Neural Ordinary Differential Equations for Building a new Efficient Deep Neural Network Model

Yan Nan yann@cs.wisc.edu

Yuhan Liu yliu738@wisc.edu

Wei Hao whao8@wisc.edu

Problem Proposal

Ordinary Differential Equations (ODE) proves a very interesting and novel approach to thinking about neural networks, which will bring an important advantage of model that one may freely choose the balance between a fast and a precise solution by affecting the precision of the numerical integration during evaluation and/or training. Also, the model is very generally applicable (requiring only that the nonlinearities of the neural network be Lipschitz-continuous) and may be applied to time-series modeling, supervised learning, density estimation or other sequential processes.

In our study, we are going to apply the ODE-Net on some more complex datasets such as CIFAR-10 or CIFAR-100 (https://www.cs.toronto.edu/~kriz/cifar.html) and more complicated models to see the effect of ODE-Net. We plan to apply ODE-Net on object detection problems to generate comparison between the 6 block ResNet and ODE-Net on precision, recall, and also the computation speed. We would like to apply ODE-Net on the Faster R-CNN with ResNet (Ren et al., 2017), which was trained on ResNet101. (https://github.com/endernewton/tf-faster-rcnn) We also want to apply ODE-Net on depth of object problem. Researchers have been working on learning depth from single images using pre-trained VGG, which we would like to replace it with ODE-Net.

Importance and Meaning of the Problem

Many researchers are exploring the applications of Neural Ordinary Differential Equations. According to the paper (Chen et al., 2018), the Neural Ordinary Differential Equations is able to improve memory efficiency, parameter efficiency, and computation speed. The authors compared the results of utilizing Res-Net, RK-Net, and ODE-Net and showed that while achieving the almost the same test error rate, ODE-Net used only O(1) of memory comparing to O(L) memory used up by Res-Net. However, the ODE-Net was applied on the MNIST dataset (Le Cun et al.), which is a relatively old and simple dataset. This is not a very powerful test of the performance and computation speed of ODE-Net. Accordingly, it is very important to apply the ODE-Net on some more complicated dataset to examine the model performance, which will be more applicable to real life datasets. Additionally, the paper reported the experimental results of replacing 6 residual blocks with an ODESolve module, which is a small and simple network that may not apply to many complex datasets. As a consequence, it is also very significant to analyze the effects of applying ODE-Net on deeper neural networks.

We are interested in applying ODE-Net on image analysis problems because the scientists have been working on solving ODE for more than 100 years, and it is really interesting and challenging to integrate neural network and ODE. ODE-Net allows us to improve the neural network from discrete layers to continuous number of evaluations, which enhances parameter efficiency and computing efficiency. This allows us to apply it on more complex datasets and perform more detailed analysis.

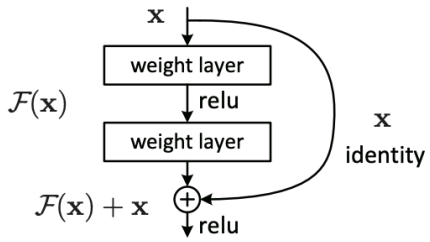
Current State-of-the-Art

In our project, we re-implement Neural Ordinary Differential Equations as a model component in a new deep neural network model. The use of black-box Ordinary Differential Equations (ODE) solvers as a model component, developing new models for time-series modeling, supervised learning, and density estimation.

Currently, there are models such as residual networks, recurrent neural network decoders and normalizing flows build complicated transformations by composing a sequence of transformations to a hidden state. We will first introduce the three model and then introduce the model in our project.

Compared with Convolutional Neural Network (CNN), Deep residual networks are improved in efficiency.[1]

Instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicitly let these layers fit a residual mapping. Denoting the desired underlying mapping as H(x), let the stacked nonlinear layers fit another mapping of F(x) := H(x) – x. The original mapping is recast into F(x) + x. With the hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers. The residual block is as followed.



The example network architectures for ImageNet is as followed.

A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit temporal dynamic behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled.

Both finite impulse and infinite impulse recurrent networks can have additional stored state, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of long short-term memory networks (LSTMs) and gated recurrent units.

Normalizing flows are invertible transformations of distributions. They enable one to transform simple probability densities to complex ones through a series of non-linear transformations, just as in a neural network. Hereby, they make use of the formula for variable exchange in a distribution:[2]

$$\ln q_K(z_K) = \ln q_0(z_0) - \sum_{k=1}^{K} \ln \left| \frac{\partial f_k}{\partial Z_{k-1}} \right|$$

, where q0(z0) is the initial distribution and $q_K(z_K)$ is the transformed distribution, with transformations f_k, k=0..K. The Jacobi determinant in the sum above hereby assures that the integral of the distribution function remains 1 throughout the transformation. Unfortunately, it is rather costly to calculate this determinant for all but some simple transformations.

For the OED in our project, we have sequences of transformations to Neural Differential Equations

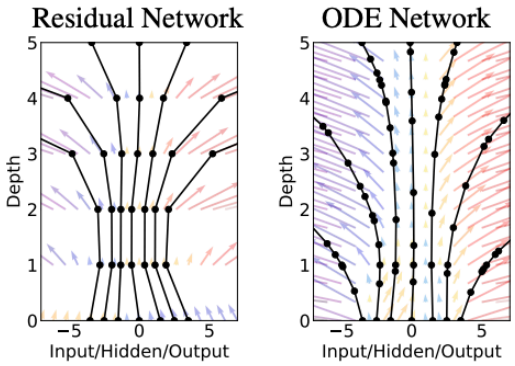
$$h_{t+1} = h_t + f(h_t, \theta_t)$$

By using OED to make discrete to be continues,

$$\frac{dh(t)}{dt} = f(h(t), t, \theta)$$

Our model provides ordinary differential equation (ODE) solvers implemented in PyTorch. Backpropagation through all solvers is supported using the adjoint method. The usage of ODE solvers in deep learning applications, see [3] The comparison of residual network and the ODE network is as followed.

Left is a Residual network defines a discrete sequence of finite transformations. Right is a ODE network defines a vector field, which continuously transforms the state. Both, Circles represent evaluation locations.



As the solvers are implemented in PyTorch, algorithms in this repository are fully supported to run on the GPU. In our model, we train the model on dataset like MNIST to prove these models are evaluated adaptively and allow explicit control of the tradeoff between computation speed and accuracy. Finally, we derived an instantaneous version of the change of variables formula, and developed continuous-time normalizing flows, which can scale to large layer sizes.

Our Plan and Timeline

First, we will re-implement an existing solution and apply it to different datasets like ImageNet and MNIST. Second, we will replace the residual networks with ODE to check its efficiency. Also, we will try to apply the ODE to build more complex model to see if there will be a more useful model on image classification.

To evaluate the performance and to test the model performance, we would like to compare the accuracy of replacing ODE-Net with ResNet on the CIFAR-10 dataset, including precision, recall, and F1 number on object detection. We also need to compare the performance of replacing ODE-Net with ResNet101. Lastly, we will test the performance of replacing VGG with ODE-Net to build the model to estimate the depth of object in a single image. We also would like to show the comparison of computation speed of the two models.

The timeline for our project is before March 10th: apply the ODE-Net on CIFAR-10 dataset and show the results and comparison between the model using ResNet and ODE-Net. Before March 30th, apply the ODE-Net on CIFAR-10 dataset, and show the results and comparison between the model using ResNet and ODE-Net. Before April 15th, try to finish applying ODE-Net on depth of object.

References

1. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
2. Rezende, Danilo Jimenez, and Shakir Mohamed. "Variational inference with normalizing flows." *arXiv preprint arXiv:1505.05770* (2015).
3. Chen, Tian Qi, et al. "Neural ordinary differential equations." *Advances in Neural Information Processing Systems*. 2018.