Neural Ordinary Differential Equations for Building a New Efficient Deep Neural Network Model Mid-Term Report

Yuhan Liu yliu738@wisc.edu
Wei Hao whao8@wisc.edu
Yan Nan yann@cs.wisc.edu

In this report, we will give a brief summary of current progress, including our current results, the difficulties that arise during the implementation, and how our proposal may have changed in light of current progress.

1. Summary of Current Progress

   a. We have read and understood the code for ODE net from Chen et al. (2018) [1]. Currently, we have reimplemented the code from the paper. The code is available below.
   https://drive.google.com/drive/folders/1BffKkw-A4N_P0M0cXQe6vK1bPhvBG0ie.
   In our experiments, we first examined the model and trained on MNIST dataset for 160 epochs. Below are the prediction results on MNIST test dataset generated by the trained model. We tested hand writing digits first to start on easy dataset.

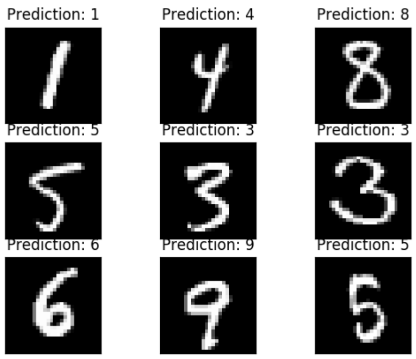


Figure 1. Prediction on handwriting digits

   b. In previous model, we trained the ODE net model on MNIST dataset. To test the reliability of ODE net, we decide to train the model with more complex dataset. We then modified the model to train on Cifar10 dataset and wrote a model visualization tool to load the trained model and show the predicted classification results of the model as shown below. We trained the ODE net model with Cifar10 for 160 epochs and 200 epochs achieving basically same test accuracy. Training with 160 epochs and 200 epochs both stuck in the same level of test accuracy (around 0.85) and the loss achieves lowest (0.1145) at epoch 181 and converges as shown in the graph.

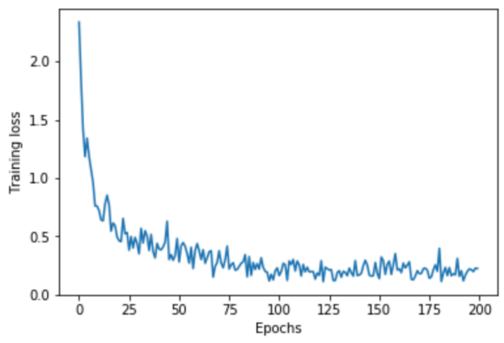


Figure 2. Training Loss as the epochs increase

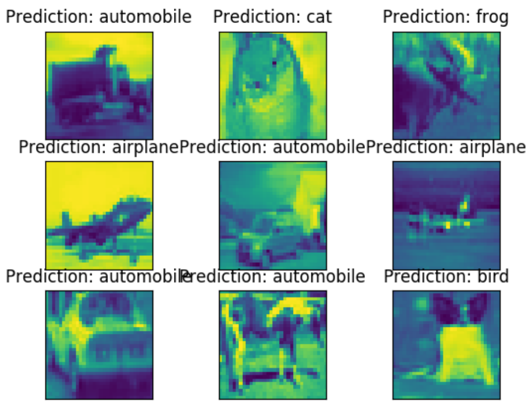
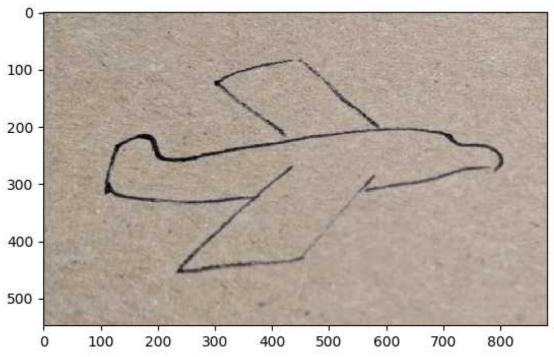
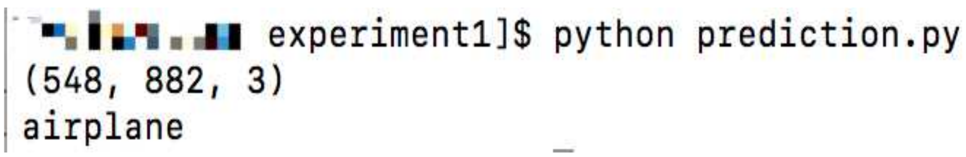


Figure 3. Prediction based on training dataset CIFAR 10

Above is the prediction of our trained model for CIFAR 10 test images using classes of 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'. After feeding the trained model with a random image on the internet and running the prediction code, we received the output which shows the prediction result (airplane) as shown below.





```
[redacted] experiment1]$ python prediction.py
(548, 882, 3)
airplane
```

Figure 4. Above is the image random from internet, below is the prediction result of our model

Attached is the link of code we reimplement the model and the experiments we made to test. Experiment1 file is the tests we make.
https://drive.google.com/drive/folders/1BffKkw-A4N_P0M0cXQe6vK1bPhvBG0ie

2. Difficulties and Challenges

Overall, the difficulties and challenges we have so far focus: on first, the classification prediction accuracy of our model and its accuracy when apply it to actual image classification system.

To improve the train model accuracy, we try to approach it from two directions:

a. The 6 layers model may be too simple for a complex data set which needs to be improved by adding more layers to make the fitting function more complex.
b. We may change the ode solver we use to try to see the difference in accuracy. To improve the accuracy when apply to actual system. We tried using the model to predict with several random images from the internet and the result is disappointed. The reason is that the image from the internet did not share the same distribution with the training data. For a more general detection model and improve its accuracy, we need to train a more widely distributed dataset, like the image net.

3. Next Steps

Our next step circles on two problems. First one is to solve the difficulties we have currently which is the accuracy of our train model and its application on more general image. We will continue working on improve the original model. As it uses a six layers architecture, we will add more non-linearity to it by adding more layers or stacking several ode blocks whichever is better on improving its' performance.

Secondly, we will apply the ode net on a new field which is video classification. During diving into the paper, the authors of the paper [1] declare OED net works well with time series dataset and cost constant memory when training without saving the intermediate variables. We believe the use of ODE net on video classification is a very state of the art. We plan to build a RCNN and ODE network applying on video data. A typical RNN looks like the figure [2] below. The intuition is to use ode solvers after every RNN block which makes use of its continuous evaluation to better fit the model.

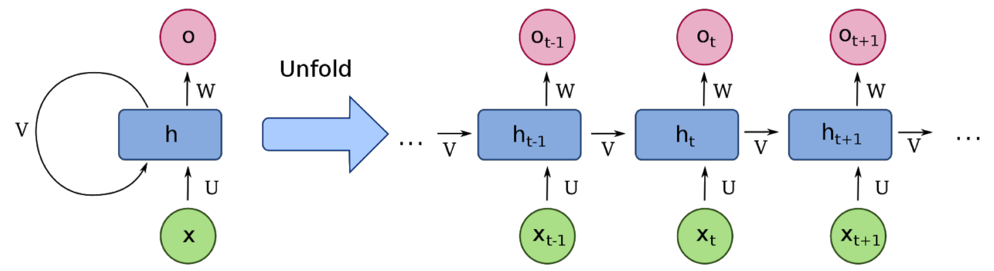


Figure 5. Basic recurrent neural network

The possible advantage of this method is saving memory during training big videos and fast detection during test. The possible disadvantage is slower training speed according to the paper. In our next step, we will use video deep learning dataset like UCF101 - Action Recognition Data Set to train.

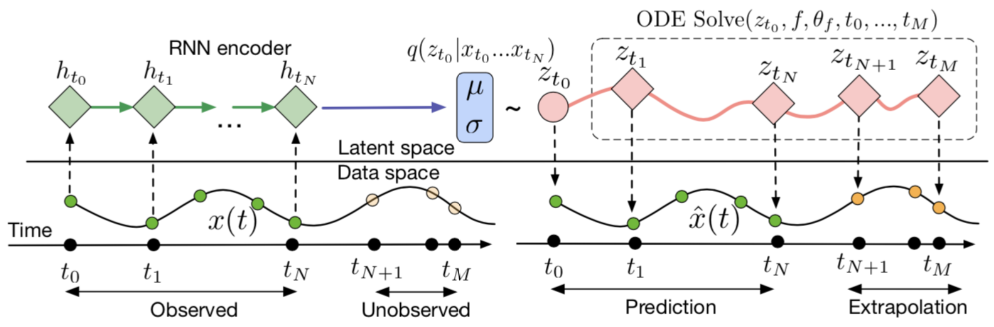


Figure 6. Left is recurrent neural network and right is ODE solver

The difficulty is that, as the figure from the paper shows, the current implementation is taking the vector at the end of the network and feeding it to the ode solver for integration. This may be sufficient for small image data because it does not have so many information. However, for big video data, the 'once for all' method may lose information because of the size of the data. This is why we need to reimplement the model to feed data to the solver 'gradually' during training.

References

1. Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. Advances in Neural Information Processing Systems, 2018.
2. https://en.wikipedia.org/wiki/Recurrent_neural_network#/media/File:Recurrent_neural_network_unfold.svg