Deep Continuous Networks - Supplementary Material

ICML 2021 Submission

Paper ID: 3234

A Appendix

A.1 Gaussian multiscale local N-jet

Based on the Schwartz theory of smooth test functions, the Gaussian scale-space paradigm states that the derivatives $L_{i_1 \dots i_n}(\vec{x}; \sigma)$ of a function $L_0(\vec{x})$ with respect to the spatial variables x_i, with $i = 1 \dots d$, and at scale σ is given by the convolution

$$L_{i_1 \dots i_n}(\vec{x}; \sigma) = L_0 * \partial_{i_1 \dots i_n} G(\vec{x}; \sigma) \tag{A.1}$$

where $\partial_{i_1 \dots i_n}$ is the n-th order partial derivative and $G(\vec{x}; \sigma)$ is the normalized, isotropic Gaussian kernel with standard deviation $\sigma_i = \sigma$ and mean $\mu_i = 0$ (Florack et al., 1996). Note that $L_0(\vec{x})$ does not need to be a smooth function, and therefore the Gaussian scale-space paradigm can be applied to obtain local image derivatives in different scales, where $\vec{x}$ denote the spatial coordinates and σ can be interpreted as the coordinate in the scale dimension.

We build upon this definition of local image derivatives to build our structured receptive fields (SRFs), similar to Jacobsen et al. (2016). The main idea we leverage is that using a Taylor approximation, one can decompose an input image into a superimposition of its local derivatives. This decomposition can then be performed by local convolution kernels in CNNs, where the relative weight of each derivative order can be learned during training. In order to show this, we observe that the N-th order Taylor expansion of an image $L : \mathbb{R}^2 \to \mathbb{R}$ around a point (a, b) is given by

$$L(x, y) = \sum_{l=0}^{N} \sum_{k=0}^{N-l} \frac{(x-a)^l (y-b)^k}{l!k!} \frac{\partial^{l+k}}{\partial x^l \partial y^k} L(a, b) \tag{A.2}$$

where the partial derivatives of L with respect to x and y can be interpreted as $L_{1_1 \dots 1_n}(x, y; \sigma_0)$ and $L_{2_1 \dots 2_n}(x, y; \sigma_0)$ from equation A.1 at some original scale σ_0. This means that, via the linearity of convolution, and under the assumption that $L(x, y; \sigma)$ does not diverge, it is equivalent to either use the N-th order derivatives of the input image, or use the N-th order derivatives of the Gaussian function $G(\vec{x}; \sigma)$, to perform the image decomposition at scale σ. The local N-jets can thus be seen to be parametrized by the coefficients in the expansion given in equation A.2. By definition, we consider the Taylor expansion coefficients to be covariant derivatives of the image $L(x, y; \sigma)$, which are independent of the local coordinate system, to reach the filter approximations given in equation 1 in the paper, where the coefficients take the form $\alpha_{l,k}$.

In short, the multiscale local N-jet provides a natural decomposition of an image in a local neighborhood in the spatial and scale dimensions. Under convolution with $G(\vec{x}; \sigma)$, this decomposition provides a framework for defining convolutional filters in a CNN using N-th order Taylor polynomials. The SRF filters we use are based on this N-jet definition and allow us to learn the scale, spatial frequency and orientation of the filters during training, which are fundamental properties of biological receptive fields (Jones & Palmer, 1987; Lindeberg, 1993).

In addition, the Gaussian scale-space formulation of SRFs (Jacobsen et al., 2016) lead to theoretically interesting properties which strengthen the motivation for our choice of filters based on

the Gaussian N-jet. For example, the semi-group property of Gaussian scale-spaces indicates for a Gaussian derivative kernel $G^{l,k}(x, y; t)$ parametrized by the variance $t = \sigma^2$

$$G^{l,k}(x, y; t + t') = G^{l,k}(x, y; t) * G(x, y; t') \tag{A.3}$$

where the superscripts (l, k) denote the partial derivatives with respect to x and y. This means that a translation in scale dimension can be simply achieved through convolution with a (0-th order) Gaussian kernel G. For DCNs, this means that we have a solid understanding of the scale of the feature maps (in the sense of the Gaussian scale-space) at every layer m in the network, as we know the value of σ_m after training. In the absence of SRFs with an explicit scale parameter, this information is lost.

Similarly, SRF filters based on Gaussian derivatives are steerable by the coefficients $a_{l,k}$. For example, a second order filter with orientation θ can be described as a sum of basis functions

$$\begin{aligned} G^{2,0}_\theta &= a_{2,0}G^{2,0} + a_{1,1}G^{1,1} + a_{0,2}G^{0,2} \\ &= \cos^2(\theta)G^{2,0} - 2\cos(\theta)\sin(\theta)G^{1,1} + \sin^2(\theta)G^{0,2}. \end{aligned} \tag{A.4}$$

Finally, the set of Gaussian derivative basis functions $G^{l,k}(x, y; \sigma)$ are spatially separable

$$G^{l,k}(x, y; \sigma) = G^l(x; \sigma)G^k(y; \sigma) \tag{A.5}$$

which is useful for computational efficiency in numerical applications.

A.2 Training procedure

The basic architecture of all our models is given in figure 2 of the paper. Unless otherwise stated, in all models we use group normalization (Wu & He, 2018) with 32 groups in every layer as the normalization function. As the nonlinear activation, we use continuously differentiable exponential linear units (Barron, 2017), or CELU for DCN models. Based on the original ODE-Net implementation (Chen et al., 2018), for ODE-Net models and ResNet-block models, we use rectified linear units (ReLU). Likewise, when defining the integration time interval T, we stick to the original implementation, with $T = 1$ for ODE-Net models, whereas we use $T = 2$ for DCN models. Otherwise, all the hyperparameters are kept constant between models. For a brief overview of hyperparameter optimization, see appendix A.3.

All models are trained for 100 epochs on the standard CIFAR-10 training set (Krizhevsky, 2009) using cross-entropy loss. As data augmentation, we use random translations up to 4 pixels in each spatial dimension and random horizontal flips. Optimization is performed using SGD with a mini-batch size of 128, initial learning rate 10^{-1}, momentum 0.9, and a learning rate decay by a factor of 0.1 at epochs 40 and 70. For continuous time models based on neural ODEs, we use the adjoint method for backpropagating the losses and ODE solvers with error tolerance set to 10^{-3}.

For convolutional layers with pixel-based $k \times k$ filters, the weights are initialized using the standard Kaiming uniform initialization (He et al., 2015). For layers using SRF filters, the initial α values are randomly sampled from a normal distribution with mean 0 and standard deviation 0.1, and initial σ values are sampled from a normal distribution with mean 0 and standard deviation 2/3.

For the restricted CIFAR-10 experiments (small-data regime), we pick the total number of training images to be a multiple of our mini-batch size of 128, or otherwise have a minimal number of samples in the final batch (which is dropped in each epoch). For comparisons with the baseline results from Arora et al. (2020), we used exactly the same number of training images as in Table 2 of Arora et al. (2020). In order to confirm convergence, for the training set sizes $[520, 1030, 5120]$, we increased the number of training epochs by a factor of $[10, 5, 2]$ respectively.

For meta-parametrized models the initial values for the learnable parameters follow normal distributions $\mathcal{N}(\mu, \sigma_{\mathcal{N}})$: $a \sim \mathcal{N}(0, 2/3)$, $b \sim \mathcal{N}(0, 0.1)$, for the DCN $\sigma(t)$ model; $a \sim \mathcal{N}(0, 2/3)$, $b \sim \mathcal{N}(0, 2/3)$, $c \sim \mathcal{N}(0, 0.1)$ for the DCN $\sigma(t^2)$ model; and $a_s \sim \mathcal{N}(0, 2/3)$, $b_s \sim \mathcal{N}(0, 0.1)$, $a_\alpha \sim \mathcal{N}(0, 0.1)$ and $b_\alpha \sim \mathcal{N}(0, 0.05)$ for the DCN $\sigma(t)$, $\alpha(t)$ model.

For all our models using SRF filters, except for the DCN σ^{ji} model, we use scale sharing within a convolutional layer such that $\sigma_m^{ji} = \sigma_m$ for all convolutional layers m, with input channel i and output channel j. This makes the GPU implementation trivial, as all filters within a layer are sampled in the interval $[-2\sigma_m, 2\sigma_m]$, and hence the convolutional kernel sizes within a layer are uniform. However, for the DCN σ^{ji} model, a GPU implementation would be highly inefficient if we truncated the kernels at a fixed factor of σ^{ji}, independently for each input and output channel i, j. Therefore for the DCN σ^{ji} model we fix the kernel size at 7×7, but we still learn the shape and the scale (bandwidth) of the filters during training.

We use the Dormand–Prince (DOPRI) method (Dormand & Prince, 1980) as the numerical ODE solver. The DOPRI method is an explicit, adaptive solver of the Runge-Kutta family, which uses 6 function evaluations to compute fourth- and fifth-order accurate solutions to ODEs, along with an error estimate. The size of the adaptive steps taken by the solver can be regulated by specifying an error tolerance on this error estimate.

As is the case with most modern ODE solvers, the GPU implementation of the DOPRI solver that we use (Chen et al., 2018) considers the input arguments for the integration time interval $t \in [0, T]$ (or $t \in [0, 2]$ in the case of all DCN models) as soft bounds. This means that the DOPRI algorithm may explore time points outside of this interval, based on its internal error estimation, and may then employ interpolation to return solutions within the specified bounds. In the meta-parametrized models, where for example σ is an explicit function of t, this may lead to very large or very small kernel sizes, ordinarily unexpected within the integration time interval. In order to avoid numerical instability and memory issues in the meta-parametrized models, we scale down and clip the integration time t when passing it into the parameter definitions as $\sigma(\tau t_{\text{clip}})$ and $\alpha(\tau t_{\text{clip}})$. We clip the t values in the interval $[-0.5, 2.5]$ and use $\tau = 0.5$.

A.3 Hyperparameter tuning

As mentioned in appendix A.2, we share all the design choices and hyperparameters between all DCN and baseline ODE-Net models, except for the nonlinear activation function and integration time interval T.

Table A.1: CIFAR-10 validation accuracy (averaged over 3 runs) in the control experiments testing the effect of DCN model design choices on the baseline ODE-Net.

Model	Accuracy (%)
DCN-ODE	89.46 ± 0.16
ODE-Net (baseline)	89.60 ± 0.28
ODE-Net with $T = 2$	89.50 ± 0.07
ODE-Net with CELU	89.33 ± 0.16
ODE-Net with CELU and $T = 2$	89.25 ± 0.30

This difference in design choices arises since for the ODE-Net baseline we stay faithful to the

original ODE-Net implementation, where ReLU is the nonlinear activation function and $T = 1$, whereas for DCN models we use CELU as the activation function, due to its generalized compatibility with the adjoint method, and $T = 2$. In order to verify that our design choices do not provide an unfair advantage over the ODE-Net baseline, we run some control experiments, where we vary the activation function and T in the ODE-Net baseline.

We find that the change of activation functions or integration interval T do not provide a significant increase to the CIFAR-10 classification performance in the ODE-Net baseline (Table A.1).

A.4 CIFAR-10 image reconstruction

For the reconstruction task, we use an encoder with 3 DCN-ODE blocks and a decoder with 3 DCN-ODE blocks (as shown in figure 2 of the paper). For the baseline networks we replace the 3 DCN-ODE blocks with ODE-Net or ResNet-blocks. The encoder is pre-trained on the CIFAR-10 classification task. We use the feature maps at the end of ODE Block 3 as the input to the decoder network. The decoder DCN network is made up of 2 ODE blocks separated by bilinear upscaling, 1×1 convolutions to reduce the number of channels, normalization, non-linear activation and a final 1×1 convolution to generate the output in RGB space.

We implement reconstruction as a regression task and use the mean squred error (MSE) as the loss function. Otherwise, the training parameters are the same as the classification experiments: We use the SGD optimizer with a mini-batch size of 128, learning rate 10^{-1} and momentum 0.9, together with the adjoint method and an error tolerance of 10^{-3}.

Some example image reconstructions (randomly selected) from the CIFAR-10 validation set by the DCN and baseline networks are shown in figure A.1.

A.5 Pattern completion in feature maps

In figure A.2, we show the feature map evolution within the first ODE block (or ResNet block) of different models with and without masking of some example input images. Size of the mask depicted here is 6×6 pixels and the example images were chosen so as to have the mask located close to the middle of the object. We picked some channels with visible mask-related artifacts in the input feature maps to the first ODE (ResNet) block. Since there is no feature map trajectory in the ResNet-blocks model, but only one input and one output feature map, the difference between the feature maps of the intact and masked image is given as a scatter plot of two data points connected by a red line.

Figure A.3 depicts the average difference of intact and masked feature maps $\overline{D}(t)$ averaged over 1000 images and the standard deviation for the DCN and baseline networks.

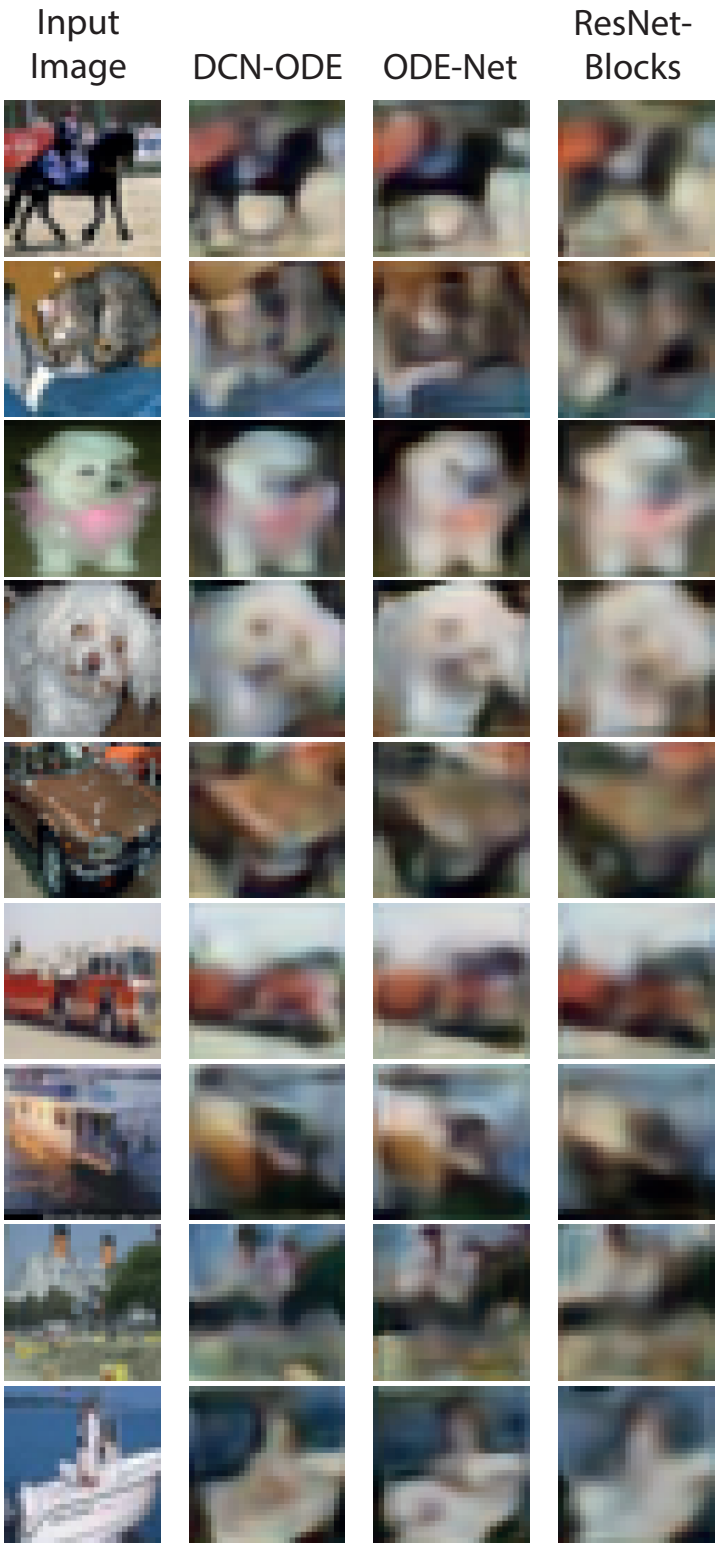

Figure A.1: Example CIFAR-10 validation images and their reconstruction by the DCN-ODE model as compared to baseline models.

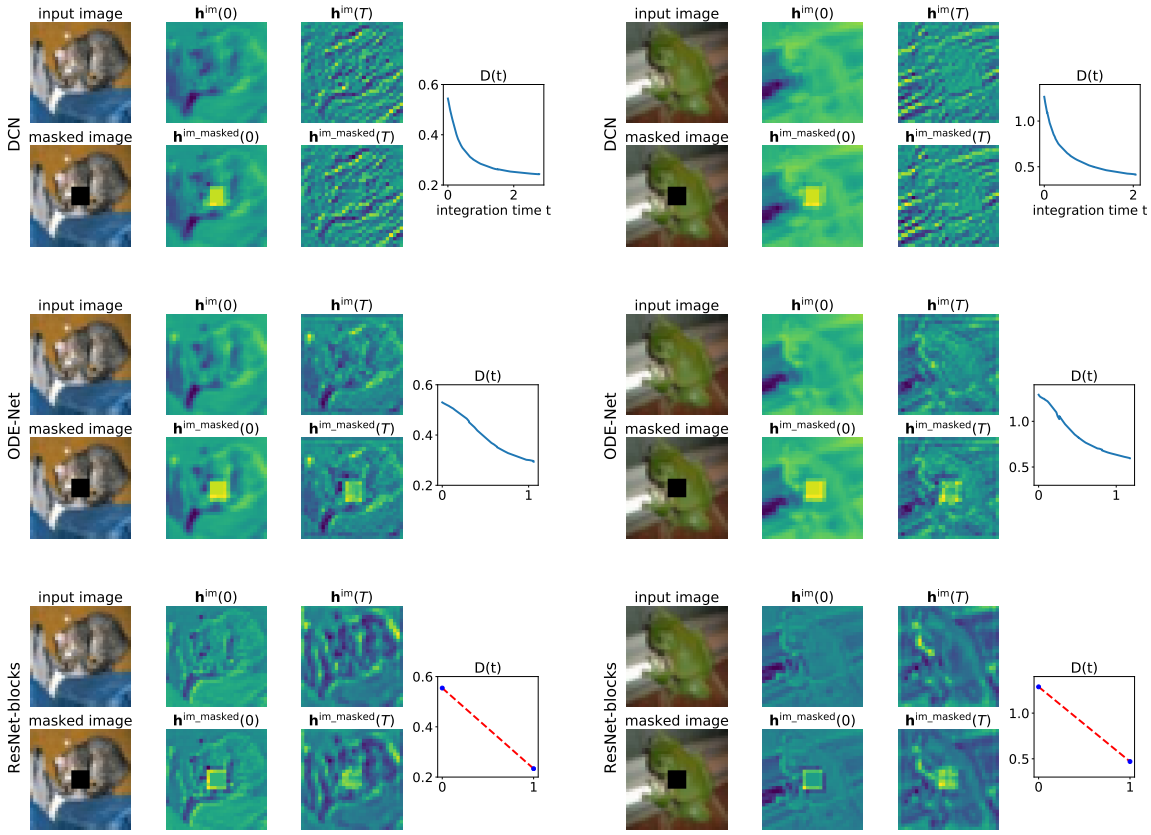

Figure A.2: Feature map evolution within the first ODE block (or ResNet block) of different models.

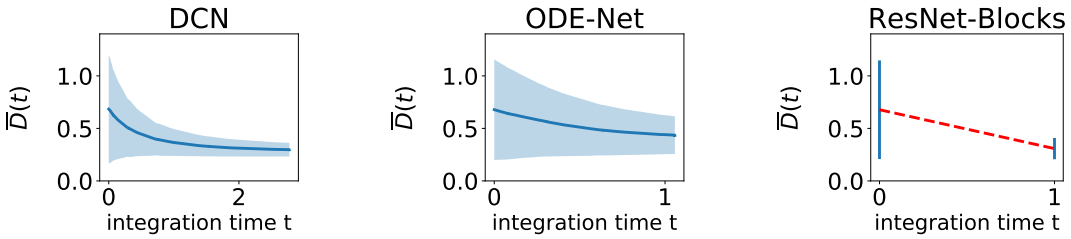

Figure A.3: Evolution of the mean difference $\overline{D}(t)$ between feature maps of an intact input image and a masked input image, averaged over 1000 images in the CIFAR-10 validation set. The shaded areas (or in the case of ResNet, the errorbars) show the standard deviation.

References

Arora, S., Du, S. S., Li, Z., Salakhutdinov, R., Wang, R., and Yu, D. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations (ICLR)*, 2020.

Barron, J. T. Continuously differentiable exponential linear units. *arXiv preprint arXiv:1704.07483*, 2017.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. *NeurIPS*, 2018.

Dormand, J. R. and Prince, P. J. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

Florack, L., Romeny, B. T. H., Viergever, M., and Koenderink, J. The gaussian scale-space paradigm and the multiscale local jet. *IJCV*, 18(1):61–75, 1996.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

Jacobsen, J.-H., van Gemert, J., Lou, Z., and Smeulders, A. W. Structured receptive fields in CNNs. In *CVPR*, pp. 2610–2619, 2016.

Jones, J. P. and Palmer, L. A. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, 58(6):1233–1258, 1987.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Department of Computer Science, 04 2009.

Lindeberg, T. Discrete derivative approximations with scale-space properties: A basis for low-level feature extraction. *Journal of Mathematical Imaging and Vision*, 3(4):349–376, 1993.

Wu, Y. and He, K. Group normalization. In *ECCV*, pp. 3–19, 2018.