COMP 551 - Mini-Project 4 - Group 88
Augmented Neural ODEs

Henry Ho
McGill University
260920034

Luca Zarow
McGill University
260686934

Eric Zimmermann
McGill University
260689451

Abstract

The Augmented Neural ODE (ANODE) is an extension of Neural Ordinary Differential Equations (NODE) that aim to represent and model continuous spaces. This class of networks are based on discrete problem modeling that Residual Networks accomplished. The purpose of this paper is to analyze the validity of ANODE in order to ensure that the proposed space augmentation method does in fact alleviate constrained network flows associated to current NODE model. This is accomplished by comparing baseline experiments on two image classification datasets, MNIST and CIFAR-10. MNIST is ran with an extensive hyperparameter search in order to validate the robustness of the model. ANODE provides improved accuracies for both datasets. For the MNIST dataset, the accuracy for ANODE is $97.25 \pm 0.543\%$, and for NODE is $95.57 \pm 0.323\%$, while for the CIFAR-10 dataset, the accuracy for ANODE is $52.42 \pm 1.19\%$, and for NODE is $51.72 \pm 0.571\%$. ANODE provides far fewer function evaluations compared to NODE. ANODE's improvement over NODE is also found over a wide variety of hyperparamters. With these ablations, we conclude that the model proposed by Dupont et al. [2019] is robust and is a significant improvement over previous models. Code is available at `https://github.com/LucaZarow/ReproduceANODE/`

1 Introduction

Recent works have introduced a new family of deep neural network models by investigating the relationship between differential equations and neural networks. The foundation of integrated differential equations is the Neural ODEs (NODE) model, which uses a neural network to parameterize the derivative of the hidden state of a network layer (Chen et al. [2018]). This extension has allowed for exploration of new sorts of time series modeling where networks can now restrict themselves to continuous states rather than discrete ones. Augmented Neural ODEs (ANODE) are an extension of the Neural ODEs (NODE) model, where restricted network flows are relaxed via dimension augmentation of the solution space on which the ODE is solved for (Dupont et al. [2019]). Such an extension aims to decouple a non-separable problem such that it is separable in the augmented space which results in simpler flows. (Dupont et al. [2019]).

The task of this project is to reproduce, validate and explore the robustness of the baseline ANODE model through hyperparameter tuning. The stated optimal hyperparameter are validated using the MNIST and CIFAR-10 datasets, while hyperparameter exploration is restricted to MNIST.

Through hyperparamter tuning, we compare the proposed optimized NODE and ANODE for both MNIST and CIFAR-10 for a batch size of 256 with various augmented dimensions, learning rates, tolerances, and convectional kernel filters.

2 Related Work

2.1 Residual Networks (ResNet)

ResNet is a highly efficient network in the field of computer vision and deep learning (He et al. [2015]). This model makes it possible to train thousands of layers and still achieves compelling performance. The main innovation of ResNet is to skip over some layers by reusing activations from a previous layer until the adjacent layer learns its weights (He et al. [2015]).

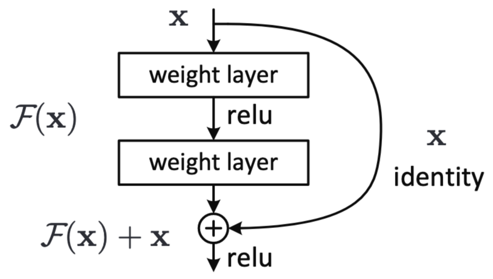


Figure 1: The “skip” strategy in ResNet (He et al. [2015]).

When implementing a typical deep neural network, vanishing gradients during back propagation may occur. The “skip connection” improvement in ResNet, shown in Fig. 1, is efficient in dealing with this issue, since it can skip through layers that it detects as non-essential during training. These skip connections also provides the discrete encoding of a function i.e. they correspond to changes in a function using discrete time steps. Solving this scheme using an ODE solver is analogous to using Euler’s method with fixed step size.

2.2 Neural ODE (NODE)

NODEs are a family of deep neural network models that are the continuous interpretation of Residual Networks (ResNets) (Dupont et al. [2019]). The model is created from the observation that the difference in the hidden state at the layers t to $t+1$ in ResNet, i.e. $\mathbf{h}_{t+1} - \mathbf{h}_t$, can be interpreted as a discretization of the derivative $\mathbf{h}'(t)$ with a timestep of $\Delta t = 1$ (Chen et al. [2018]). In the limit $\Delta t \to 0$, we get:

$$\lim_{\Delta t \to 0} \frac{\mathbf{h}_{t+\Delta t} - \mathbf{h}_t}{\Delta t} = \frac{d\mathbf{h}(t)}{dt} = \mathbf{f}(\mathbf{h}(t), t) \tag{1}$$

for some time T, where $\mathbf{x}$ is a data point. This implies the hidden state can be parameterized by an ODE. It is important to note that in NODEs, we solve an ODE starting from $\mathbf{x}$ and we adjust the internal dynamics of the system in order to appropriately map to an associated output $\mathbf{y}$ that approximates the target output $\mathbf{y}_{\text{true}}$ (Chen et al. [2018]).

The output of the network is computed using a black-box ODE solver (ex: Dormand–Prince method), while gradients are computed using the adjoint sensitivity method. The method computes gradients by solving a second, augmented ODE backwards in time (Chen et al. [2018]) This approach significantly reduces the memory cost from backpropagation.

A large contribution of the NODE model is the ability to model time series in a generative approach. Each time series is represented with a latent trajectory (Chen et al. [2018]). The model has shown promising results when modeling time series with irregular time points. For example, it achieved a root-mean-squared-error (RMSE) of 0.1346, as opposed to RMSE of 0.1813 by using RNN, on 100 randomly sampled points from each trajectory without replacement (Chen et al. [2018]). The ability to model time series accurately may allow NODEs to help mitigate the difficulty in handling missing data in irregularly-sampled data such as medical records.

2.3 Augmented Neural ODE (ANODE)

The ANODE model increases the dimension of the space in which the ODE is learned and solved in an attempt to alleviate restricted flows in the specified solution space, i.e. from $\mathbb{R}^d$ to $\mathbb{R}^{d+p}$. The ODE flow will project points onto these additional dimensions, which prevents intersecting trajectories. The augmented ODE problem can be formulated as (Dupont et al. [2019]):

$$\frac{d}{dt}\begin{bmatrix}\mathbf{h}(t)\\ \mathbf{a}(t)\end{bmatrix} = \mathbf{f}(\begin{bmatrix}\mathbf{h}(t)\\ \mathbf{a}(t)\end{bmatrix}, t), \qquad \begin{bmatrix}\mathbf{h}(0)\\ \mathbf{a}(0)\end{bmatrix} = \begin{bmatrix}\mathbf{x}\\ \mathbf{0}\end{bmatrix} \tag{2}$$

where $\mathbf{a}(t) \in \mathbb{R}^p$ is a point in the augmented space. The data point $\mathbf{x}$ is concatenated with a vector of zeros to match the dimension of the augmented space. With the extra dimensions, ANODE does not need to continuously deform the input space, allowing points in the input space to be mapped to reasonable values. It produces flows that are simpler for the ODE to solve.

For image datasets, the input $\mathbf{x}$ is an image with width w and height h. So the hidden state $\mathbf{h}(t)$ will be in $\mathbb{R}^{c\times h\times w}$, where c denotes the number of channels. For images, we augment the space from $\mathbb{R}^{c\times h\times w}$ to $\mathbb{R}^{(c+p)\times h\times w}$, where we add p channels of zeros to the input image. Image experiment are conducted using convolutional architectures for $\mathbf{f}(\mathbf{h}(t), t)$ (Dupont et al. [2019]).

3 Dataset and setup

3.1 MNIST

MNIST is a large grayscale image database of handwritten digits, each of size 28 $\times$ 28. It is a subset of a larger dataset NIST, where it is constructed from NIST's Special Database 3 (SD-3) and Special Database 1 (SD-1). Both Special Databases contain 1-bit (binary) images of handwritten digits. SD-3 was collected among Census Bureau employees, while SD-1 was collected among high-school students. The MNIST training set consists of 30,000 patterns from SD-3 and 30,000 patterns from SD-1, while the MNIST test set consists of 5,000 patterns from SD-3 and 5,000 patterns from SD-1. The digits are size-normalized and centered in a fixed-size image (LeCun et al.).

3.2 CIFAR-10

The CIFAR-10 dataset consists of 60,000, 32 $\times$ 32 colour images and 10 classes, with 6000 images for each class. There are 50,000 training images and 10,000 test images. The 10 classes are $\{$ airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck $\}$. It is important to note that the classes are mutually exclusive, i.e. there are no overlaps between different classes. The dataset is divided into 5 training batches and 1 test batch. Each training batch contains 10,000 images and the test batch contains 1000 randomly selected images from each class (Krizhevsky [2010]).

4 Proposed Approach

4.1 NODE And ANODE

The ANODE model is re-implemented and adapted as stated in (Dupont et al. [2019]). Both NODE and ANODE are constructed using convolutional blocks as a basis for image processing task. Such models are then tuned on an array of hyperparameters in an attempt to reproduce the results stated in the paper.

The re-implemented model is formed as a convolutional ODE fed to a fully connected linear section layer. This construction, shown in figure 2, follows the construction of the NeurIPS paper.

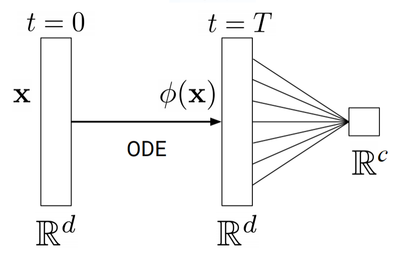


Figure 2: Architecture of the Neural ODE model (Dupont et al. [2019]).

In the model, the time dependent ODE ϕ (x) maps the d-dimensional input from time 0 to time T. The linear layer maps the d-dimensional ODE solution to a c-dimensional output. For classification on MNIST and CIFAR-10, d is ((number of channels + number of augmented dimensions) $\times$ height $\times$ width) of image, and c is the number of categories, which is 10 in both cases.

The ODE section of the model contains the ODE solver used to to solve the function on which the task is based, and an initial value. Since the task is categorized based on an image database, the function is a modified version of convolution, and our initial value is the dataset. The ODE solver used in this model comes from the `torchdiffeq` library (Chen et al. [2018]). As a baseline, the error tolerance of 1e-3 and a (fixed) maximum step of 1000 is used. This maximum step of 1000 is set to timeout the program due to too many function calls. The solver uses an the adaptive step method of Runge-Kutta (4,5) (Schober et al. [2014]), as defined by the library.

Depending on whether this instance of the model represents a NODE or an ANODE, the dataset may be modified. In the case of NODE, the dataset is unmodified, but in the case of ANODE, the dimension of the dataset is augmented according to the number of augmented dimensions fed into the model. These extra dimensions simply represent the augmented space that the ODE will be solved in, and so is initialized with zeros. In the case of convolution, we augment the space by dimensions specified as a hyperparameter.

Once the input to the solver has been prepared, the solver must also be fed a function. We define a modified version of a Convolutional Neural Network (CNN). The modification comes from the convolution itself, which in our model is time dependent. The time dependent convolution adds an extra channel to the input, representing time. In the forward pass, both the image data and time are used as input. Time is concatenated along the image and fed into a standard 2D convolutional node with a predefined kernel size, stride and padding. The function uses 3 time-dependent convolutions with a ReLU layer between each inner layer. The structure of our ODE function is shown in figure 3.

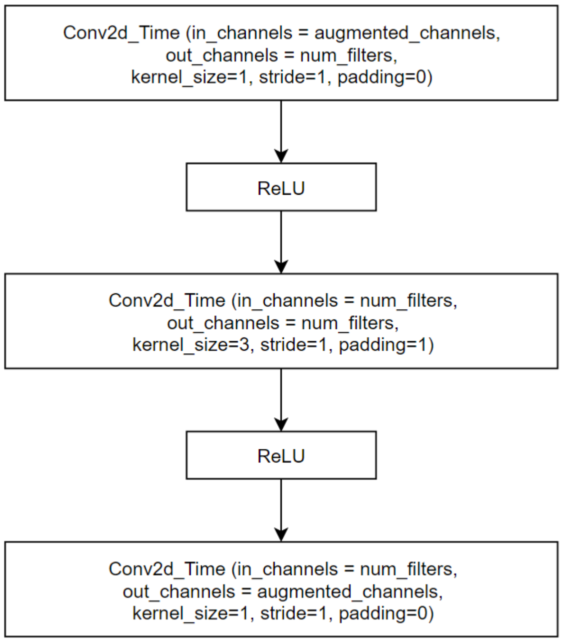

Conv2d_Time (in_channels = augmented_channels,
out_channels = num_filters,
kernel_size=1, stride=1, padding=0)
ReLU
Conv2d_Time (in_channels = num_filters,
out_channels = num_filters,
kernel_size=3, stride=1, padding=1)
ReLU
Conv2d_Time (in_channels = num_filters,
out_channels = augmented_channels,
kernel_size=1, stride=1, padding=0)

Figure 3: Architecture of our convolutional ODE function.

Here, the convolutional kernels are of size 1x1, 3x3, 1x1, augmented channels is the number of channels in the image + the number of augmented dimensions, which is 0 for NODE greater than 0 for ANODE. These are hyperparameters to be validated.

4.2 Training

Each model is trained on MNIST and CIFAR-10 for a total of 3 runs. All models are tuned using 3-fold cross validation and a cross entropy loss over a maximum of 10 epochs with an early stopping criterion that selects for maximum validation score. A batch-size of 256 is used for all tests and is not considered during hyperparameter tuning. Class scores are predicted by taking the $argmax$ of the softmax given the output of the model.

It should be noted that no transformations were applied to either dataset as preprocessing. In practice, classification performance of MNIST digits and CIFAR-10 images are both known to improve by normalizing image intensities between [0,1] as well as applying simple transformations to the image.

4.3 Optimization

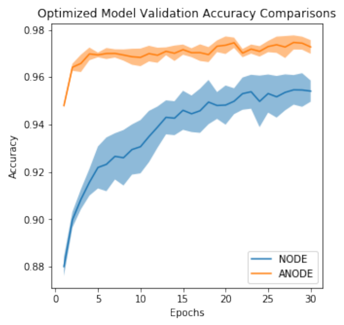
Each model is optimized using a combination of the Adaptive Moment Estimation (Adam) optimizer (Kingma and Ba [2014]), and the built-in adjoint sensitivity method of back propagation provided as a library function call. Adam has an adaptive learning rate according to the gradient momentum and is used to update modal parameters. It should be noted that all experiment results are displayed without the use of weight-decay. Moreover, experimentally, models trained without the adjoint sensitivity method had a noticeable increase in performance (CIFAR-10: 10%, MNIST: 2%). These results are not displayed.

5 Results

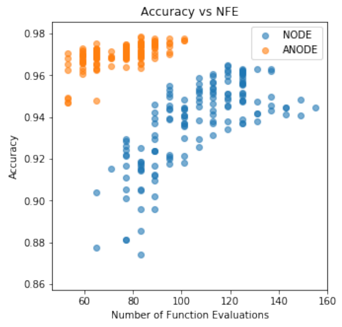
5.1 Baselines

5.1.1 MNIST

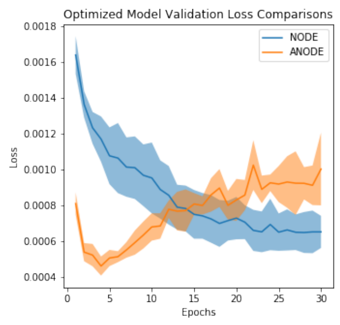
A NODE and ANODE model are both trained on the MNIST dataset for a total of 30 epochs. For each model, a batch size of 256, learning rate of 1e-3, tolerance of 1e-3 are used throughout both experiments. ANODE has an augmented dimension of 5 and 64 hidden state filter, whereas NODE has 92 hidden state filters.



(a) NODE vs ANODE Validation accuracy for MNIST



(b) NODE vs ANODE Accuracy vs NFE for MNIST

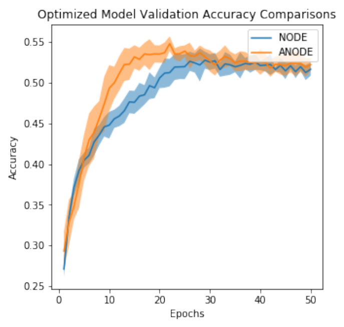


(c) NODE vs ANODE Validation loss comparison for MNIST

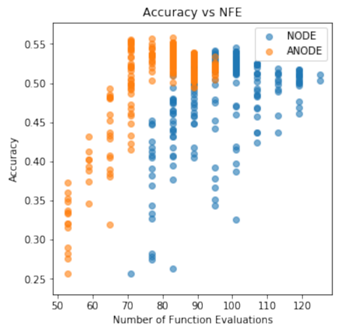
The final reported accuracy for ANODE is $\mathbf{97.25 \pm 0.543\%}$ and is NODE is $\mathbf{95.57 \pm 0.323\%}$. The difference between the two is significant. The number of function calls for ANODE is observed to be significantly less than that of NODE and trains exponentially quicker. To support this, the function loss decreases much faster than that of NODE. As a whole, this system significantly improved the network as well as training time. Overall results are inline with the suggested results.

5.1.2 CIFAR-10

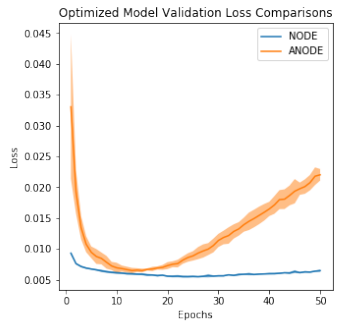
A NODE and ANODE model are both trained on the CIFAR dataset for a total of 50 epochs. For each model, a batch size of 256, learning rate of 1e-3, tolerance of 1e-3 are used throughout both experiments. ANODE has an augmented dimension of 10 and 64 hidden state filter, whereas NODE has 125 hidden state filters.



(a) NODE vs ANODE Validation accuracy for CIFAR-10



(b) NODE vs ANODE Accuracy vs NFE for CIFAR-10



(c) NODE vs ANODE Validation loss comparison for MNIST

The final reported accuracy for ANODE is $\mathbf{52.42 \pm 1.19\%}$ and is NODE is $\mathbf{51.72 \pm 0.571\%}$. The difference between the two is not significant, however, the model performs much better on validation and achieves notable performance much faster than NODE. As follows the results in MNIST, the number of function executions is significantly reduced, but loss does not reach a much lower minimum value,. Training time is significantly reduced. Overall results were not perfectly inline with the suggested results.

5.2 Hyperparameter Tuning

Hyperparameters are tuned for ANODE in order to verify the robustness of the model as well as the general reliability. Since there are max iteration executions and tolerances that specify convergence of an ODE, it must be confirmed that the augmentation does not destroy performance is spaces where parameters and flows are sub-optimal.

The specified hyper parameters that re tuned for are listed as follows with a given grid to be searched:

- Number of Filters: 32, 64, 92
- Augmented Dimension: 3, 5, 8
- Tolerance : 1e-2, 1e-3
- Learning Rate: 1e-2, 1e-3

In total, there are a total of 36 combinations of hyperparameters that are tested. A detailed list of the results are found in the Appendix 1. Observations show the the model is robust and operates efficiently for all sets of filters and augmentation spaces as long as the learning rate is below 1e-2. Moreover, a higher tolerance did not impact the performance of the model whatsoever, it simply made experiments converge with fewer number of executions. This demonstrates that the impositions of additional dimensions directly contributes to better network flows, regardless of its value above a certain threshold. Additions of too may dimensions and/or filters simply reduced the speed at which the model trains.

5.3 Modified Baseline

A challenge with the proposed approach for vision tasks is with respect to the constraints that an ODE places on the system. For the ODE solver to be used, the dimensions cannot be downsampled which is a critical component required for vision tasks. The first modification done is with respect to the convolutions. The scheme is 1x1, 3x3, 1x1. This is extended to be three sets of 3x3 filters instead. Rather than flatten the output directly, a global average pooling (GAP) layer is used. It is theorized that the capacity increase from additional convolutions provide a means of extracting more meaningful and useful features while the GAP layer allows the model to become more invariant to spatial factors. Moreover, a global average pooling layer further reduces the number of required parameters on the output making the model easier to train and requires less time. It should be noted however, that this method requires large augmented dimensions in order to perform inference since the output is pooled over the augmentation dimension plus 1. The reported final accuracy for this model is 97.41 % using 64 filters, 12 augmented dimensions, tolerance of 1e-3, learning rate of 1e-3, batch-size of 256, and 25 epochs. Batch-size is kept consistent for reliable comparisons.

6 Discussion and Conclusion

It is concluded that the ANODE implementation does indeed alleviate restricted flows in image analysis by augmenting the solution space to a higher dimension (analogous to a kernel trick). This is confirmed by reduced execution calls, faster training times, minimized loss, and improved accuracy. These optimizations have been verified to hold and are robust for a variety of model hyperparameters. Regardless of the number of filters or augmented dimension (greater than a min threshold), ANODE is able to be perfectly tuned to provide considerably better results than its predecessor NODE. These results demonstrate that models with comparable number of parameters (NODE vs ANODE), ANODE is able to utilize its weights more efficiently seen through the latter minimization of loss, function calls, time to train and maximization of accuracy. Although time series problems were not explored in this study, there seems to be a lot of promise with this technology and the inclusion of continuous flow modeling is the first step forwards.

7 Statement of Contributions

Henry Ho:

- Researched on NODE and ANODE and also the datasets.
- Wrote the Abstract, Introduction, Related Work and Dataset and Setup sections of the report.
- Formatted and edited the report.

Luca Zarow:

- Models
  - NODE
  - ANODE
  - Convolutional ODE Function
- Training pipeline
  - Baseline
- Report
  - Writing Introduction, Related Work and Proposed Approach
  - Editing

Eric Zimmermann:

- Training pipeline
  - Loaders
  - Metric logging
  - Automated training and testing
  - Modified baseline
- Hyperparameter search
  - Configurations
  - Automated search
  - Automated analysis
- Metrics
  - Plots
  - Misc analysis
  - Writing Proposed Approach, Results, and Discussion and Conclusion
  - Editing
- Modified Baseline

References

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018.

E. Dupont, A. Doucet, and Y. W. Teh. Augmented Neural ODEs, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition, 2015.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2014.

A. Krizhevsky. Convolutional Deep Belief Networks on CIFAR-10, 2010.

Y. LeCun, C. Cortes, and C. J. Burges. THE MNIST DATABASE of handwritten digits.

M. Schober, D. Duvenaud, and P. Hennig. Probabilistic ODE Solvers with Runge-Kutta Means, 2014.

8 Appendix

Experiment #	# Filters	Augmented Dimension	Tolerance	Learning Rate	Accuracy (%)
1	32	3	1e-3	1e-3	96.74
2	64	3	1e-3	1e-3	96.83
3	92	3	1e-3	1e-3	97.13
4	32	5	1e-3	1e-3	97.13
5	64	5	1e-3	1e-3	97.20
6	92	5	1e-3	1e-3	97.05
7	32	8	1e-3	1e-3	97.01
8	64	8	1e-3	1e-3	97.14
9	92	8	1e-3	1e-3	96.72
10	32	3	1e-3	1e-2	96.84
11	64	3	1e-3	1e-2	96.79
12	92	3	1e-3	1e-2	97.10
13	32	5	1e-3	1e-2	96.84
14	64	5	1e-3	1e-2	96.85
15	92	5	1e-3	1e-2	97.28
16	32	8	1e-3	1e-2	97.15
17	64	8	1e-3	1e-2	96.86
18	92	8	1e-3	1e-2	97.01
19	32	3	1e-3	1e-2	95.48
20	64	3	1e-3	1e-2	95.67
21	92	3	1e-3	1e-2	95.53
22	32	5	1e-3	1e-2	95.90
23	64	5	1e-3	1e-2	95.81
24	92	5	1e-3	1e-2	96.08
25	32	8	1e-3	1e-2	95.46
26	64	8	1e-3	1e-2	95.73
27	92	8	1e-3	1e-2	96.18
28	32	3	1e-2	1e-2	94.83
29	64	3	1e-2	1e-2	95.61
30	92	3	1e-2	1e-2	95.47
31	32	5	1e-2	1e-2	94.32
32	64	5	1e-2	1e-2	94.58
33	92	5	1e-2	1e-2	91.42
34	32	8	1e-2	1e-2	93.84
35	64	8	1e-2	1e-2	94.31
36	92	8	1e-2	1e-2	94.53

Table 1: Table of hyperparameters for MNIST ANODE and reported accuracy for each experiment. It can be observed that as the tolerance decreases, the accuracy deteriorates, with the accuracy being as low as 91%.