Traffic Sign Classification Using ODENet

Yaratapalli Nitheesh Chandra Sainath(✉), Reethesh Venkataraman, Abhishek Dinesan, Ashni Manish Bhagvandas, and Padmamala Sriram

Department of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India
nitheesh.my@gmail.com, reetheshv.rv@gmail.com, abhishekdinesh3@gmail.com, ashnibhagvandas16@gmail.com, padmamala@am.amrita.edu
https://www.amrita.edu

Abstract. In the family of deep neural network models, deeper the model is, the longer it takes to predict and larger the memory space it utilizes. It is very much likely that use-cases have constraints to be respected, especially on embedded devices, i.e, low powered, memory-constrained systems. Finding a suitable model under constraints is repeated trial-and-error to find optimal trade-off. A novel technique known as Neural Ordinary Differential Equation Networks (ODENet) was proposed in NeurIPS2018, where instead of a distinct arrangement of internal hidden layers of a Residual Neural Network (RNN), they used parametrized derivatives of internal states in the neural system. Any differential equation solver can be used to calculate the final output. These models have constant depth and can trade between speed and accuracy. We propose a methodology for Traffic Sign Detection using ODENet and subsequently conclude that ODENets are more robust and perform better in comparison to ResNets. We also conclude that though training time is high in ODENets, they can trade-off between speed and accuracy when it comes to both training and testing.

Keywords: Differential equations · Neural networks · Embedded systems · Traffic sign · Deep learning

1 Introduction

1.1 Traffic Sign Detection(TSD)

An innovation by which a vehicle can discern the road signs that are placed on sides of the road such as, “Speed Limit”, “No Parking” or “No U-Turn”. It is being innovated by various self-driving automotive manufacturers. It can use various techniques ranging from image processing to Lidar analysis to locate and identify traffic signs. The techniques can be commonly grouped based on the color, shape and the type of learning [5,7].

TSD is a real world task which involves lot of constraints and complications. Even a minor misclassification of the traffic sign could lead to catastrophic outcomes and can even lead to loss of life. This is implemented as a sub system in various ADAS and in autonomous vehicles [13].

S. M. Thampi et al. (Eds.): SoMMA 2020, CCIS 1366, pp. 187–197, 2021.
https://doi.org/10.1007/978-981-16-0419-5_15

Typically, a camera facing the forward direction will be present on the dashboard of the vehicle and it captures the real time video feed which is sampled into frames and fed to a series of methods and techniques. One of which is a deep learning model which is deployed inside an automotive embedded board. As the vehicle is driven in various environments, lighting conditions, speeds and geographies it is essential for the deep learning algorithm to be robust and reliable at all times. The camera can catch the traffic sign in different orientations and positions but the algorithm should recognize the correct sign [13].

Detection and classification are the 2 main parts of the algorithm. The detection module takes the image and localizes any traffic signs in them after which the classification module identifies which sign it is. Also, both modules work based on the colour, shape and edges of traffic signs. There are ways to determine both detection and classification in the same technique [18].

1.2 Neural Networks as Differential Equations

All different variants of residual networks (ResNets) create complicated changes to hidden state with each transformations:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \Theta_t) \tag{1}$$

where $t \in \{0...T\}$ and $h_t \in \mathbb{R}^D$.

Ordinary differential equation can be used as a substitute for 1:

$$\frac{\mathrm{d}\mathbf{h}(\mathrm{t})}{\mathrm{d}t} = f(\mathbf{h}(t), t, \theta)) \tag{2}$$

The initial layer $\mathbf{h}(0)$ and designate the output layer $\mathbf{h}(T)$ is the result of this initial value problem at a certain depth T. By utilizing any differential equation solver, referred as black-box solver in [7] that computes the concealed dynamics f. These solvers can consists of tunable parameters which in this case are used for speed-accuracy trade off.

For other advantages of NeuralODEs, refer [7].

2 Related Work

2.1 Traffic Sign Detection (TSD)

Can be broken down to subsets which include tasks like detection, classification and tracking. There has been extensive research effort carried by researchers in this area.

Computer Vision Feature Extraction. Computer vision algorithms and methodologies were few of early approaches before the advent of machine learning. Techniques like Histogram Oriented Gradients (HOG) [12] is initially popularized for the detection of pedestrians. In this method the usage of color gradients in images are computed along with various weighted, normalized histograms.

Scale Invariant Function Transform, popularly know as SIFT [10] was used to classify and the sliding window method was used simultaneously to perform both classification and detection tasks.

Machine Learning. Machine learning algorithms uses many techniques namely Support Vector Machines [14], Linear Discriminant Analysis [17], Ensemble Classifiers, Random Forest, and KD-Trees [23].

Linear Discriminate Analysis (LDA) is based on the maximum posterior estimation of the class membership. Class densities are assumed to have multivariate Gaussian and common co-variance matrix [21].

Random Forest is an ensemble classifier method [20] which is a set of decision trees. Each decision tree is trained using randomly taken training data. Testing data is analyzed by all different decision trees for classification. The classification output is based on majority voting, which takes into account the majority decision trees' decisions.

Support Vector Machines (SVMs) are classification algorithms that use a hyperplane to divide the n-dimensional data [14]. SVM can also isolate data distributed non-linearly by projecting the classification hyperplane into higher dimensions using a non-linear kernel function.

Machine learning approaches [1] were unable to manage images of various aspect ratios, sizes and dimensions and had to be manually produced which is a very time-consuming operation that can produce a lot of errors.

Deep Learning. To overcome the disadvantages of above mentioned conventional methodologies new implementations based on deep learning algorithms which presented a novel approach than previous methods. In recent years with an increase in computing power and availability of standardized data sets and access to a huge amount of data.

LENET Architecture was the first CNN architecture for traffic sign classification. Convolutional neural networks are multi-stage neural network architecture that learns the invariant features automatically [6]. Each stage consists of convolution layer, a RELU layer, and a pooling layer [19]. The pooling layer lowers spatial information and functions as complex cells within the visual cortex. A gradient descent based optimizer is used for training and updating each filter to minimize the loss function. The output of all the layers is fed to the classifier for improving the accuracy of classification.

2.2 Neural Ordinary Differential Equation

There was no practical application of adjoint training approach for continuous-time neural networks, even though it was proposed [8]. The concept of exploiting reversibility and approximate computation came into research after viewing residual networks [4] as an approximate ODE solver [24]. The different sections of the said concepts that came under research were adaptive computation (ODE solvers give generalizable and resource and CPU non-intensive solutions to adjust the computation number) [3], constant memory backpropagation

(reversible resnets made of restricted architectures with the same memory workload as normal approach, except our approach does not have such restricted architectures) [2], approximate differential equations from given data (fluid simulation applications are one such example) [22] and using adjoint sensitivity analysis which uses linear-time number of variables in comparison to forward analysis which is quadratic-time [11].

3 Reverse-Mode Automatic Differentiation of ODE Solutions

A package provided by the authors of Neural Ordinary Differential Equations at https://github.com/rtqichen/torchdiffeq.git [3] is being used with Scikit-learn [15].

In training continuous-depth networks, the key technological challenge is to execute backpropagation via the ODE solver. In forward pass operations, differentiating is easy, but it involves a high processing cost and introduces unnecessary computational error.

By using a BlackBox ODE solver, gradients are calculated using *the adjoint sensitivity technique* [16]. This method measures gradients in time, solving an increased ODE backwards. This solving is valid to all ODE solvers. This method also has low memory expenses, scales and problem size are linearly proportional and manages computational error directly.

$$\mathbf{L}(\mathbf{z}(t_1)) = \mathbf{L}\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt\right) \tag{3}$$

$$= \mathbf{L}\left(ODESolve(\mathbf{z}(t_0), f, t_0, t_1, \theta)\right) \tag{4}$$

where $\mathbf{L}()$ is ascalar-valued loss function and input is the result of an ODE solver:

Gradients corresponding to θ are required to optimize L. In the first step, the dependency between the hidden state $\mathbf{z}(t)$ and the gradient of loss is determined. Also known as the Adjoint $a(t) = \frac{\partial L}{\partial \mathbf{z}(t)}$. Another Ordinary Differential Equation gives the dynamics:

$$\frac{\mathrm{d}\mathbf{a}(t))}{\mathrm{d}t} = -\mathbf{a}(t)^T \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}} \tag{5}$$

Common ODE solvers can output the state $\mathbf{z}(t)$ at any point. The reverse-mode derivative is seperated inot multiple solutions each bearing the middle of 2 consequtive output times as the loss depends on the in between states. At each iteration, the Adj (Adjoint) matrix will be updated in the direction of the respective partial derivative $\frac{\partial L}{\partial \mathbf{t}_i}$. Full derivation can be found in [7].

3.1 ODE-Nets: Error Control

Tolerance of the true solution can be ensured by setting parameters for ODE solvers. Tolerance has direct effect on the behavior of the network. Also the number of feature evaluations is proportional to the amount of time.

Adjusting the tolerance value allows trade-off between accuracy and computational costs. Training for sensitive changes can be done with high precision and test the same system with test will lower tolerance as a trade-off for speed and vice-versa.

3.2 Network Depth

As of now, number of internal evaluations (depends on the initial state) for hidden state dynamics can be used as depth for an ODE solution, but the term is still unclear.

4 Experimental Setup

4.1 Dataset

The GTSRB dataset (German Traffic Sign Recognition Benchmark) [20] is a multi-class classification dataset accumulated as part of IJCNN 2012. It has over 50000 images and 43 different classes. The annotations are given in CSV files with details such as: height and width of the image, bounding box co-ordinates, filename and class label of the traffic sign. A more in-depth review of the dataset was worked upon in Wen Lihua et al. 2017 [9].

4.2 Augmentation and Preprocessing Techniques

Resizing all images to the most prominent width, height and depth of the dataset i.e, $(33 \times 34 \times 3)$. Also, the dataset is unbalanced which could be handled by using some augmenting techniques with rules like

- Allowed rotation: $\pm 25°$ more than this could cause changes to certain samples
- Horizontal shift: $\pm 20\%$
- Vertical shift: $\pm 20\%$, most of the image has to be retained
- Horizontal flip: $False$
- Vertical flip: $False$, As flipping could change the sign
- Color space transformation: ± 0.1, as signs are subject to lighting under different seasons
- Scale in: 0.5
- Scale out: 2.0, to make predictions size invariant
- crop: Allowed with a min height or width as at least 40%
- Noise Injection: Gaussian or Salt and Pepper, adding noise to prevent adversarial attacks

4.3 Additional Augmentation Techniques

In the real-world, data can exist in a variety of unpredictable circumstances which can not be accounted for by the simple methods described above. Conditional GANs have the power to generate images with variations in the wild with input images.

The above method is robust but computationally intensive. Anything called neural design transfer would be a cheaper option. This takes one image's texture/ambience/appearance (aka, the "style") and combines this with another's material.

The only downside to this approach is that rather than practical, the performance appears to look more artistic. However, there are other advancements that have amazing effects, such as Deep Picture Style Transfer, shown below.

4.4 Brief on Interpolation

In general, the original image size has to be maintained after performing the above transformations. Our picture has no knowledge about what's outside its borders. In general, the space outside the boundary of the image is zeros. Therefore you get a black area when you do these transformations where the image is not specified.

But the images do not have a black colour group of pixels which may not be what is desired. There are various algorithms both in image processing and Machine Learning algorithms to fill the missing padding or empty space:

- **Constant** A simplest gap filler would be to fill the empty part with a fixed value.
  This works monochromatic images but not for images but does work for.
- **Reflect** Image values will be reflected along the missing values from the boundary of the image.
  Useful for natural or continuous contexts.
- **Edge** The image's edge values are extended beyond the boundary.
  For mild translations, this method can work
- **Symmetric** Similar to reflecting, despite the fact that a copy of the edge pixels is rendered at the boundary of reflection.
  Noticeable when dealing with very small scale patterns or images.
- **Wrap** The picture is only replicated as if it were tiled outside its limits.
  This does not make sense for a lot of scenarios, not as popularly used.

Custom methods can be used for handling missing values. For most classification problems, these methods will typically do well.

4.5 Deep Learning Techniques

Choosing an impainting approach is both temporally consistent and maintains a clear object boundary.

4.6 Network Architecture

The architecture used for the TSD consists of 2 convolutional layers as with 64 and 128 filters respectively. Followed by an ODE Block with two convolutions with a flatten and a dense layer for output. Above mentioned each convolution is a group of Convolution, Batch Normalization, MaxPooling.

Input Layer. Input Layer is used to instantiate a tensor. This layer takes the images as input thus the size will be $(33 \times 34 \times 3)$ to take RGB image as input.

Convolutional Layer. The 2D convolution layer is the most common form of convolution used and is typically abbreviated as conv2D. In a conv2D layer, a filter or kernel has a height and a width. It is usually smaller than the image input and so it is transferred over the entire image. The region in which the filter appears on the image is called the receptive field.

Every filter in this layer is initialized randomly into some distribution (Normal, Gaussian, etc.). Each filter is trained slightly differently by having different initialisation criteria. Eventually, they learn how to detect various features in the image. Unless they both were equally configured, then the odds of two filters learning similar features are significantly rising. Random initialization helps each filter to learn to recognise different characteristics.

Since each conv2D filter learns a separate feature, many of them are used to identify different features within a single layer. The best part is that every filter is learnt automatically. Here a kernel size of 3×3 is used along with 64, 128 filters in conv_1 and conv_2.

Pooling Layer. Its purpose is to gradually bring down the spatial size which in turn allows to decrease trainable parameters, thereby decreasing network computation. Pooling layer functions independently on every function diagram.

Max and average pooling are the most common methods employed in pooling.

Batch Normalization Layer. By modifying and scaling the activations, the input of a layer is normalized. This allows all layers to work independently.

It eliminates overfitting, since it has a small effect of regularization. Hence, less dropout is used along with batch normalization, which is a good thing since the loss of details is minimised. Nonetheless, for regularization, batch normalization cannot be solely relied on; its better to be used along with dropout.

Dropout Layer. On a neural network, Dropout is implemented per row. Almost all layers can be used with a dropout layer. Dropout parameter specifies probability of an output retaining for next layer. Generally a probability of 0.5 is used.

ODEBlock Layer. A chain of residual blocks in a neural network is basically a solution of the ODE with the Euler method as

$$y_{n+1} = y_n + f(t_n, y_n) \tag{6}$$

in this case, the the system's initial condition is "time" 0, which indicates the very first layer of the neural network, and as x(0) will serve the normal input, which can be time series, image, whatever you want! The final condition at "time" t will be the desired output of the neural network: a scalar value, a vector representing classes or anything else.

These residual connections are discrete time steps of the Euler method, which means that the depth of the neural network can be regulated just by choosing the discretizing scheme, hence, making the solution (aka neural network) more or less accurate, even making it a pseudo-infinite-layer.

Flatten Layer. The spatial dimensions of the input collapse into the channel dimension by a flattening layer.

Dense Layer. Every node in this layer is connected to every other node in the previous layer. Also know as fully connected layer. In this case, a dense layer with 43 nodes is used for output.

5 Results

The model is compared with ResNet, a similar network. The model is evaluated using the testing data set of 12,630 testing images (Table 1).

$$Accuracy = \frac{\Sigma CorrectlyIdentified}{Totalimages} \tag{7}$$

Table 1. Comparison of results

Model	Loss (training)	Accuracy (training)	Loss (testing)	Accuracy (testing)	Best epochs
ResNet	0.1066	0.9689	0.7283	0.8046	15
ODENet*	0.0100	0.9968	0.1990	0.9561	17

*Our approach

Training was done using 128 batch size, and categorical cross entropy was used as the loss function. Adam optimizer was been used.

Based on the loss and accuracy metrics of the best epoch, it is concluded that ODENets perform better in comparison to ResNets. It is observed that the number of epochs (iterations) for ODENets is higher, proving that ODENets take longer to converge.

6 Conclusion

Traffic sign detection is a challenging task as they are implemented on a embedded device, thus having a constraint on processing power. By using Neural Ordinary Differential Equation techniques, a trade-off between speed and accuracy of the system can be facilitated to correctly perform image classification and recognition tasks faster, even on blurred, rotated and distorted images. Also as concluded in the previous section, even though training time is longer than in other similar methods, ODENets provide a flexible and faster way of getting results.

7 Future Work

7.1 Minibatching

Usage of mini-batches do not currently have standardization as a regular neural networks. Evaluations are batched together through the differential solver by joining states of batches together to create a combined ODE with a dimension of value **D**x**K**. For a case of different errors for each batch, a combined error is required, which is K times more system intensive to solve individually. But, no substantial change was noticed in number of evaluations during the practical usage of mini-batches.

7.2 Setting Tolerances

The usage of adaptive solver for ODEs has a trade off between precision vs speed, but this requires the manual setting of tolerances for both forward and backward passes in training. The tolerance was brought down to $1e-3$ and $1e-5$ for atol and rtol respectively without any changes in performance.

7.3 Functions Neural ODEs Cannot Represent

Classes of functions are introduced in arbitrary dimension d which NODEs cannot represent. Let $0 < r1 < r2 < r3$ and let $g : \mathbb{R} \rightarrow \mathbb{R}$ be a function such that

$$\begin{cases} g(\mathbf{x}) = -1 & if \|x\| \leq r_1 \\ g(\mathbf{x}) = 1 & if r_2 \leq \|x\| \leq r_3 \end{cases} \tag{8}$$

References

1. Andreas Mogelmose, M.M.T., Moeslund, T.B.: Vision-based traffic sign detection and analysis for intelligent driver assistance systems: perspectives and survey. IEEE Trans. Intell. Transp. Syst. **13**(4), 1484–1497 (2012)

2. Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., Holtham, E.: Reversible architectures for arbitrarily deep residual neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
3. Chen, T., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: Advances in Neural Information Processing Systems, vol. 31, pp. 6571–6583. Curran Associates, Inc. (2018). http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf
4. He, R., Lin, C., Wang, J., McAuley, J.: Sherlock: sparse hierarchical embeddings for visually-aware one-class collaborative filtering (2016)
5. Hecht, J.: Lidar for self-driving cars. Opt. Photon. News **29**(1), 26–33 (2018)
6. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition. Neural Netw. **32**, 323–332 (2012)
7. Lim, K., Hong, Y., Choi, Y., Byun, H.: Real-time traffic sign recognition based on a general-purpose GPU and deep-learning. PLOS One **12**(3), 1–22 (2017). https://doi.org/10.1371/journal.pone.0173317
8. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient backpropagation. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 9–48. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_3
9. Lihua, W., Jo, K.H.: Traffic sign recognition and classification with modified residual networks. In: 2017 IEEE/SICE International Symposium on System Integration (SII), pp. 835–840, December 2017
10. Lowe, D.G.: Object recognition from local scale-invariant features. In: International Conference on Computer Vision, vol. 2, p. 1150. Computer Society, Washington (1999)
11. Melicher, W., Uet al.: Fast, lean, and accurate: modeling password guessability using neural networks. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 175–191. USENIX Association, Austin, August 2016. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher
12. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886–893, June 2005
13. Deepika, N., Variyar, V.: Obstacle classification and detection for vision-based navigation for autonomous driving. In: International Conference on Advances in Computing, Communications, and informatics (ICACCI)(2017), pp. 2092–2097 (2017). https://www.semanticscholar.org/paper/Obstacle-classification-and-detection-for-vision-Deepika-Variyar/1100b4de94fc4ca6307c09d57901d52a01e18b74
14. Park, J.G., Kim, K.J.: Design of a visual perception model with edge-adaptive gabor filter and support vector machine for traffic sign detection. Expert Syst. Appl. **40**(9), 3679–3687 (2013)
15. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
16. Pontryagin, L., Lohwater, A.: Ordinary differential equations (1962). https://cds.cern.ch/record/113444
17. Rabia Malik, J.K., Ahmad, S.N.: Road sign detection and recognition using color segmentation, shape analysis, and template matching. In: International Conference on Machine Learning and Cybernetics, vol. 6, pp. 3556–3560. IEEE (2007)
18. Rani, N.S., Rao, P., Clinton, P.: Visual recognition and classification of videos using deep convolutional neural networks. Int. J. Eng. Technol. (UAE) **7**, 85–88 (2018)

19. Sermanet, P., LeCun, Y.: Traffic sign recognition with multi-scale convolutional networks. In: IJCNN, pp. 2809–2813. IEEE (2011)
20. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German traffic sign recognition benchmark: a multi-class classification competition. In: The 2011 International Joint Conference on Neural Networks (IJCNN), pp. 1453–1460. IEEE (2011)
21. Wu, Y., Liu, Y., Li, J., Liu, H., Hu, X.: Traffic sign detection based on convolutional neural networks. In: The 2013 International Joint Conference on Neural Networks (IJCNN), pp. 1–7 (2013)
22. Xie, X., Zhang, G., Webster, C.: Data-driven reduced order modeling of fluid dynamics using linear multistep network (2018)
23. Zaklouta, F., Stanciulescu, B.: Real-time traffic sign recognition in three stages. Robot. Auton. Syst. **62**(1), 16–24 (2014)
24. Zhang, Z., et al.: Modeling of a CO2-piperazine-membrane absorption system. Chem. Eng. Res. Des. **131**, 375–384 (2018)