that near layers produce comparable features, since each layer can produce a large discontinuous update of the hidden state.

ODE-Nets instead can be naturally applied in this context, and in particular ODE-only nets aim to maximize the amount of adaptive computation in the model. Moreover, also the dynamics of the hidden state are different for each particular input, and an adaptive ODE solver can perform the optimal number of iterations depending on it, as can be seen by the standard deviation of the NFE in Figure 2.

5 EXPERIMENTAL EVALUATION

To evaluate the quality of image features extracted from ODE-Nets, we perform experiments in an multimedia information retrieval scenario.

As a preliminary evaluation, we adopt MNIST [23] and CIFAR-10 [21], two 10-class classification datasets with respectively 60k and 50k training images and 10k test images both. We choose these datasets both to directly compare to results in the NeurIPS best paper [9], in which the authors employed MNIST, and to better support our analysis with additional experiments on CIFAR-10. Due to the high computational cost incurred when training ODE-Nets, we leave experimentation on larger-scale and higher-resolutions datasets to future work.

For each dataset, we employ the training set to train all the models (i.e. ResNet, Res-ODE, and ODE-only) as image classifiers and thus obtain a hierarchy of feature extractors. For CIFAR-10, we employ 256-kernel convolutions instead of 64 in the second part of all architectures, and in the training phase, we adopt commonly used data augmentation techniques — random crop, random horizontal flip, and color jitter. A careful tuning of hyperparameters for each model and dataset would further increase the perfomance globally, but since classification performance was not our main objective, we keep all other hyperparameters and training procedures fixed for all datasets and networks: dropout with 0.5 drop probability applied before the classifier, the SGD optimizer with momentum of 0.9, weight decay of 10^{-4}, batch size of 128, and learning rate of 0.1 reduced by a factor 10 every time the error plateaus. To numerically solve ODEs, we employ the Dormand–Prince method [10], an adaptive solver belonging to the explicit Runge–Kutta family[2]. The solver step size is variable and determined by comparing the difference between fourth- and fifth-order solution; a tolerance parameter (set to 10^{-3} in our experiments) controls whether larger integration steps are rejected based on the absolute and/or relative error. The performance of the trained classifiers are reported in Table 2.

[2]A PyTorch implementation is available at https://github.com/rtqichen/torchdiffeq.

As already done in other works [5, 24], we formulate a retrieval groundtruth relying on the labeled test set of each dataset: we consider each image as query, and we mark the images belonging to the same class as relevant for that query. For each image, we extract the output of the last layer before the classifier as its representation, that is the vector obtained after the global average pooling operation. For Res-ODE and ODE-only nets, this corresponds to the output of the ODE block $\mathbf{h}(t)$ after the average pooling is applied. We extract features choosing values of t between 0 and 1 with step 0.05 to simulate different possible early-stopping events. Also for ResNets, we extract intermediate features and consider them as approximations of the final feature layer: we consider the output of each of the six last residual blocks (block #1 to #6) plus their initial input (block #0) for a total of 7 different feature representations.

We obtain a score $s(q, i)$ of the image i for the query image q by computing the cosine similarity between the representations of q and i

$$s(q,i) = \frac{\mathbf{h}_q(t_q) \cdot \mathbf{h}_i(t_i)}{||\mathbf{h}_q(t_q)||_2 ||\mathbf{h}_i(t_i)||_2}\,, \tag{11}$$

where t_q and t_i respectively represents the final time value for the evolution of ODE-defined hidden states. We consider two possible scenarios: in the *symmetric* scenario, we compute the score using the representations of q and i both obtained evolving the hidden state of the ODE block to the same time value $t_q = t_i = t$; in the *asymmetric* scenario, we consider the case $t_i = 1$ in which the database feature is computed without early-stopping (e.g. in an off-line database indexing procedure), and we vary only $t_q = t$. We repeat the same procedure for ResNets, using the features from the same residual block for both q and i in the symmetric scenario while adopting the features from block #6 for i in the asymmetric scenario. In Figure 3, we report the mean average precision (mAP) over all queries in both symmetric and asymmetric scenarios for each model, dataset, in function of t and also in function of the mean number of function evaluations (mNFE) needed to extract features (which is itself a function of t already plotted in Figure 2).

ODE-only nets provide full and fine-grained control over the desired performance in terms of mAP, as they span most of the y-axis in Figure 3. On the other hand, ResNets only provide a finite number of configuration points represented by the horizontal black lines. The behavior of Res-ODE lies in between, since they do provide a continuous control over performance but only in the adaptive ODE-defined part of the architecture.

Interestingly, the asymmetric scenario always performs better than the symmetric one. This may initially be counter-intuitive, since we expected that features coming from different depths of the network would encode different information and thus not be comparable. Instead, each iteration or layer seems to incrementally add useful information to the current state in a coherent way. However, we claim that this behavior is caused by the similarity between the image classification task and the derived image retrieval benchmark, in which we reward the retrieval of object belonging to the same set of classes.

To further analyze this aspect, we perform additional experiments using Tiny-ImageNet-200[3], a 200-class classification dataset with 64x64 images extracted from the famous ImageNet subset used

[3]https://tiny-imagenet.herokuapp.com/

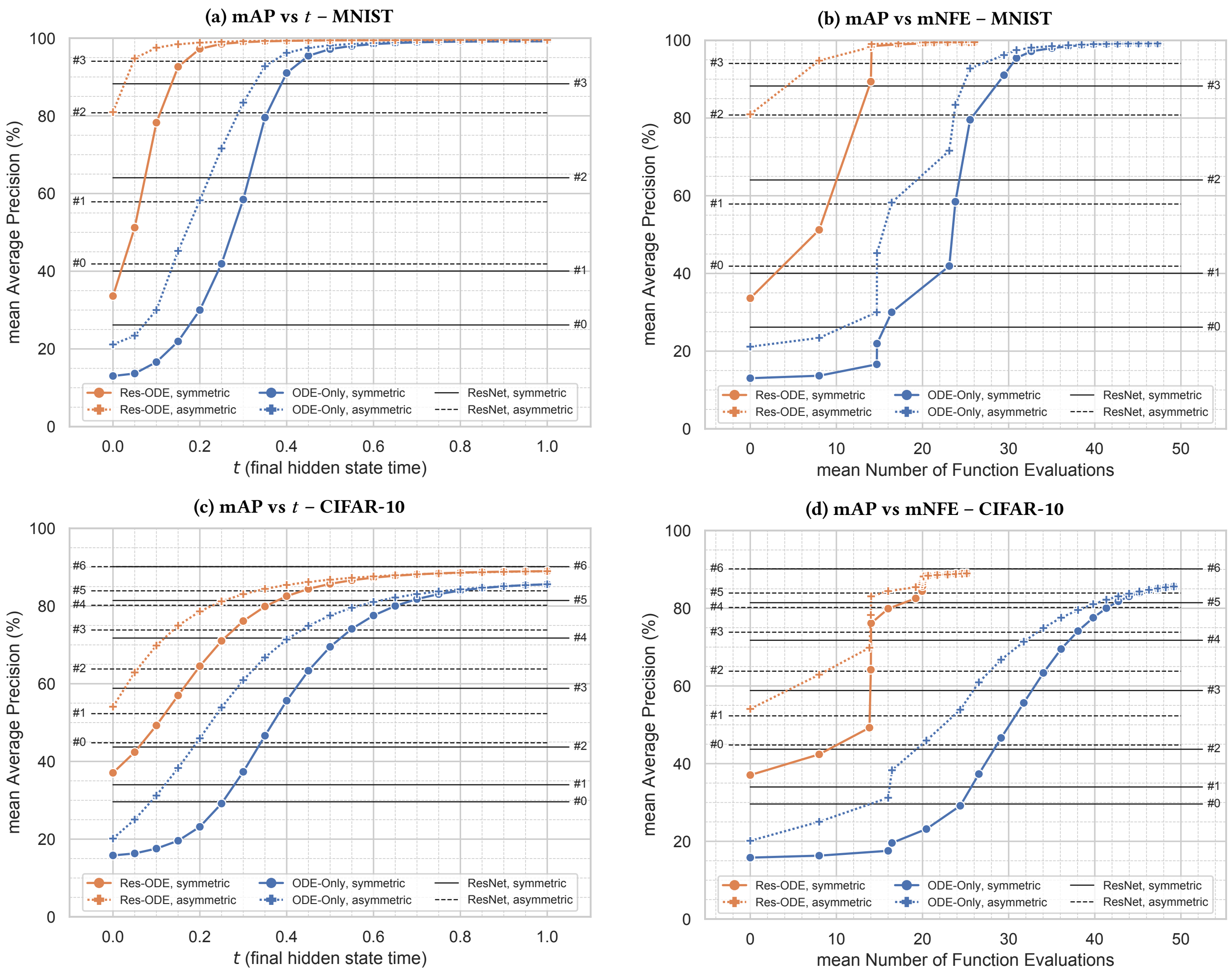
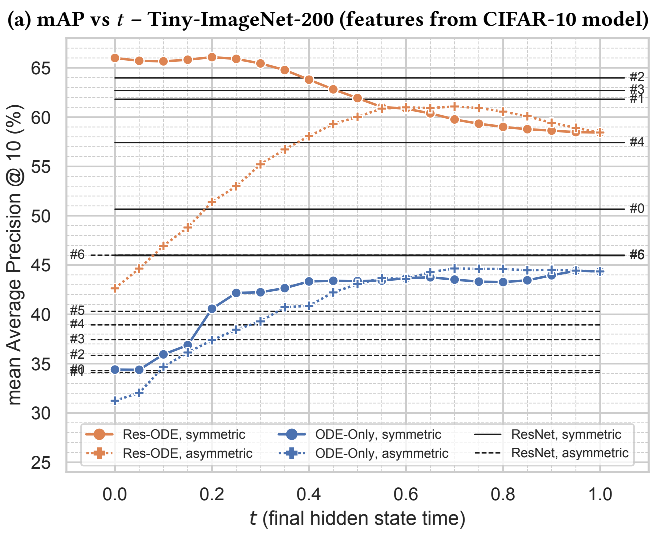


Figure 3: Mean Average Precision (mAP, %) when adopting $h(t)$ as image representation, in function of t (left) and in function of the mean NFE (right). Dashed lines refer to the asymmetrical case in which the features of the query $\mathrm{h}(t)$ are matched against the most accurate representations $\mathrm{h}(1)$. The black line indicates baseline values obtained using features extracted from residual blocks of the standard ResNet model; the number of the block from which the features of the queries are extracted is reported near each horizontal line. For MNIST, horizontal lines above 95% mAP are omitted to reduce clutter.

for the ILSVRC challenge. As previously done, we build a retrieval groundtruth over the validation set of Tiny-ImageNet-200, that has 50 images per class for a total of 10k images. Given the similarity between the domains of CIFAR-10 and Tiny-ImageNet-200, we tested the features extracted from the models trained on CIFAR-10 to perform image retrieval. In Figure 4, we report the mean Average Precision at position 10 (mAP@10).

As expected, we obtain lower absolute values of mAP in this configuration, since we are targeting a more complex visual task with many unseen classes. However, this is sufficient to observe the trend in the results caused by common or similar classes between the two datasets. The best performing features are the early activations of ODE-Nets that reach about 66% of mAP@10 and even perform slightly better than standard ResNets. Features from ODE-only nets instead transfer poorly to the new dataset obtaining at most 44% of mAP@10; this may be an indicator of domain overfit and suggests that this architecture needs to be better regularized. We observe that now the asymmetric setting performs worse that the symmetric one when using lower-level features (smaller values of t) and provide only small improvements when using higher-level features. For ResNet and Res-ODE, late activations are no more the best performing ones as image representations, as they focus on the CIFAR-10 classification task; instead, intermediate activations capture less specific features that better transfer to Tiny-ImageNet-200



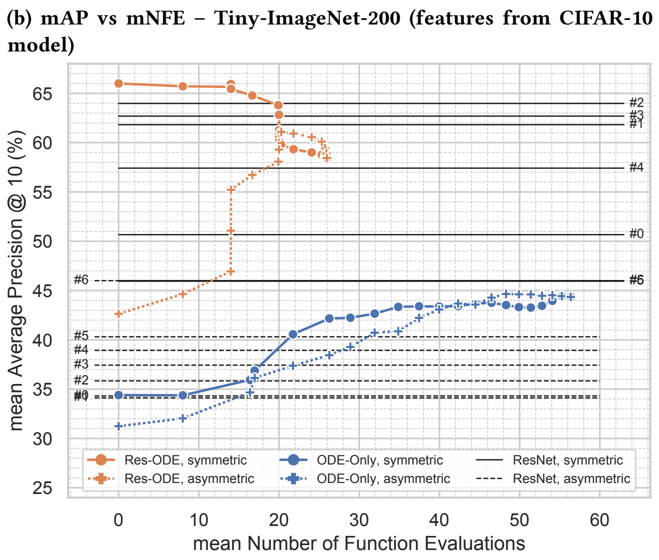


Figure 4: Mean Average Precision @ 10 (mAP@10, %) on Tiny-ImageNet-200 using features extracted from CIFAR-10 models. See the caption of Figure 3 for details.

images thus confirming findings of previous studies on standard neural networks [4]. Overall, results from the model with mixed layer types (Res-ODE) are promising for two reasons. First, they provide slightly better results suggesting more robust and general representations with respect to residual networks. Second, they combine the benefits coming from both classical and continuous layers, i.e. fast low-level feature extraction through classical layers, and adaptivity and controllability through ODE-defined layers.

6 CONCLUSIONS

In this work, we investigate ODE-defined continuous neural networks for learning image representations, and we evaluate them in an content-based image retrieval scenario. We exploit continuous models trained for image classification and extract features from different points of their continuous evolution. Thanks to variable-step ODE solvers, the feature extraction computational cost is automatically adjusted sample-wise without the need of additional parameters or control structures. Additionally, we explore early-stopping of the forward pass as a simple technique to approximate features at a reduced computational cost. We also expored a fully ODE-defined network, ODE-only Net, to adopt continuous representations in most of the feature extraction process, i.e. from initial low-level to class-level features.

Preliminary experiments on MNIST and CIFAR-10 show that both tested ODE-Nets (Res-ODE and ODE-only) perform comparably to standard residual networks in both the classification and retrieval tasks while offering a more fine-grained control on the effectiveness-efficiency trade-off thanks to continuity of representations. As concerns the transferability of image representation, the mixed architecture (Res-ODE) shows promising results obtaining higher average precision values when retrieving Tiny-ImageNet-200 images with representations trained on CIFAR-10. The obtained considerations are still preliminary, as additional experiments on more robust image retrieval benchmarks (such as Oxford Buildings [28, 29], INRIA Holidays [19] are required to gain deeper insight. However, the high computational cost of currently available methods and implementations for ODE-Nets represent a major limitation to performing larger-scale evaluations. Hopefully, new optimized learning strategies and technological advances for ODE-defined neural networks, such as ad-hoc learning-aware ODE solvers, will accelerate research in this direction.

As future work, we plan to analyze the effect of different ODE solvers and their parameters (especially the tolerance parameter of adaptive ones) to the effectiveness-efficiency trade-off of image representation extraction.

ACKNOWLEDGMENTS

This work was partially funded by “Automatic Data and documents Analysis to enhance human-based processes" (ADA, CUP CIPE D55F17000290009) and 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 and a Jetson TX2 used for this research.

REFERENCES

[1] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo. 2019. Large-scale instance-level image retrieval. *Information Processing & Management* (2019), 102100.
[2] Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo. 2016. Deep permutations: deep convolutional neural networks and permutation-based indexing. In *International Conference on Similarity Search and Applications*. Springer, 93–106.
[3] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. 2016. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5297–5307.
[4] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. 2014. Neural codes for image retrieval. In *European conference on computer vision*. Springer, 584–599.

[5] Fatih Cakir, Kun He, Sarah Adel Bargal, and Stan Sclaroff. 2017. MIHash: Online hashing with mutual information. In *Proceedings of the IEEE International Conference on Computer Vision*. 437–445.

[6] Fabio Carrara, Giuseppe Amato, Fabrizio Falchi, and Claudio Gennaro. 2019. Evaluation of continuous image features learned by ode nets. In *International Conference on Image Analysis and Processing*. Springer, 432–442.

[7] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. 2018. Reversible architectures for arbitrarily deep residual neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[8] Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. 2018. Multi-level Residual Networks from Dynamical Systems View. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SyJS-OgR-

[9] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*. 6572–6583.

[10] John R Dormand and Peter J Prince. 1980. A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics* 6, 1 (1980), 19–26.

[11] Matthijs Douze, Hervé Jégou, and Florent Perronnin. 2016. Polysemous codes. In *European Conference on Computer Vision*. Springer, 785–801.

[12] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. 2019. Augmented Neural ODEs. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. 3134–3144. http://papers.nips.cc/paper/8577-augmented-neural-odes

[13] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. 2017. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision* 124, 2 (2017), 237–254.

[14] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. 2019. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. *International Conference on Learning Representations* (2019).

[15] Eldad Haber and Lars Ruthotto. 2017. Stable architectures for deep neural networks. *Inverse Problems* 34, 1 (2017), 014004.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*. Springer, 630–645.

[18] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[19] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2008. Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*. Springer, 304–317.

[20] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. 2016. Cross-dimensional weighting for aggregated deep convolutional features. In *European conference on computer vision*. Springer, 685–701.

[21] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Citeseer.

[22] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2017. FractalNet: Ultra-Deep Neural Networks without Residuals. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. https://openreview.net/forum?id=S1VaB4cex

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[24] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2074–2081.

[25] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. 2018. Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 3282–3291. http://proceedings.mlr.press/v80/lu18d.html

[26] Yury A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[27] Nicola Messina, Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro. 2018. Learning relationship-aware visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 0–0.

[28] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 1–8.

[29] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. 2018. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5706–5715.

[30] Filip Radenović, Giorgos Tolias, and Ondrej Chum. 2018. Fine-tuning CNN image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[31] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. 2016. Visual instance retrieval with deep convolutional networks. *ITE Transactions on Media Technology and Applications* 4, 3 (2016), 251–258.

[32] Yulia Rubanova, Ricky TQ Chen, and David Duvenaud. 2019. Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907* (2019).

[33] Lars Ruthotto and Eldad Haber. 2019. Deep Neural Networks Motivated by Partial Differential Equations. *Journal of Mathematical Imaging and Vision* (18 Sep 2019). https://doi.org/10.1007/s10851-019-00903-1

[34] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 806–813.

[35] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. 2015. Particular object retrieval with integral max-pooling of CNN activations. *arXiv preprint arXiv:1511.05879* (2015).

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[37] Yuxin Wu and Kaiming He. 2018. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 3–19.

[38] Xingcheng Zhang, Zhizhong Li, Chen Change Loy, and Dahua Lin. 2017. Polynet: A pursuit of structural diversity in very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 718–726.

[39] Mai Zhu, Bo Chang, and Chong Fu. 2018. Convolutional Neural Networks combined with Runge-Kutta Methods. *arXiv preprint arXiv:1802.08831* (2018).