Condensing CNNs with Partial Differential Equations

Sohom Mukherjee
Student Number: 7010515
somu00003@stud.uni-saarland.de

Abstract

In this project, we explore the so called *Global* layer proposed in Kag and Saligrama [2022], which is a PDE-based feature layer aimed as capturing global receptive field without additional overheads of large kernel size or large depth which is common in existing CNN architectures. Such Global layers created by enforcing PDE constraints on feature maps have been suggested to create richer feature maps, and can be embedded in any existing CNN architecture reducing its depth. We conduct extensive experiments comparing Global layer-based CNNs versus existing CNN (across datasets and architectures as well as ablations studies), and arrive at the conclusion that the Global layer can successfully reduce the computational and storage budget of CNNs by a substantial amount at negligible of no loss in performance. The TensorFlow code for this project is availabe at https://github.com/mukherjeesohom/MDDL_Project

1 Introduction

Convolutional kernels that are the building blocks of existing Convolutional Neural Networks (CNNs) are characterised by a small receptive field, which necessitates moderns CNNs to be very deep Simonyan and Zisserman [2014] for capturing global information. This in turn increases the computational complexity (i.e., high training and inference times) and storage budget (i.e, large model sizes) for such CNNs. The advent of the concept of skip connection largely popularised by ResNets He et al. [2016], solved the problem of vanishing gradient in deeper networks and gave major improvements in large scale image recognition tasks. Deeper networks remain computationally and storage expensive, nonetheless.

Recently, there has been an emergence of interest in developing smaller and more compact deep networks. One line of work in this direction involves network pruning popularised by the Lottery Ticket Hypothesis Frankle and Carbin [2018]. On another note, there has also recently been a wide range of work connecting partial differential equations (PDEs) and CNNs: Ruthotto and Haber [2020] proposes new architectures called parabolic and hyperbolic CNNs based on the corresponding PDEs. Alt et al. [2022] studies connections between numerical algorithms for solving PDEs (in particular, diffusion equations) and CNN, thereby transferring classical properties like stability and well-posedness of diffusion schemes to CNNs.

For existing CNN architectures, the same convolutional block is stacked m times, at each resolution. In this paper, the authors suggest retaining only one such block in each resolution and replacing the rest of $m-1$ blocks by a single Global feature layer (schematic diagram depicted in Figure 1). The Global layer approximately solves a PDE constraint (based on a standard PDE and numerical solver) that couples the input and output feature maps. Thus the Global layer can be embedded in any existing discrete CNN, and also makes the resulting CNN much shallower. The experiments conducted in this project verifies that the Global layer can be embedded in any existing CNN and the resulting network is (refer Table 2 for results):

- **Swallow.** In terms of depth. For our experiments in Section 3.2, the proposed ResNet32-Global architecture has around $3\times$ less depth than ResNet32.

Model-driven Deep Learning Lab for Image Analysis 2022.

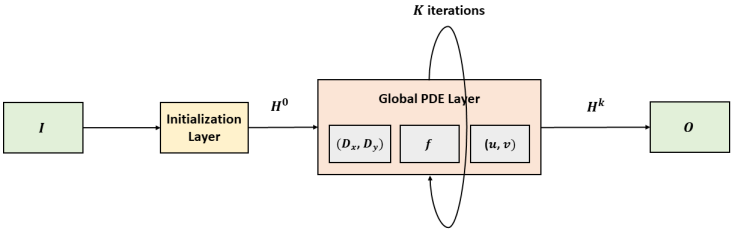


Figure 1: Schematic diagram for the proposed Global PDE layer.

- **Less Storage.** In terms of number of parameters. ResNet32-Global architecture has around $5\times$ less parameters than ResNet32.
- **Less Computation.** In terms of MACs. ResNet32-Global architecture uses around $5\times$ less MACs than ResNet32.

2 Proposed Method

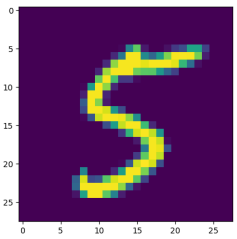
In this section we take a brief look at the theory behind the aforementioned PDE constraint. We first look at the modelling of the PDE, followed by a simple iterative numerical solver, initialization, and finally choice of PDE parameters. Consider an input feature map $I(x, y) \in \mathrm{R}^{h\times w}$ and output feature map $H(x, y) \in \mathrm{R}^{h\times w}$.

1. **PDE Modelling.** Following the paper, our Global layer is based on the generic advection-diffusion equation:
$$\frac{\partial}{\partial t}H = \nabla \cdot (D\nabla H) - \nabla \cdot (\mathbf{v}H) + f(I) \tag{1}$$
where $\nabla \cdot (D\nabla H)$ represents the diffusion term with diffusivity $D = (D_x, D_y)$, $\nabla \cdot (\mathbf{v}H)$ represents the advection term with velocity coefficient $\mathbf{v} = (u, v)$, and f represents a source. We also consider specialized cases of this general equation, namely the diffusion equation, and the advection equation separately in our ablation studies (Section 3.3). A special case is the nonlinear isotropic diffusion proposed by Perona and Malik [1990], where the diffusivity is given by $D = \frac{1}{1+|\nabla H|^2/\lambda^2}$.
2. **Discretization.** We can discretize the 2D version of Equation 1, using finite difference scheme as discussed in the paper, with step size δ_x, δ_y, and δ_t for x, y, and t as follows:
$$\begin{aligned} LH^{k+1}_{x,y} = MH^{k-1}_{x,y} - 2(u_x + v_y)\delta_t H^k_{x,y} + 2\delta_t f(I(x, y)) + (-A_x + 2B_x)H^k_{x+1,y} \\ + (A_x + 2B_x)H^k_{x-1,y} + (-A_y + 2B_y)H^k_{x,y+1} + (A_y + 2B_y)H^k_{x,y-1} \end{aligned} \tag{2}$$
where $L = (1 + 2B_x + 2B_y)$, and $M = (1 - 2B_x - 2B_y)$. $u_x = \frac{u_{x+1,y}-u_{x-1,y}}{2\delta_x}$, and $v_y = \frac{v_{x,y+1}-v_{x,y-1}}{2\delta_y}$. $A_x = \frac{u\delta_t}{\delta_x}$, $A_y = \frac{v\delta_t}{\delta_y}$, $B_x = \frac{D_x\delta_t}{\delta_x^2}$, $B_y = \frac{D_y\delta_t}{\delta_y^2}$. We perform K iterations of the above iterative solver, to obtain the output feature map H at time $t = T = K\delta_t$.
3. **Initialization.** A good choice of initialization is necessary for convergence of the solver. The paper suggests using a learnable initialization, where $H(t = 0)$ is initialized with the output of the first block of the CNN and its parameters is learned. We also explore the option of choosing initialization as an identity function of the input feature map in our ablation experiments.
4. **PDE Parameters.** The free parameters of the PDE Equation 1 are chosen and experimented with as follows:
   (a) Diffusivity $D = (D_x, D_y)$ is chosen to be learnable depth-wise convolution[1] by default. We also explore other options like learnable Residual Block, constant value, and a Perona-Malik diffusivity Perona and Malik [1990] in ablation study (Section 3.3).

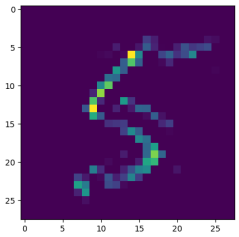
[1]Depth-wise convolution uses same kernel size as original block.

(b) Velocity Coefficient $\mathbf{v} = (u, v)$ is chosen to be learnable depth-wise convolution by default. We also explore other options like identity function in ablation study (Section 3.3).

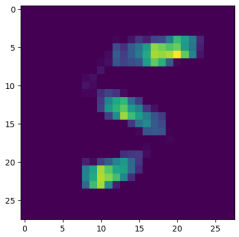
(c) Function $f(I)$ is chosen as an identity function.



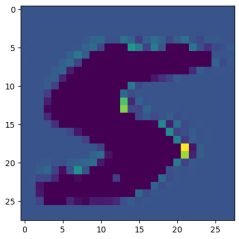
(a) Input image.



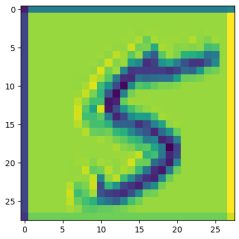
(b) Conv feature map.



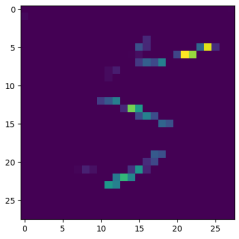
(c) Residual feature map.



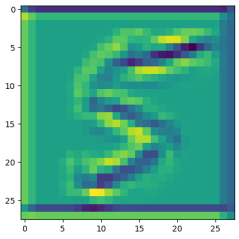
(d) PDE feature map ($D = 1$).



(e) PDE velocity map ($u = v$)



(f) PDE feature map (Nonlin Iso).



(g) PDE velocity map (Nonlin Iso).

Figure 2: Visualizations for MNIST illustrative example corresponding to the first half of Table 1 where the Conv net has 526 parameters and the Residual net and PDE net have 530 parameters each.

Table 1: Results for MNIST illustrative example.

Backbone	**Accuracy (%)**	**# Params**
Conv	64.59	526
Residual	66.04	530
PDE (constant $D = 1$)	65.31	530
PDE (Nonlinear Isotropic)	**66.24**	530
Conv	93.02	4,614
Residual	95.58	4,646
PDE (constant $D = 1$)	**95.8**	4,646
PDE (Nonlinear Isotropic)	95.2	4,646

3 Experiments and Results

All experiments have been conducted on an a laptop computer equipped with a NVIDIA RTX 3080 GPU, and 16 GB RAM. We follow the Algorithm 1 for the proposed method from the paper Kag and Saligrama [2022], as well as the official PyTorch implementation to create our TensorFlow implementation for this project. For the PDE solver, we used $K = 5$ iterations, and discretization $\delta_t = 0.2$, and $\delta_x = \delta_y = 1$. The code for the porject has been made publicly available at [2].

3.1 MNIST Illustrative Example

The MNIST dataset LeCun et al. [2010] consists of 10 classes and 28×28 images. We conduct experiments on MNIST to obtain a visual understanding of the global feature layer (in replication for Figure 2 of the original paper). The experiments use the basic data augmentation techniques (horizontal flip, padding, and random crop). We train with a batch size of 32, for 100 epochs, initial learning rate of 10^{-2} with CosineDecay learning rate scheduler.

For this illustrative example, we construct a simple network with an input convolutional layer, a feature layer, followed by a average pooling (with stride 4) and dense output layer with softmax activation. The feature layer can be (a) a convolutional layer, (b) a residual layer, or (c) a PDE-based Global layer. In all cases we use 2D convolution with kernel size 3 and stride 1, and convolution is followed by batch normalization and ReLU layers. The original paper suggests experiments with convolutional kernel having output channels 1, resulting in 524 parameters. However, we were not able to achieve the accuracy reported in the paper ($\sim 90\ \%$) using this many parameters. So besides output channels 1, we also report experiments for output channels 8. In these experiments, the network with Global layer is found to perform equal to or slightly better than the residual network, and strictly better than the convolutional network. The results can be found in Table 1.

We obtain visualizations for the input, the intermediate feature representations for (a), (b), and (c) Figure 2. For (c) we also obtain visualizations for the learned velocity coefficient $\mathbf{v} = (u, v)$ (here assumed $u = v$ for this simple illustrative example). Moreover, for (c) we make two sets of visualizations, firstly for the default constant diffusivity $D_x = D_y = 1$ suggested by authors, and also for the the nonlinear isotropic diffusion (Perona-Malik diffusivity).

Unfortunately the visualizations are not as conclusive as in case of the original paper in spite of exact replication of their experiment in TensorFlow.[3] However, we do observe that the residual net and PDE net have a sharper feature representation than the conv net, explaining their better performance. In the visualization of the learned velocity coefficients, we observe that there is non-zero features along the edges of the digit. Moreover, using nonlinear isotropic diffusion with Perona-Malik diffusivity gives a sharper feature map than using constant diffusivity $D = 1$.

[2] `https://github.com/mukherjeesohom/MDDL_Project`. Please refer to the `README` section for details on how to run baseline and ablation experiments.

[3] The difference might be attributed to the fact that original PyTorch code uses log-softmax activation and negative log likelihood loss, while we use softmax activation and cross entropy loss in TensorFlow.

Table 2: Results for CFAR-10 experiments. The rows have been grouped and partitioned from top to bottom: baseline, ablations on $\mathbf{v}$ and D, ablation on initialization, ablation on PDE model.

Architecture	Velocity Coeff. $\mathbf{v}$	Diffusivity D	Accuracy (%)	Initialization	# Params	# MACs
ResNet32	–	–	**93.92**	Identity	844K	75M
ResNet32-Global	Depth-wise Conv	Depth-wise Conv	91.27	Identity	163K	16M
ResNet32-Global	Identity	Constant ($D = 1$)	90.03	Identity	158K	14M
ResNet32-Global	Basic Residual Block	Basic Residual Block	**92.73**	Identity	550K	72M
ResNet32-Global	Depth-wise Conv	Nonlin Iso (Perona-Malik)	89.79	Identity	160K	16M
ResNet32-Global	Depth-wise Conv	Depth-wise Conv	92.18	Basic Residual Block	261K	30M
ResNet32-Global	0	Depth-wise Conv	90.06	Identity	Diffusion Equation	
ResNet32-Global	Depth-wise Conv	0	90.21	Identity	Advection Equation	

3.2 CIFAR-10 Experiments

The CIFAR-10 dataset Krizhevsky et al. [2009] consists of 10 classes and 32×32 images. We conduct experiments on CIFAR-10 dataset and ResNet32 architecture He et al. [2016] to benchmark the performance of the proposed Global layer. The experiments use the basic data augmentation techniques (horizontal flip, padding, and random crop). We train with a batch size of 32, for 200 epochs, initial learning rate of 10^{-2} with CosineDecay learning rate scheduler.

Following the original implementation of the paper, we build the ResNet32 architecture consisting of four resolutions with output channel count $[16, 32, 64, 64]$, and strides [1, 2, 2, 2], each having $m = 5$ blocks. We obtain the ResNet32-Global architecture by replacing the repeating residual basic blocks by a single basic residual block followed by a Global layer (i.e., here $m = 1$). For this set of experiments we use learnable diffusion coefficient D and velocity coefficient $\mathbf{v}$ with depth-wise convolutions. The results can be found in Table 2. For this baseline experiment, ResNet32-Global with depth-wise learnable $\mathbf{v}$ and D performs slightly worse than the ResNet32, but offers a $5\times$ reduddction in number of parameters, and needs $5\times$ less MACs.

3.3 Ablation Studies

We conduct further ablation studies based on the previous setup (CIFAR-10 dataset and ResNet32-Global architecture), to decide the best set of PDE parameters, as well as the effect of choosing a certain PDE model.

1. **PDE Modelling.** We perform experiments considering only the (a) diffusion equation and (b) advection equation as our PDE model. Experiments show that these are slightly worse performing than the diffusion advection baseline.
2. **PDE Initilization.** We try two initializations: (a) basic residual block, and (b) identity, as mentioned before. The basic residual block initialization performs slightly better than the identity but at the cost of around $2\times$ parameters and $2\times$ MACs.
3. **PDE Parameters.** Besides using both D and $\mathbf{v}$ as depth-wise convolution, we experiment using both as basic residual blocks. We observe that using the learnable parameters as basic residual blocks gives marginally better performance than depth-wise convolution, but costs $3.5\times$ parameters and $4.5\times$ MACs. Further, we also try making $\mathbf{v}$ as an identity function and set $D_x = D_y = 1$, to see if learnable parameters indeed affect the performance. Experiments show that using constant parameters lowers the accuracy by a small margin ($\sim 1\%$).

3.3.1 Extension

As an extension of the paper, we experiment with nonlinear isotropic diffusion using the Perona-Malik diffusivity (choosing $\lambda = 2$). The Perona-Malik diffusivity basically causes slower diffusion at edges, and vice versa. Therefore, we approximate the gradient information $|\nabla H|$ by the Sobel edge map. Experimental results show that, using the Perona-Malik diffusivity improves the sharpness of the feature map for MNIST visualization experiment, and the accuracy is slightly better than using constant diffusivity $D = 1$. However, for CIFAR-10 experiments, nonlinear isotropic diffsuion with Perona-Malik diffusivity performs slightly worse when compared to the baseline of learnable depth-wise convolution diffusivity.

4 Conclusion and Future Work

In this project, we did extensive experiments to replicate results in the paper Kag and Saligrama [2022], verify their claims, and perform some new ablation studies and provide a minor extension. While some minor aspects from our experiments do not match with the original paper, the bigger idea proposed by the authors does come through and is verified by our experiments. CNNs with the embedded Global layer do perform at par or slightly worse than the normal CNNs, but have the benefit of lesser depth, memory, and compute. Ablations show that using more learnable components in the the PDE (such as PDE initialization, PDE parameters) will improve the performance, but also cost more memory and compute. Therefore, we can conclude that using an optimal balance between learnable and deterministic components should be the way to go, to get the best trade-off between performance and complexity.

References

Anil Kag and Venkatesh Saligrama. Condensing cnns with partial differential equations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 610–619, 2022.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62(3):352–364, 2020.

Tobias Alt, Karl Schrader, Matthias Augustin, Pascal Peter, and Joachim Weickert. Connections between numerical algorithms for pdes and neural networks. *Journal of Mathematical Imaging and Vision*, pages 1–24, 2022.

Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 12(7):629–639, 1990.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. att labs, 2010.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.