Dia Rule Discovery using Neural ODEs
Using EODNet for ASL Classification

Name -Supratim Sarkar
Reg. No: - 17BCE2203
Mobile No: - +96896258843
Mail Id: - supratim.sarkar2017@vit.ac.in

Name -Subhaditya Mukherjee
Reg. No: - 17BCE2193
Mobile No: - +961555953688
Mail Id: - msubhaditya@gmail.com

Guide Name: -Vijayarajan V
Designation: - Associate Professor
Mobile No: - +919841101779
Mail ID: -vijayarajan.v@vit.ac.in

B.Tech.

in

Computer Science and Engineering

School of Computer Science & Engineering

®

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

1. Introduction
    1.1. Theoretical Background
    1.2. Motivation
    1.3. Aim of the proposed Work
    1.4. Objective(s) of the proposed work
2. Literature Survey
    2.1. Survey of the Existing Models/Work
    2.2. Summary/Gaps identified in the Survey
3. Overview of the Proposed System
    3.1. Introduction and Related Concepts
    3.2. Framework, Architecture or Module for the Proposed System(with explanation)
    3.3. Proposed System Model(ER Diagram/UML Diagram/Mathematical Modeling)
4. Proposed System Analysis and Design
    4.1. Introduction
    4.2. Requirement Analysis
        4.2.1. Functional Requirements
            4.2.1.1. Product Perspective
            4.2.1.2. Product features
            4.2.1.3. User characteristics
            4.2.1.4. Assumption & Dependencies
            4.2.1.5. Domain Requirements
            4.2.1.6. User Requirements
        4.2.2. Non Functional Requirements
            4.2.2.1. Product Requirements
                4.2.2.1.1. Efficiency (in terms of Time and Space)
                4.2.2.1.2. Reliability
                4.2.2.1.3. Portability
                4.2.2.1.4. Usability
            4.2.2.2. Organizational Requirements
                4.2.2.2.1. Implementation Requirements (in terms of deployment)
                4.2.2.2.2. Engineering Standard Requirements

4.2.2.3. Operational Requirements (Explain the applicability for your work w.r.to the following operational requirement(s))

- Economic
- Environmental
- Social
- Political
- Ethical
- Health and Safety
- Sustainability
- Legality
- Inspectability

4.2.3. System Requirements

4.2.3.1. H/W Requirements(details about Application Specific Hardware)

4.2.3.2. S/W Requirements(details about Application Specific Software)

5. Results and Discussion
6. References

\

Abstract

Neural Networks are important for the approximation of bounded continuous functions in the field of Machine Learning. Such networks provide various frameworks and procedures for solving ODE's and PDE's in a numerical manner. In any field, data modeling is an extremely important factor in understanding a system. In classical cases, Ordinary differential equations were used to perform this discovery. A recent paper [1] merged the concepts of ODEs with Deep learning and created a neural ODE in which a ResNet architecture's res-blocks were substituted with an ODE solver which led to not only better results, but also more memory efficiency. Various architectures of Deep Learning have now been considered as Neural ODE's. It helps bring closer the gap between deep learning and systems that are dynamic in nature. This project aims to attempt to come up with a better architecture that gives SOTA results on the ASL classification dataset. The classifier will able to recognize patterns and identify the correct image and sign from the dataset which will be trained using the Neural ODE for solving the system. We aim to create a new architecture which generalizes and does well in a variety of tasks and uses the principles of Neural ODEs.

We also propose EODNet : Efficient ODE Net as the novel architecture.

Keywords:- ResNet ,Neural Networks ,Neural ODE, PDE, Classification, Deep Learning,SOTA,ASL

1.Introduction

1.1 Theoretical Background

The concept of Neural Networks has been used in approximation of bounded continuous functions in the field of Machine Learning. Neural Networks are algorithms that recognize relationships among complex datasets to identify patterns and complete tasks. Neural Networks have been efficient in solving time series data and NLP problems. These networks provide various frameworks and procedures for solving ODE and PDE in a numerical manner. ODE is referred to as Ordinary Differential Equation which consist of one or multiple functions of an

independent variable and the derivatives of the function in the equation. Neural ODE solvers have been instrumental for solving complex equations containing various functions of variables. Neural ODE's have performed better than normal ResNets in terms of various parameters. It has been shown to be more memory efficient and computationally adaptive. Both speed of solving equations and the accuracy of results can be balanced. Parameter efficiency is said to be better in Neural ODE solvers.

1.2 Motivation

In any field, data modeling is an extremely important factor in understanding a system. In classical cases, Ordinary differential equations were used to perform this discovery. A recent paper [1] merged the concepts of ODEs with Deep learning and created a neural ODE in which a ResNet architecture's res-blocks were substituted with an ODE solver which led to not only better results, but also more memory efficiency. This project aims to attempt to come up with a better architecture that gives SOTA results on the ASL classification dataset.

1.3 Aim of the proposed work

The aim is to come up with EODNet : Efficient ODE Net as the novel architecture which takes the concepts of efficient neural networks and combines them with neural ODEs to generalize better to newer tasks.

2.Literature Survey

2.1 Survey of the Existing Models/Work

Neural Networks are important for the approximation of bounded continuous functions in the field of Machine Learning. Such networks provide various frameworks and procedures for solving ODEâ and PDEâs in a numerical manner. The paper [9] discusses about the use of a solver for ODEâs and PDEâs using Neural Network frameworks. Such Neural Networks use function approximation for performing the tasks. The solver has proven to show high accuracy for both initial value problems and boundary value problems.it works for the functions as well as the derivatives. The solver is tested using collocation points using the Burgers equation and the heat equations. The ODE solver works well for the ODEâs and shows great accuracy. Boundary value problems are challenging to solve using the equation mentioned and another drawback is

the computational efficiency of the ODE solver. Accuracy must be improved in the case of boundary value problems. The paper [5] discusses that the stochastic gradient for a given ResNet neural network is made to converge to the stochastic gradient descent for a Neural ODE. All the layers in the ResNet neural network share the same weighted matrix. It proves that Neural ODEâs are the deep limit of such neural networks. Several equations have been tried and the result has been obtained using the Fokker-Plank equationsâ equation and theorems resulted in a large number of convergence results. Convergence rates are high and are optimal for the results. It only considered the convergence of minimizers and not the optimization procedure. Regularization cannot solely explain the numerical simulation. The use of DiffEqFlux library is described in [10] and an explanation for how it is used for solving several ODEâs and other type of differential equations is given. All the ODEâs are taken as a flux defined neural network. The purpose of using the DiffEqFlux library is to solve such ODEâs where simple integration strategies are not sufficient to get optimal results and solutions with high accuracy. The ODEâs are defined as neural networks and proper description of the Flux model zoo is provided that includes neural stochastic differential equations. The library helps in developing highly accurate ODE solvers that can play a major role in various applications of Machine Learning and Data Science. Various architectures of Deep Learning have now been considered as Neural ODEâs. It helps bring closer the gap between deep learning and systems that are dynamic in nature. The major challenges faced by [4] are figuring out the working of the models and the dynamics associated with them. The design choices for building the modules have to be elucidiated.The framework is set up for building a general Neural ODE and the components associated with it are considered. Infinite dimensional problem has been solved using numerical approximations that lead to better models for solving ODEâs. There are a variety of linear Ordinary differential equations that exist today for which various methods exist to solve them. The paper[3] discusses about the multiquadric radial basis function networks and its use in solving linear ODEâs in an efficient manner. By approximating functions and its derivatives using radial basis function networks, another RBFN is formed which can be efficient in solving Linear ODEâs with better accuracy. It is the most optimal method to determine the existing parameters. It fixes the problem of not having enough data samples as it deals with only the domain and the existing boundaries associated with it.The method can be used to solve even PDEâs.A sample set is selected from the training set and positions of centers of basis functions are selected-means clustering algorithm is

used on the training set and the centers of the clusters are used as the centers of the basis functions. It is useful for large-scale problems that exist in several engineering fields.The major challenge faced by this approach is that the number of basis fucntions and data set must be provided with a priori. Function approximation has always been an unstable task when it comes to using differential equations. The challenge of achieving a stable flow and also having a low computational cost has been tackled by the authors[8]. The techniques proposed in this paper use an energy function to guarantee the asymptomatic stability of the solution. Every deep learning algorithm is function composition between an input and an output space and is basically a mapping between them. This process was known but a proper optimization process was not identified so far. This paper proposes an algorithm to reduce the instability and stiffness that a solver might face by using a parametrized energy function. This takes care of of the floor of the network States and moves it along the negative gradient. Instead of trying to learn a single function, the algorithm tries to learn functions from the function space. There are -3- multiple types of models that are possible and observed. Autonomous port Hamiltonian model, second order and stochastic models are also defined. In training the model smooth scaler cost function was used to understand how well the model performs and also to minimize the loss in an optimal setting. The computing of gradients is is efficient with respect to to the choice of the parameters. To ensure that the energy function remains steady, a soft constraint was also added. Neural network architectures are composed of states as well as hidden layers. Using this concept, it is possible to understand the mapping between the vector spaces of the input and outputs. We can further take this idea and replace the entire component of the hidden states by using a differential equation solver. And framing the network problem as an ODE problem. This is the premise of a paper by Chen et al. [6] The major advantage of using models success this is the near constant memory costs as well as a higher customization by means of allowance for trade off between speed as well as precision values. The second advantage is that of huge boost in efficiency due to the process of back propagation not being required as a differential equation solver is being used instead. This also leads to pretty high guarantees about stagnating errors as well as the ability to be trained using the maximum likelihood model. It is also possible to to train on data that is not static and arrives non concurrently due to its nature. The method used to solve the equations makes use of the calculation of a secondary equation backwards which allows for efficiency in memory as well as the ability to define the error threshold. These benefits do not generally come

with standard deep learning models. Making use of this black box idea, the authors replace the standard ResNet models with an architecture that uses the solver instead of hidden layers. The only issue is that the analogue to the depth of a network cannot be obtained in this scenario. The model RK-Net performs similar to the original ResNet on the MNIST dataset and many others as described in the paper. Using a solver also allows the model to have multiple hidden units without affecting the time complexity of the training. Being able to discover new equations using data has always been a dream when it comes to the field of computer science. A recent paper [1] shows the possibility of harnessing deep learning for the same by augmenting the capabilities of scientific modeling. This leads to the creation of an UDE which would potentially enable the user to extrapolate factors that do not exist in the data as well as be able to simulate better models more accurately. Using the principles of physics, it is also possible to incorporate the existing knowledge into the network. The combination of differential equations in an ML concept to create a PINN using the UAT has also been shown. The paper [2] demonstrates the ability to obtain the equations that define a system using just data and also the possibility of recovering an equation from the time series data to extrapolate it. It also talks about the possibility of using the physics laws of conservation to to identify new systems by means of transferring previous knowledge. The approach of using physics and form networks allows the creation of solvers that can efficiently create gradient related calculations that are required to to perform any kind of machine learning task. Even though the existence of residual networks have been present for a long time the technique for or constructing such models for a continuous and invertible function was not present. This paper[7] shows how to prove any homeomorphism that can be approximated by using a combination of neural networks and ODEs in Euclidean space. This leads to the ability of using residual connections and blocks to create a reverse mapping between the output space as well as input space and allow for applications such as deep learning-oriented ones.

2.2 Summary/Gaps identified in the Survey

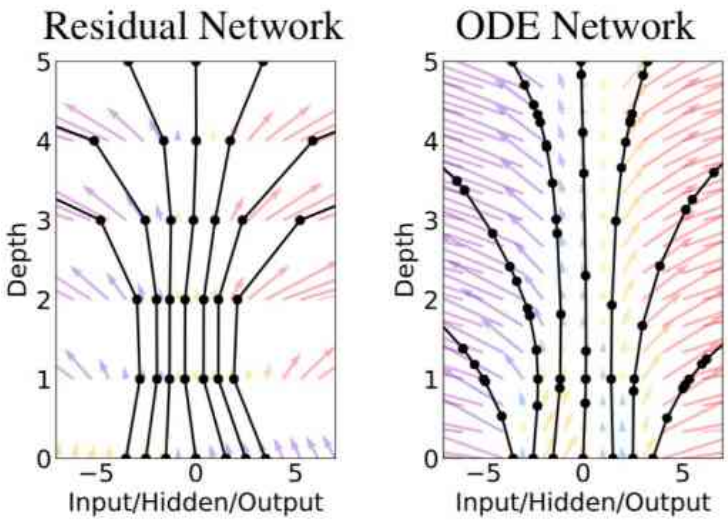
The major challenge faced by most of the approach is that the number of basis functions and data set must be provided with a priorate convergence functions are also not well defined and are mostly based on the dataset for performing computations. For ResNet architectures, the depth of the network cannot be accurately determined. The further challenges faced by the Neural ODE's

used are that they cannot be compressed into dynamic Neural Networks very easily and fitting time series datasets with high efficiency. In terms of computation, memory required can be high with larger computational times due to large number of layers and not a fixed architecture.

3. Overview of the Proposed System

3.1. Introduction and Related Concepts

This project is a research attempt into ODEs and combining them with neural networks to form neural ODEs. A novel architecture is proposed that combines a EfficientNet b5 backbone customized with a Neural ODE structure. This allows for more explanibility and further efficiency. The ideas are applied to the task of ASL recognition.



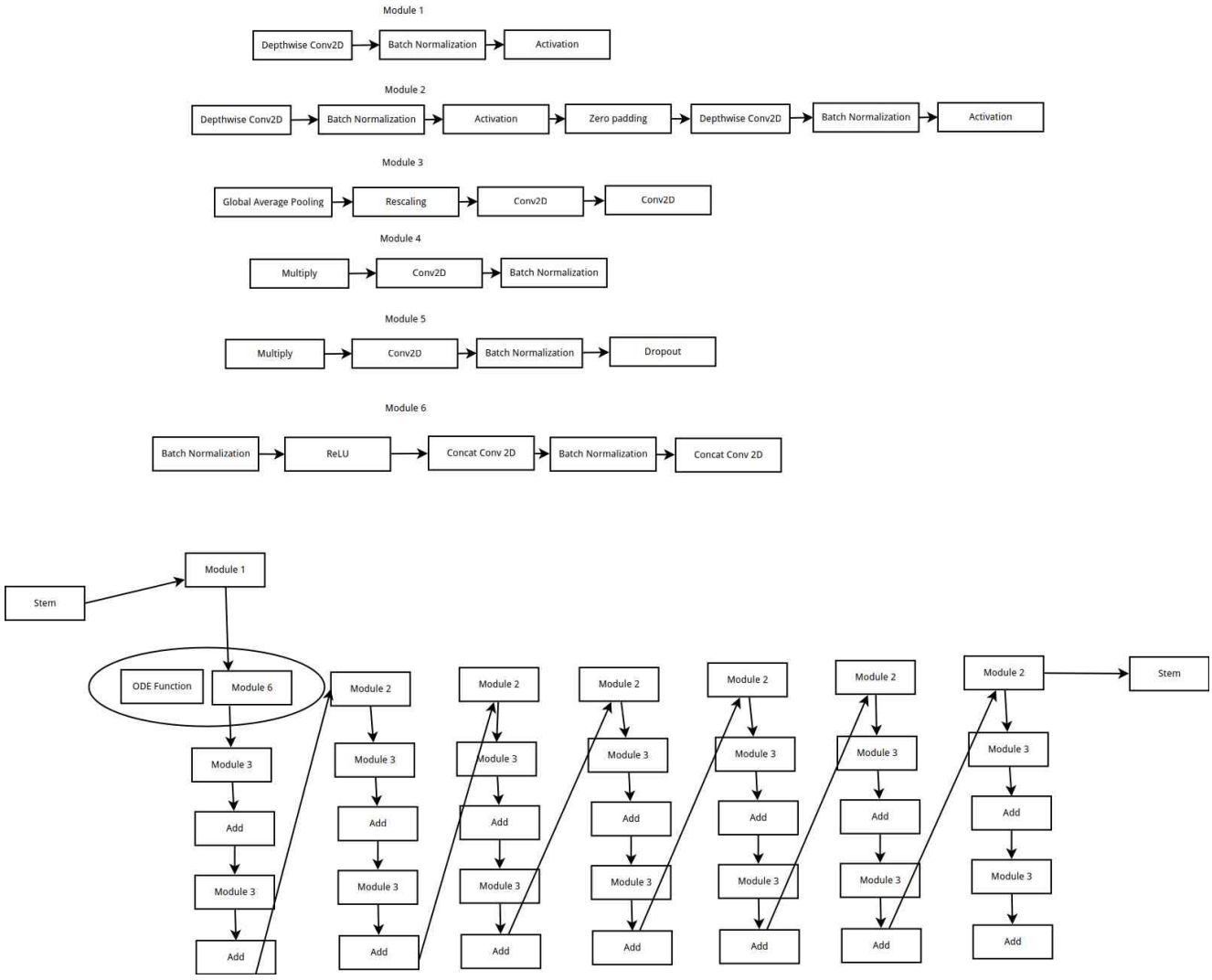
Compared to a normal Residual network, the depth and the flow of parameters in an ODE network are much more smoother and predictable which makes it easier to understand decisions mathematically.

3.2. Framework, Architecture or Module for the Proposed System(with explanation)

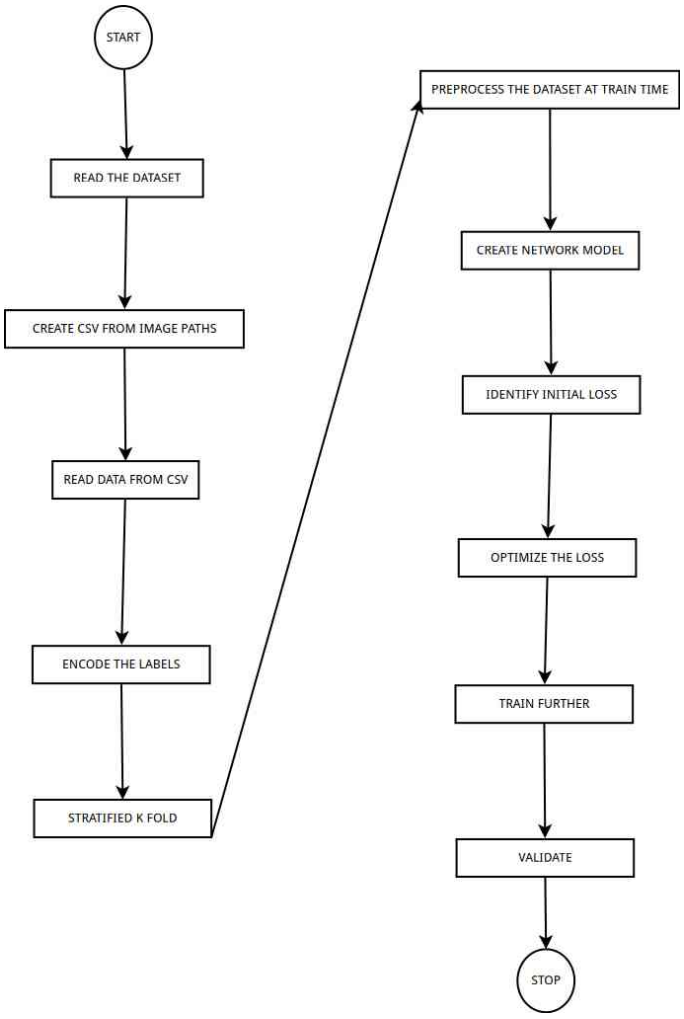
We propose EODNet : Efficient ODE Net and its subsequent architecture.

The framework for the system is built on a Python library called Pytorch. It is a deep learning library that is flexible and easy to work with. A separate trainer called pytorch lightning is also used, which would enable multi GPU support etc.

Since this is a neural network, the deep learning architecture that is used is a custom one. The backbone of efficient net b5 [11] is combined with a neural ODE. This is a new architecture and has not been found in the literature previously.



3.3. Proposed System Model(ER Diagram/UML Diagram/Mathematical Modeling)



4.2.1. Functional Requirements

Functional requirements are defined as any kind of requirement that is essential for the system and outlines the major features that would be required at the end as well as the limitations, assumptions and perspective of the end user.

4.2.1.1. Product Perspective

4.2.1.2. Product features

Being a research project, the features of this project are in development. The objective is to create a system able to identify American sign language from images without requiring any specialised hardware in near real time.

The system would be able to take images using a camera and predict which letter of the alphabet the end user is showing. This code could be used to predict what the end user is saying and output words by combining letters.

The novelty of the feature is a system using a newer approach which is more explainable in the long run compared to just using a traditional deep learning network.

4.2.1.3. User characteristics

The users can be of two categories. The first being someone who cannot speak properly and uses ASL to communicate. This category is the main target group. It is assumed that they have access to a phone and can see and interact with a simple system if required.

The second type of user category includes people who want to understand what someone else is saying. They will be more familiar with a point and identify type of system.

The central purpose of both the users are the same and there will not be much difference between their interfaces.

4.2.1.4. Assumption & Dependencies

The system is dependant on the availibility of a phone/computer with a camera. It also requires a minimum CPU for inference along with the capability to run inference from any neural network.

There was a single assumption that was made to simplify the use case. It was assumed that the users taking part would be able to see. Future development may include a voice interface but for now that will not be worked on.

4.2.1.5. Domain Requirements

The domain in which our system falls is social understanding. This would require a moderate amount of accuracy and a way of ensuring that the predictions are accurate and unbiased. It would also require explainability which is ensured by the addition of the neural ODE.

4.2.1.6. User Requirements

The user would require an interface that would allow for a camera input and then have a separate screen for inference. This inference would include some sort of sentences that the user has captured using the app from the person using sign language.

The user would also require an accurate application as the absence of accuracy would bring about challenges in communication.

4.2.2. Non-Functional Requirements

4.2.2.1. Product Requirements

4.2.2.1.1 Efficiency (in terms of Time and Space)

The inference takes place in constant time and time and space complexity varies as per the training of the dataset. Test dataset computation is performed accordingly and the efficiency and complexity are determined.

4.2.2.1.2. Reliability

It can detect the sign language successfully at the end of the test dataset. The user can identify the correct sign language used and the meaning of it. The results are accurate and the Neural ODE solvers compute the task in less memory space and faster execution time.

4.2.2.1.3. Portability

It can be used in various systems and interfaces. Any system with a functioning camera and GPU can run the software and use it. The Sign language detector accurately gives results in all such systems

4.2.2.1.4. Usability

Users can use the software with ease if there is a camera and good GPU in the system. The interface is easy to navigate and the user can use the Sign language detector very easily. It gives a very accurate result and helps in communication with people that need sign language to communicate with others. It gives users a detailed view of the sign language system and how it works in classifying sign language symbols and showing the result.

4.2.2.2. Organizational Requirements

4.2.2.2.1. Implementation Requirements (in terms of deployment)

There should a functioning camera and a GPU present. Python libraries and JupyterLabs are used for training the dataset and running the code. The dataset that will be used is the ASL dataset

from https://www.kaggle.com/grassknoted/asl-alphabet. This has about 29 classes of which, 10 classes will initially be used to test the system. This will be done using pandas and a data frame will be created from the dataset to allow for easier processing. Stratify will also be used to ensure that the classes remain the same count while sending it to the training loop. Data preprocessing is done on the dataset, data is then transformed and then the right model is evaluated for testing on the dataset After the dataset training is done, the Neural ODE solvers are used on the test dataset to classify the ASL images and detect the right symbol.

4.2.2.2.2. Engineering Standard Requirements

The primary requirement is identified which is need of classification of symbols of the ASL dataset. System planning takes place where all the user requirements and functionalities of the program is discussed. System specification and conceptual review is done to determine how the classification will be done using the Neural ODE solvers and recognize symbols of the ASL dataset. User flow and navigation is determined and implemented. Once the dataset is imported, data is sorted and transformed for the training of dataset to begin. The dataset is split into train and test data. Neural ODE solvers work on these classes of data for classification problem. System operational requirements are noted and implemented accordingly. Maintenance work is done for the right model evaluation and improving the training time and parameter tuning to achieve results with faster execution speeds and high efficiency. After all the analysis is done and requirements are defined, the system is developed and functionalities are implemented.

4.2.2.2.3 Operational Requirements

The ASL classifier that is solved by Neural ODE can have a great economic impact on the industry that is trying to help the handicapped people and people who only use sign language to communicate. The software can help such section of the society and they can communicate with normal people as well much more easily and faster. The usability and portability of the software can help bolster the emerging technologies that aim to help such sections of the society. It is of great social assistance and plays a pivotal role in making communication with them easier and promote better social culture and harmony. More people can interact with those that use sign language for communication. It serves ethical purposes as well and is very much sustainable for long term use.

4.2.3. System Requirements

4.2.3.1 H/W Requirements

● Intel i7-7700HQ 2.80GHz

● 16 Gb RAM The specifications of the GPU is as follows


```
CUDADevice with properties:

                  Name: 'GeForce 940M'
                 Index: 1
       ComputeCapability: '5.0'
          SupportsDouble: 1
           DriverVersion: 10.1000
          ToolkitVersion: 7.5000
       MaxThreadsPerBlock: 1024
         MaxShmemPerBlock: 49152
       MaxThreadBlockSize: [1024 1024 64]
                MaxGridSize: [1x3 double]
                  SIMDWidth: 32
                TotalMemory: 2.1475e+09
            AvailableMemory: 1.7149e+09
       MultiprocessorCount: 3
              ClockRateKHz: 1176000
               ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1
```

4.2.3.2. S/W Requirements

- numpy
- matplotlib
- pytorch
- pytorch lightning
- torchdiffeq
- JupyterLabs
- albumentation

5. Results and Discussion

As of review 2, the 70% implementation of the project has been done. The major modules have been implemented and extra features such as logging and visualization have also been implemented.

We obtain a 60% validation accuracy in just 16 epochs.

6. References

[1] Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., ... & Ramadhan, A. (2020). Universal differential equations for scientific machine learning. arXiv preprint arXiv:2001.04385.

[2] Filici, C. (2008). On a neural approximator to ODEs. IEEE transactions on neural networks, 19(3), 539-543.

[3] Jianyu, L., Siwei, L., Yingjian, Q., & Yaping, H. (2003). Numerical solution of elliptic partial differential equation using radial basis function neural networks. Neural Networks, 16(5-6), 729-734.

[4] Massaroli, S., Poli, M., Park, J., Yamashita, A., & Asama, H. (2020). Dissecting neural odes. arXiv preprint arXiv:2002.08071.

[5] Avelin, B., & Nyström, K. (2019). Neural ODEs as the deep limit of ResNets with constant weights. arXiv preprint arXiv:1906.12183.

[6] Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. arXiv preprint arXiv:1806.07366.

[7] Zhang, H., Gao, X., Unterman, J., & Arodz, T. (2020, November). Approximation capabilities of neural ODEs and invertible residual networks. In International Conference on Machine Learning (pp. 11086-11095). PMLR.

[8] Massaroli, S., Poli, M., Bin, M., Park, J., Yamashita, A., & Asama, H. (2020). Stable neural flows. arXiv preprint arXiv:2003.08063.

[9] Liu, Z., Yang, Y., & Cai, Q. (2019). Neural network as a function approximator and its application in solving differential equations. Applied Mathematics and Mechanics, 40(2), 237-248.

[10] Rackauckas, C., Innes, M., Ma, Y., Bettencourt, J., White, L., & Dixit, V. (2019). Diffeqflux. jl-A julia library for neural differential equations. arXiv preprint arXiv:1902.02376.
[11] Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning* (pp. 6105-6114). PMLR.