Generative Models for Trajectory Prediction

by

Venkata Anil Kota

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2021 by the
Graduate Supervisory Committee:

Heni Ben Amor, Chair
Sangram Redkar
Hemanth D. Venkateswara

ARIZONA STATE UNIVERSITY

December 2021

ABSTRACT

Trajectory forecasting is used in many fields such as vehicle future trajectory prediction, stock market price prediction, human motion prediction and so on. Also, robots having the capability to reason about human behavior is an important aspect in human robot interaction. In trajectory prediction with regards to human motion prediction, implicit learning and reproduction of human behavior is the major challenge. This work tries to compare some of the recent advances taking a phenomenological approach to trajectory prediction.

The work is expected to mainly target on generating future events or trajectories based on the previous data observed across many time intervals. In particular, this work presents and compares machine learning models to generate various human handwriting trajectories. Although the behavior of every individual is unique, it is still possible to broadly generalize and learn the underlying human behavior from the current observations to predict future human writing trajectories. This enables the machine or the robot to generate future handwriting trajectories given an initial trajectory from the individual thus helping the person to fill up the rest of the letter or curve. This work tests and compares the performance of Conditional Variational Autoencoders and Sinusoidal Representation Network models on handwriting trajectory prediction and reconstruction.

TABLE OF CONTENTS

Page

LIST OF FIGURES iv

CHAPTER

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Objectives 3

1.3 Trajectory Prediction 3

2 RELATED WORK 5

3 METHODOLOGY 8

3.1 Machine learning Methods 8

3.1.1 Types of Learning Methods 9

3.2 Multi-layered Perceptron Model 11

3.3 Latent Variable Modeling 13

3.3.1 Introduction 13

3.3.2 Autoencoders 13

3.3.3 Variational Autoencoders 15

3.3.4 Conditional Variational Autoencoders 17

3.4 SIREN 18

3.4.1 Implicit Neural Representations 18

3.4.2 SIREN and Periodic Activation Functions 20

3.4.3 SIREN for Trajectory Prediction 24

4 EXPERIMENTS AND RESULTS 25

4.1 Implementation Details 25

4.1.1 CVAE 25

4.1.2 Loss function of CVAE 26

CHAPTER Page

4.1.3 SIREN 26
4.1.4 Loss function of the SIREN-based model 27
4.2 Plots 28
4.2.1 Stock market price Prediction 28
4.2.2 Sine curve trajectory prediction 30
4.2.3 Human Handwriting trajectory prediction 32
5 DISCUSSION AND FUTURE WORK 37
REFERENCES 38

LIST OF FIGURES

Figure Page

1.1 General Case of Trajectory Prediction 4

2.1 Overview of Covernet Where a Trajectory Set Is Generated and Classified Over (Phan-Minh *et al.* (2020)) 5

2.2 TrafficPredict Architecture for a Node in Category Layer(Ma *et al.* (2019)) 6

2.3 Trajectron++ Model Representation(Salzmann *et al.* (2020)) 7

3.1 Schematic Diagram of a Mcculloch-pitts Neuron. The Index of the Neuron Is I, It Receives Inputs from N Neurons. The Strength of the Connection from Neuron J to Neuron I Is Denoted by W . The Threshold Value for Neuron I Is Denoted by . The Index T = 0,1,2,3,... Labels the Discrete Time Sequence of Computation Steps, and Sgn(B) Stands for the Signum Function 11

3.2 Multi-layered Perceptron Model 12

3.3 A Simple Autoencoder Layout 13

3.4 The Graph on the Left Represents a Plot of the Datapoints or Extracted Features from the Dataset. The Plot on the Right Shows the 2 Principal Component Analysis Axes When the Data Is Tried to Dimensionally Reduced to 2 Dimensions. 14

3.5 Outline of Conditional Variational Autoencoder 17

3.6 Outline of an Implicit Neural Representation Network 19

3.7 Outline of an Implicit Neural Representation Network Architecture with Periodic Activation Function (Siren) 21

Figure Page

3.8 The Reconstruction of Images, Their Gradients and Their Second Derivatives by Architectures with Different Activation Functions (Sitzmann *et al.* (2020)) 23

4.1 Generic Overview of Conditional Variational Autoencoder Architecture Implementation 25

4.2 Generic Overview of Siren Architecture Implementation 27

4.3 Nike and American Express Stock Price Prediction 28

4.4 Bank of America Stock Price Prediction 28

4.5 Model Loss Plot 29

4.6 Sin(t) Curve Trajectory Predictions. The Orange Line Shows the Actual Trajectory While the Plot in Blue Is the Predicted Trajectory by Cvae. The Figure on the Left and Right Are Predictions Taking a Kl Divergence Loss Weightage Of .01 And .1 Respectively 31

4.7 Sin(4t) Curve Trajectory Predictions. The Orange Line Shows the Actual Trajectory While the Plot in Blue Is the Predicted Trajectory by Cvae. The Figure on the Left and Right Are Predictions Taking a Kl Divergence Loss Weightage Of .01 And .1 31

4.8 Sin(8t) Curve Trajectory Predictions. The Orange Line Shows the Actual Trajectory While the Plot in Blue Is the Predicted Trajectory by Cvae. The Figure on the Left and Right Are Predictions Taking a Kl Divergence Loss Weightage Of .01 And .1 Respectively 31

4.9 Sin(4t) Curve Trajectory Predictions by Cvae When Conditioned on a Wrong Class of 8 Instead of 4 32

4.10 All the Different Letters and Curves in the Lasa Handwriting Dataset . 32

Figure Page

4.11 C Shape Curve Trajectory Predictions of 2 Different Behaviors on the Left and Right Respectively.The Blue Curve Is the Predicted Trajectory While the Orange Curve Is the Actual Human Motion Trajectory 34
4.12 L Shape Curve Trajectory Predictions of 2 Different Individual Behaviors on the Left and Right Respectively. The Blue Curve Is the Predicted Trajectory While the Orange Curve Is the Actual Human Motion Trajectory 34
4.13 Z Shape Curve Trajectory Predictions of 2 Different Individual Behaviors on the Left and Right Respectively. The Blue Curve Is the Predicted Trajectory While the Orange Curve Is the Actual Human Motion Trajectory 34
4.14 Depiction of Some Letters in the EMNIST Dataset (Cohen *et al.* (2017)) 35
4.15 Reconstructed Letters from Sparse Images by Siren-based Model 35
4.16 Predicted Letters from a Different User's Partially Written Letter Images by Siren-based Model 36

Chapter 1

INTRODUCTION

1.1 Motivation

Artificial Intelligence(AI) is the field or science of exceeding or imitating human intelligence. It has many applications such as in computer vision, natural language processing, knowledge reasoning, game playing, knowledge reasoning etc. Some more visible and tangible applications would be its use in self-driving cars, disease mapping, automated finance investing, conversational marketing bots, smart assistants and many more. Machine learning is a specific subset of AI that trains the machine to learn specific things and in some cases, to learn to learn skills etc. It mainly includes data handling and building, pattern recognition which allow a machine to make decisions with minimal human intervention. Robotics is an interdisciplinary field which involves design, construction and controls of the mechanical components of the robots to making the robots learn many different skills which humans might or might not be capable of.

Robots generally have various constraints and thus, whether the job of a person is to design or to control, it is not a very easy task. This gives impetus for the development of technology which would take manual design, control and planning out of the picture and allow the technology and algorithms to automatically perform these tasks with no human intervention. Also, humans are by nature very erroneous and not very precise. Humans are also not accurate in performing most of the tasks. These problems can be overcome by machines or robots which are very good at carrying out these tasks. Apart from all these obvious human limitations, even the decision

making capability of humans is highly questionable. There are many cases where humans make huge blunders ranging from placing some machine part or component in the wrong place to clicking the wrong button on a control panel or a computer which makes the person regret their decision later on. This might even cost a lot of money and time. There might also be small erroneous decisions which a human might make which does have any effect in the short term but there might be some significant undesirable changes in the long term. A typical example of this might be the frequent consumption or intake of addictive substances like alcohol which might not show its effect in the short term but as the person ages, the effects gradually show up. This would be very detrimental to the health of the humans.

The above stated points and examples show that humans are by nature very erroneous and act in a very unoptimized manner. Thus there then arises a need to develop technology which can handle everything that a general human fails to do in a satisfactory manner. Even better would be technology which performs better than humans. This work deals with developing decision making technology category which helps in predicting human behavior. The inaccuracy of human decision making might be due the lapse in concentration, loss of memory, laziness or lack of sufficient intelligence. Sometimes, human minds are not fast enough in computation or they lack the physical speed which might be considered satisfactory. It is mostly the case that the notion 'the faster the better' is true in the domain of performing physical tasks by the humans or robots for that matter.

Thus all of this is very crisply and cleanly as stated by the 3 laws of Isaac Asimov in his novel I, Robot stating:

1. A robot should not injure a human being or, stay put and allow a human being to be harmed.

2. A robot should obey all orders given by the human being unless the situation clashes with the first Law.

3. A robot should protect itself unless the situation is in conflict with the First or Second Law.

This leads to the formulation of the human robot interaction problem. The above stated points were one of the earliest formulations of the human robot interaction. Though the formulation contains only 3 points, it basically sums up all the goals to be achieved in the field of human robot interaction. This thesis work aims to solve one such problem where the robot learns an algorithm to predict human handwriting motions.

1.2 Objectives

The key objective of this work is to generate trajectories of data given a set of previous observations of the agent's trajectories. In particular, this work presents a comparison between 2 machine learning models used for trajectory prediction:

- Conditional Variational Autoencoders
- Sinusoidal neural representations

The performance of both these algorithms is to be analysed to gain a newer and deeper understanding as to how the latent space and the space of priors for neural representations work. The amount of sensitivity and accuracy in sampling from these spaces to generate future trajectories needs to be validated.

1.3 Trajectory Prediction

Trajectory prediction is an important component in most of today's advancing fields such as autonomous vehicles, stock market prices, human motion prediction

and so on. It mainly involves predicting the next trajectory of the vehicle for the next few seconds(uptil 5 seconds) in the case of autonomous vehicles.

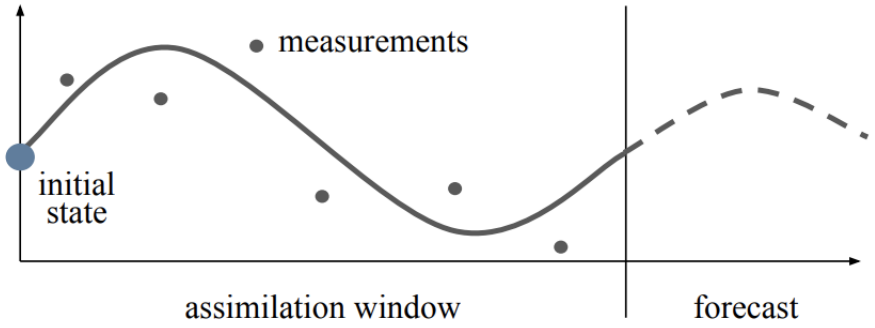

Figure 1.1: General Case of Trajectory Prediction

Fig. 1.1 gives an idea of what trajectory prediction in any scenario refers to. There is first an initial state of the agent or the system being referred to. Then, the measurements of the past positions or the values are used to forecast the future trajectory indicated by the dotted line.

Also, trajectory prediction models are also highly useful in predicting future trajectories of pedestrians. This involves both short and long term predictions. Whereas in scenarios such as stock market prices, the prices for the next few days can also be predicted given the prices on many previous days from the time of prediction. Developing close to accurate models for stock market price prediction have been traditionally very difficult to achieve as the stock market prices are often very volatile. With the state of art machine learning models, it is only possible to accurately predict the overall trend in the prices in the next many days. Trajectory prediction in general involves time series models due to the requirement of data from previous time steps to predict the trajectory in the next time steps.

Chapter 2

RELATED WORK

There are a lot of domains where trajectory forecasting is highly useful. Some of the most interesting domains related to robotics are autonomous vehicle trajectory prediction, future pedestrian motion prediction and human movement and behaviour trajectory forecasting. Below are some of the popular works related to these fields of trajectory prediction.

One of the most challenging tasks is to generate suitable trajectories for all the agents in the traffic in order to make reasonable navigation decisions. This majorly involves the prediction of the spatial coordinates of the vehicles in the short and long terms. Different kinds of vehicles have different dynamics and motion primitives. This affects the trajectory prediction due to their driving styles. There are different methods and models developed for trajectory prediction. In one of those works, they frame the problem as a classification problem over many sets of trajectories. These trajectories are provided to the model to train on.

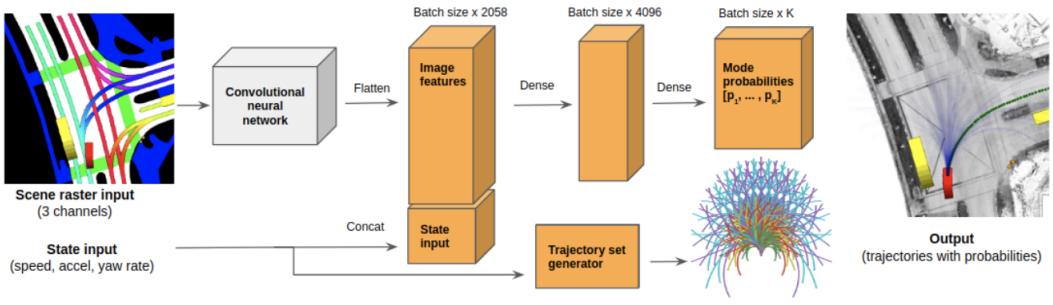

Figure 2.1: Overview of Covernet Where a Trajectory Set Is Generated and Classified Over (Phan-Minh *et al.* (2020))

The model then learns to classify the situation and then learn which trajectory

the vehicle should take from the already known set of trajectories like in the paper by Phan-Minh *et al.* (2020). Fig. 2.1 gives an overview of the Covernet model.

There are also other works which are long short term memory(LSTMs) based approaches like Ma *et al.* (2019) where they introduce an algorithm called TrafficPredict. Here, the authors use an LSTM based approach to predict the trajectories of the traffic where they learn all the instances' movements and behaviour and then learn the similarities among the instances to easily categorize and for good prediction. The model overview for TrafficPredict is shown in Fig. 2.2.

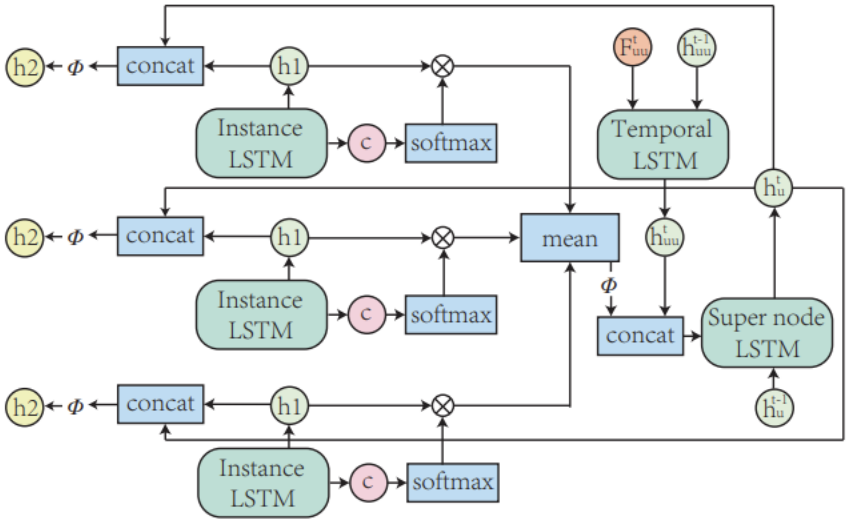

Figure 2.2: TrafficPredict Architecture for a Node in Category Layer(Ma *et al.* (2019))

Another one of the more phenomenal works would be Trajectron++ by Salzmann *et al.* (2020). In this paper, they present Trajectron++ which is a graph-structured recurrent model which forecasts the trajectories of some agents like humans taking the agent dynamics into consideration. This model basically produces control sequences

which are true for any kind of agent. These control sequences are then integrated into the specific agent dynamics for forecasting its trajectory. This is advantageous when we know the dynamics of the agent perfectly which is true in most cases. Thus, the neural network does not have to learn the dynamics as well which is the case if we want to output the position waypoints instead of control sequences. The model overview is shown in Fig. 2.3.

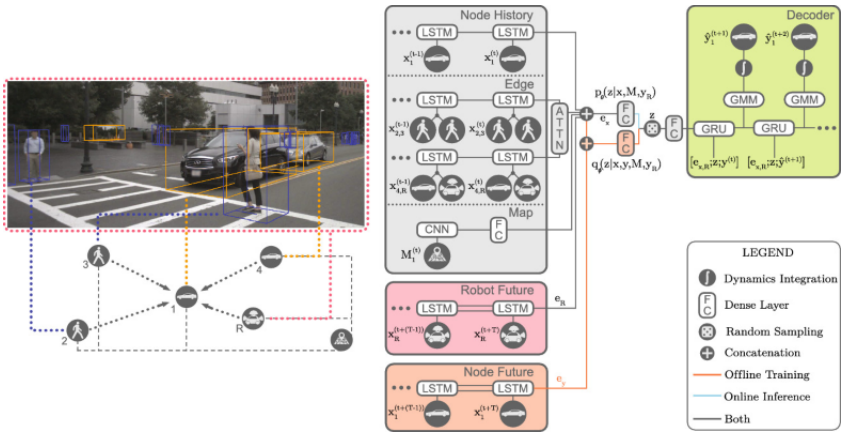

Figure 2.3: Trajectron++ Model Representation(Salzmann *et al.* (2020))

There are also other important works in this field like Social LSTMs by Alahi *et al.* (2016) and Social Attention by Vemula *et al.* (2018). In the work by Alahi *et al.* (2016), the authors present an LSTM-based model which is used to learn human movement and predict their future trajectories. They test this model on various other baseline models like Linear model, collision avoidance, social force model, iterative gaussian process and LSTM with occupancy maps and show that this model has a better performance than the other mentioned models.

Multi-modal trajectory prediction is also seen in works by Cui *et al.* (2019) and by Dong *et al.* (2021). In these works, the focus is more on generating a number of closely feasible trajectories.

Chapter 3

METHODOLOGY

3.1 Machine learning Methods

Machine learning methods are the most widely used for all kinds of tasks with which machines can perform certain functions similar to the human brain. In particular, it is about the learning tasks which the human brain performs. Growing from a child to an adult, humans always keep learning new concepts, methods and acquire new skills in the day-to-day life. A step above that would be the humans developing and improving the ability of abstraction and forming generic correlations and causality. This helps in the improvement of the generic problem solving ability of humans also known in colloquial language as intelligence.

Machine learning methods mainly broadly deal with pattern recognition in which the machines learn to deduct and observe similar patterns in various applications and problems similar to how humans start learning any new task. In general, when a human becomes very good at pattern recognition, the person is usually deemed as an intelligent person. Similarly, artificial intelligence is the intelligence pertaining to a machine when it becomes very good at pattern recognition. Recent advances have shown the algorithms to be very good at performing specific learning operations like classifying data like images, learning the underlying structure in the data and so on. It is similar to the human brain in the sense that the algorithms are very good at performing certain tasks which the human brain can perform. The ideal human brain or a super human brain can be developed when an algorithm can perform all the tasks which a human brain can achieve. The machine would then have some attributes like

it would be very fast, large memory and also good at problem solving than humans. This is sometimes called as artificial general intelligence. Thus, machine learning is a powerful way to achieve artificial general intelligence.

3.1.1 Types of Learning Methods

There are different kinds of learning techniques such as supervised learning, unsupervised learning and semi-supervised learning methods. Starting first with supervised learning:

Formulation

Supervised learning is a category of machine learning techniques where a function $f : \mathbb{R}^{N \times M} \rightarrow \mathbb{R}^{P}$ is learnt to map X values to y in the following equation:

$$f(X) = y \tag{3.1}$$

We have some samples of $X \in \mathbb{R}^{N \times M}$ to $y \in \mathbb{R}^{P}$ mappings and this is called the dataset which is used to learn the the function. Here, X consists of N input vectors each of size M while y is a vector of size P. In general, the X values in the dataset are termed as inputs and the corresponding y values as their labels.

Bias-Variance

Bias indicates how likely the machine learning model is to not predict the output for particular inputs of X. Whereas variance indicates how likely the model is to not have a constant prediction value of X when it is trained on different datasets which are equally good. It is harmful for the model to either have a large bias or large variance and vice-versa. So, there needs to be a tradeoff between the 2 quantities while designing supervised learning models.

Some examples of supervised learning algorithms include Bayesian Networks, Naive Bayes, Logistic Regression, K-Nearest Neighbours, Decision Trees, Random Forests, Support Vector Machines and Neural Networks.

Unsupervised learning is a category of machine learning techniques where the algorithms take data as input and try to find some patterns in the data on their own to perform specific tasks or to learn some representations from the given data. Thus, they majorly differ from supervised learning algorithms in their ability to learn from unlabelled datasets.

One of the classic examples would be the clustering problem where the algorithm needs to learn to cluster the data based on the specific patterns it observes in the given data on its own. This includes different variety of clustering problems such as k-means, DBSCAN, mixture models and so on. Other algorithms such as anomaly detection, generative adversarial networks (GANs), self-organizing maps also belong to this class of learning algorithms since they do not have any target labels or values associated with the input data, but rather analyze the changes and patterns in the data itself. Dimensionality reduction algorithms also fall under this category. Some of them are Non-negative matrix factorization (NMF), Linear discriminant analysis (LDA), Generalized discriminant analysis (GDA), t-distributed stochastic neighbor embedding (t-SNE), Uniform manifold approximation and projection (UMAP), Principal Component Analysis (PCA), Autoencoder models and many more.

One of the techniques used for trajectory prediction in this work is based on autoencoders and conditional variational autoencoders which can also be used as a dimensionality reduction technique. In this work, it is used to encode the richer and the main behavioral representations from data of quite large dimensions.

Sometimes, it becomes difficult to get labelled data and it is not feasible to use supervised learning in these scenarios. At the same time, it might not be possible

to train the algorithm on unlabelled data. So, there will only be some labelled data points while others would be unlabelled. In this case, semi-supervised learning is used to learn from the given data and labels. One of the main reasons this problem occurs is because there is a huge amount of data to be annotated and labelled which is not manually possible. There are some cases where the human may not even know how to label some of the data.

These methods can either be used to find the labels of the unlabelled data or map the whole input domain to output space. They are called transductive and inductive learning respectively. So, semi-supervised learning helps to deal with these scenarios. Some of the methods belonging to this class of learning algorithms are Generative Models, Low-density separation, Heuristic approaches and Laplacian regularization.

3.2 Multi-layered Perceptron Model

There is a linear regression model in which a linear curve is fitted to the data points in a single or multi-dimensional space. Whereas in logistic regression, we introduce non-linearity factor which gives the algorithm more flexibility.

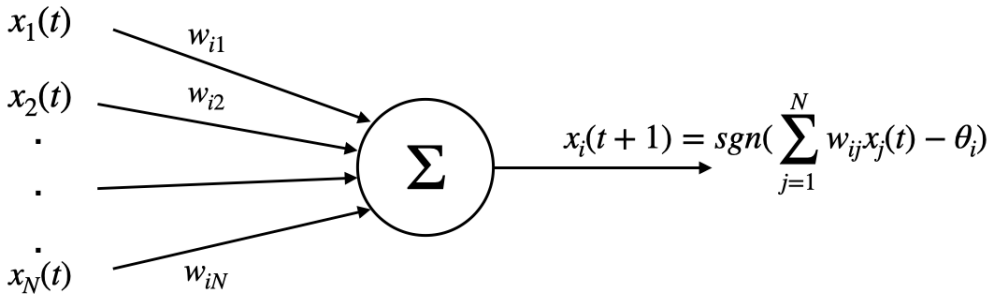

Figure 3.1: Schematic Diagram of a Mcculloch-pitts Neuron. The Index of the Neuron Is I, It Receives Inputs from N Neurons. The Strength of the Connection from Neuron J to Neuron I Is Denoted by W . The Threshold Value for Neuron I Is Denoted by . The Index T = 0,1,2,3,... Labels the Discrete Time Sequence of Computation Steps, and Sgn(B) Stands for the Signum Function

A perceptron basically does logistic regression. It is a function which takes in one

or more inputs, then takes the sum of all the outputs after multiplying them with some weights. Finally, this is passed through an activation function like sigmoid, "rectified linear unit (ReLu)" and so on to gain the advantage of non-linearity after adding the bias or the threshold term to it. The resulting value is the output of the perceptron. This is shown in Fig. 3.1.

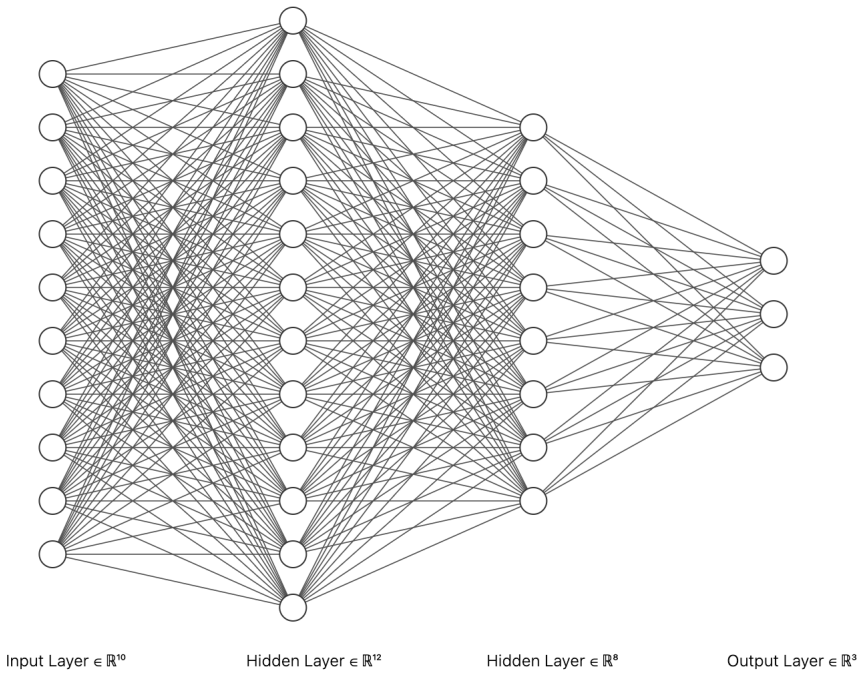

Figure 3.2: Multi-layered Perceptron Model

Also, the perceptrons can be stacked up and the weights can be interlinked between them to form a more complex function. This is called a "multilayered perceptron (MLP)". This model is used because in most of the real-world scenarios, a complex function is required to perform a task or fit some data. Fig. 3.2 shows an outline of an MLP. An MLP contains an input layer followed by some hidden layers of varying or the same dimensions. These hidden layers are finally followed by an output layer. Each of the neurons in the hidden layers may have different activation functionsaccording to the desired non-linearity at these layers.

3.3 Latent Variable Modeling

3.3.1 Introduction

Latent variable models map a given set of observations or data to the variables in the latent space. This latent space is rich in the information about the observed data and they are very good at encoding the patterns, features and attributes of data. Some of the examples include gaussian mixture models and autoencoders. In this thesis, one of the types of variational autoencoders is used to perform trajectory prediction. They are called conditional variational autoencoders and they are very good at storing latent data in a sophisticated manner.

3.3.2 Autoencoders

An autoencoder is usually a neural network that tries to learn to copy its input and produce an output. It contains two components: encoder which encodes the input data and the decoder tries to decode this encoded representation to produce meaningful results. This is the main idea which is to stack autoencoders and then greedily train it. Autoencoders are used in various fields and has many applications like dimensionality reduction, information retrieval, data compression, clustering and generating new examples.

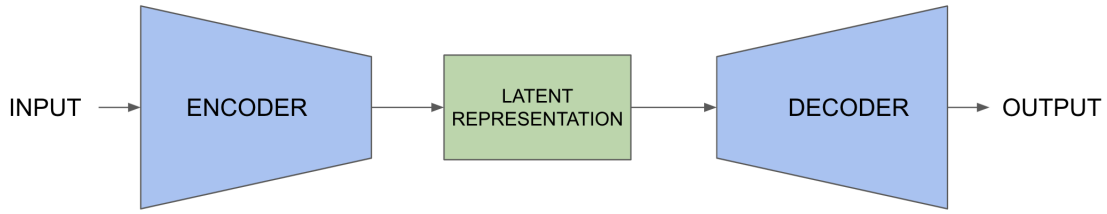

Figure 3.3: A Simple Autoencoder Layout

Conditional variational autoencoder is a special kind of autoencoder and it was used as one of the models for trajectory forecasting. Conditional variational autoen-

coders are the one of the best models for reconstruction and data generation. The prediction of trajectory in the various models as shown in the results section show that the trained model has a decent accuracy.

Autoencoders try to reconstruct input data using an unsupervised learning technique. They compress the input data to learn the underlying representation of the data and then try transform the data back to the input data. The component which tries to compress the data is an encoder and the one which tries to reconstruct data from the compressed representation is a decoder. This is basically identity mapping with constraints which learns the compressed data representation.

We can extend it to a many layered neural networks acting as the encoders and decoders. A simple autoencoder model outline can be seen in Fig. 3.3. The loss function over which optimization takes place includes the reconstruction loss term as well as a regularizer term to avoid overfitting. Though autoencoders might work well to just reconstruct input, they do not perform well as a generative model. This is because a random input to the decoder does not produce good output. This is mainly because the new input might be very far from the trained inputs of the decoder.

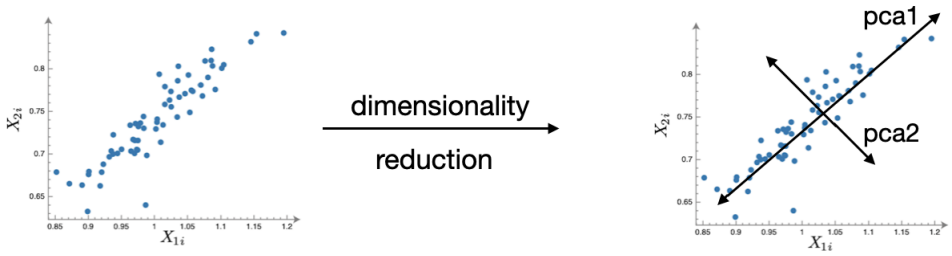

Figure 3.4: The Graph on the Left Represents a Plot of the Datapoints or Extracted Features from the Dataset. The Plot on the Right Shows the 2 Principal Component Analysis Axes When the Data Is Tried to Dimensionally Reduced to 2 Dimensions.

Autoencoders are in one way similar to Principal Component Analysis(PCA) because they can be seen as an efficient dimensionality reduction technique. This is

because encoder basically reduces the dimensionality of the input data in a meaningful way and store important information so that the decoder can then interpret this information and increase the dimensionality of the encoded information. Thus, it is very necessary that the encoder model is chosen very well since it needs to encode only rich and useful information from the input which indirectly also depends on the encoder model choice. This thereby leads to solving many problems in the domain of data compression where data loss is one of the most important factors. Since PCA is also a data compression technique for low dimensional representation of data, it is very similar to an encoder. This is due to the fact that many input features are correlated and this correlations are learnt by the autoencoder to encode information. A practical example of an application of dimensionality reduction using PCA is shown in Fig. 3.4.

3.3.3 Variational Autoencoders

Latent variables in variational autoencoders(VAE) by Kingma and Welling (2013) are used to describe or represent the data. This makes the model more expressive. Some of the notations used are $X \in \mathbb{R}^{N \times M}$ which represents data of dimensions M and N, $z \in \mathbb{R}^{P \times Q}$ which belongs to a lower dimensional space of dimensions P and Q. $P(X)$ represents probability distribution of data, $P(z)$ indicates probability representation of latent variable and $P(X|z)$ is the conditional probability distribution given latent variable.

The goal is to find P(X). Since any distribution can be modelled as a Gaussian mixture, we marginalize z and try to find $P(z)$ using $P(z|X)$. $P(z|X)$ is inferred using variational inference where it can be thought of as an optimization problem and model $P(z|X)$ as a simple gaussian or normal distribution. The loss between $Q(z|X)$ and $P(z|X)$ is minimized using KL divergence metric i.e., $D_{KL}[Q(z|X)||P(z|X)]$. Here,

it is seen that $Q(z|X)$ and $P(z|X)$ are the encoder and decoder respectively. After applying the Bayes rule and deriving, we get the objective function to be:

$$\begin{aligned} &\log P(X) - D_{KL}[Q(z|X)||P(z|X)] \\ &= \mathbb{E}[\log P(X|z)] - D_{KL}[Q(z|X)||P(z)] \end{aligned} \tag{3.2}$$

where $Q(z|X)$ is the model's predicted latent space distribution and P(z—X) is the actual latent distribution. $\mathbb{E}[\log P(X|z)]$ is the expectation of the approximation of the true latent space distribution.

The latent variable distribution $P(z)$ is modelled as $\mathcal{N}(0, 1)$ since it is the easiest choice. Now, $Q(z|X)$ can be assumed to be Gaussian with parameters $\mu(X)$ and $\sum(X)$. Now, the equation 3.2 can be written as follows:

$$D_{KL}[\mathcal{N}(\mu(X),\, \sigma(X))||\mathcal{N}(0, 1)] =$$
$$\frac{1}{2}\sum_{k}(exp(\sum(X)) \,+\, \mu^2 \,-\, 1 \,-\, \sum(X))$$

where k is the dimension of the Gaussian.

Reparameterization trick

If z is sampled from the Gaussian parameters which the encoder outputs directly, it becomes difficult to train the model as the sampling operation does not have gradient. This poses a problem while executing backpropagation. This is solved by factoring out the parameters of the latent distribution. ϵ is sampled from a standard normal distribution and it can be easily converted to a Gaussian with the mean and variance which the encoder outputs. We use the fact that any gaussian can be described by only sampling from a normal distribution. This is one of the key tricks which makes variational autoencoders work in practice. This reparameterization equation is given as:

$$z = \mu(X) + \sum \tfrac{1}{2}(X)\epsilon \tag{3.3}$$

where $\epsilon \sim \mathcal{N}(0, 1)$

The data points with the same label are close together in the latent space distribution. VAEs can be used for data reconstruction and also for generating new samples by sampling points from the normal distribution.

3.3.4 Conditional Variational Autoencoders

The problem with variational autoencoders is that there is no control over the data generation process. That is, VAEs cannot explicitly produce data of a particular label in demand. This is solved with the help of a "conditional variational autoencoder(CVAE)". CVAE by Sohn *et al.* (2015) models the latent variables and data, conditioned to a random variable which is usually given as an extra input to the encoder and decoder. The layout of CVAE can be seen in Fig. 3.5. Some of the works using CVAE in trajectory prediction are by Rhinehart *et al.* (2019) and Ivanovic and Pavone (2019).

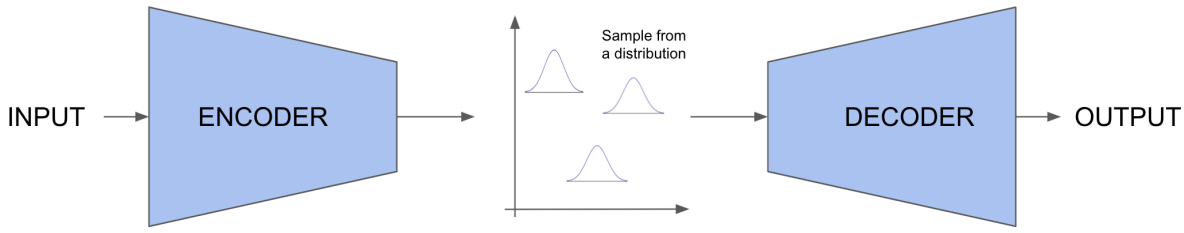

Figure 3.5: Outline of Conditional Variational Autoencoder

In VAE, the encoder does not take the different type of X into account. Thus, it will not be able to generate specific data. Usually, while executing CVAEs, we condition the latent variables and data on the label of X. That is, we also give the label of X as an input to both the encoder and decoder. Now, if the variable on

which the data is conditioned on is c, the encoder and decoder become $Q(z|X,c)$ and $P(X|z,c)$ respectively. Thus, the new objective function is given as:

$$\begin{aligned} &\log P(X|c) - D_{KL}[Q(z|X,c)||P(z|X,c)] \\ &= \mathbb{E}[\log P(X|z,c)] - D_{KL}[Q(z|X,c)||P(z|c)] \end{aligned} \tag{3.4}$$

This implies that there is a separate P(z) for each value of c. Thus, even if we input the same point from the standard normal distribution of the data in the latent space, under different labels, it will produce different outputs.

Since each datapoint under a specific label has its own distribution, it is very easy to sample unlike VAE, where all the datapoints of all labels are mixed. Also, the reconstructions in VAE suffer from edge cases in the latent distribution and may not result in correct reconstructions.

Another advantage of CVAE is that we can generate new data under our specific condition given by the label. This grants a lot of control over data generation.

3.4 SIREN

3.4.1 Implicit Neural Representations

Many of the works in machine learning include taking some kind of signal as input to our model and then learn something about the real world with the help of this signal. In case of computer vision for instance, one look at signals which are images and then learn about our 3D world. The way one chooses to represent a signal has a huge impact on the kinds of models we build to analyze the signal afterwards. It is worth asking how to represent the signals today or in recent times. Most of the time, the signals are represented discreetly. For example, images are discrete grids of pixels, shapes are discrete voxels or meshes and audio waves are represented as discrete amplitude values.

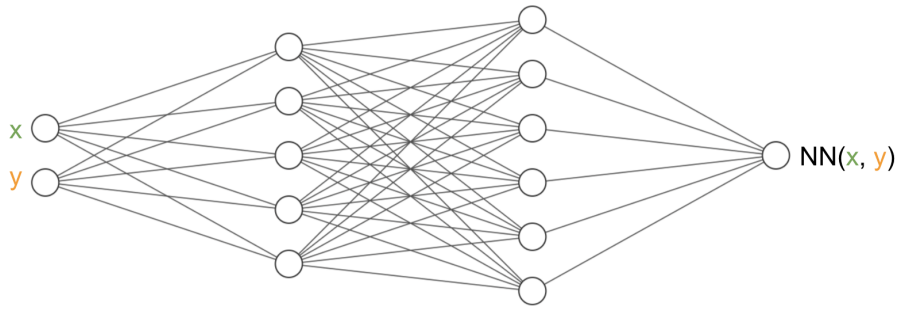

Figure 3.6: Outline of an Implicit Neural Representation Network

If the images were not represented as grids of pixels, it is very unlikely that convolutional neural networks would still be used today. Still, this is quite an intriguing question because seeing how the choice of parameterization has impacted the algorithm is worthwile to think about. This way of parameterization into pixels or amplitudes is not obvious whether it is a particularly good or bad way of parameterization. So, there has been a new kind of representation proposed for 3D shapes and the key idea here is to encode a 3D shape in the weights of a neural network. In particular, the 3D shape is represented as a neural network that maps the (x, y, z) coordinate to the occupancy of the 3D shape at that coordinate or the signed distance of the shape at that (x, y, z) coordinate. Some of the important benefits to this representation are:

1. The representation is agnostic to grid resolution or any sampling criterion.

2. Model memory scales only with signal complexity and is independent of the sampling density.

3. It admits effective learning of priors. This is really a key benefit as can be seen in Mescheder *et al.* (2019) where they reconstruct 3D shapes and also used in the paper by Chen *et al.* (2018)

Thus, these representations are called implicit neural representations as shown in

Fig. 3.6. There has also been similar kind of work done by Berg and Nyström (2018).

3.4.2 SIREN and Periodic Activation Functions

There are a few papers which use implicit neural representations to parameterize simple 3D shapes but fail to parameterize more complex shapes and signals. For example, a scene such as representing a room is very difficult to model for the proposed neural networks with "multilayered perceptron(MLP)" with ReLu activation functions. This model fails to encode all the geometry and fine detail in the scene. It was also shown that the implicit neural representations work only on simple 3D shapes. So, the next question would be to see if they work on image signals, audio signals and so on. But it is found that the ReLu MLPs fail to properly parameterize signals like audio and images.

Also, a lot of natural signals are shown as the solutions to partial differential equations in the physical world like lightwaves, audiowaves etc. Parameterizing these natural signals with ReLu MLPs also does not work. Thus, modelling physics in this world with these kinds of implicit neural representations becomes a challenge. The reason they do not work is not completely clear and it is currently a hot research topic. But a few things can still be pointed out as problems with using Relu MLPs:

- The learned function is not shift-invariant. That is the ReLu MLP finds it very difficult to apply the same function at two or more different (x, y, z) coordinates. This becomes a big problem if we have signals with their information spread out over a large domain or the ones which have fine detail information across the whole domain like an image.

- With regards to physical signals, it turns out that the conventional architectures fail to parameterize the derivatives of the signals in the physical domain. It can

be seen intuitively for ReLu MLPs because all functions parameterized by ReLu MLPs are piecewise linear and this indicates that the first derivative is piecewise constant and the second derivative would be zero. This would imply that any signal carrying information in the second derivative cannot be parameterized by a ReLu MLP like soluitons to the wave equation. It is also the same case with other non-linearities like the tanh non-linearity which do not parameterize signals with second derivatives effectively.

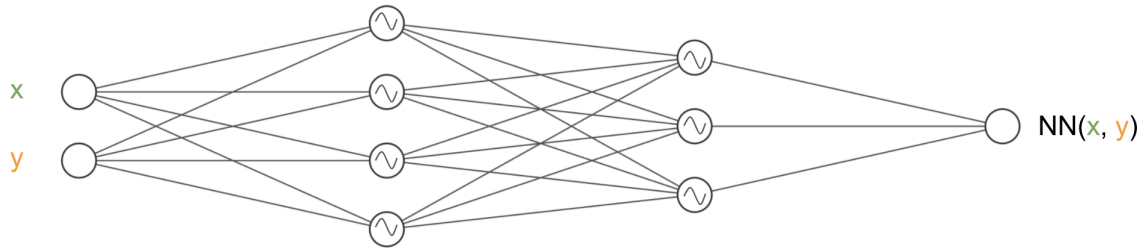

Figure 3.7: Outline of an Implicit Neural Representation Network Architecture with Periodic Activation Function (Siren)

To address the above two issues, a new model called "Sinusoidal Representation Networks (SIREN)" Sitzmann *et al.* (2020) is used. It is a multilayered perceptron network with periodic sine function as their activation function instead of the conventional ReLu, ReLu with positional encoding, tanh etc. ReLu with positional encoding is used in various recent works including transformers by Vaswani *et al.* (2017) and some other works by Bilan and Roth (2018). This is used along with the initialization scheme to get good results. So, the above two issues are addressed by this architecture as follows:

- It can be observed that there is a gain of shift-invariance in SIREN which is not the case with conventional activation functions. This can be seen intuitively as in if the first layer of the MLP generates the same set of activations for more than one coordinate, then the rest of the MLP also generates and applies the

same function at these coordinates. Then, if a single neuron with sine activation function is considered, it can be seen that it produces the same embedding or output for many points in its input domain. This helps in gaining some shift invariance. Thus, it is very easy for this architecture to apply the same function at different (x, y, z) coordinates.

- The sine activation function also has a very beautiful property which is that the derivative of the function is cosine or the same periodic sine function with some shift. This means that the neural network that parameterizes the derivatives of the input coordinates is also a similar MLP network with sine non-linearities as activation functions. Thus, the derivatives of the network inherit some of the properties of the network itself.

The SIREN architecture as shown in Fig. 3.7 can parameterize signals like images, shapes, audio, video and physical signals along with their dervatives very well. For example, when it is trained to parameterize an audio signal, the training data would be the time points and their corresponding amplitudes as their labels. The L2 loss between the output of the network $\Phi(t) : \mathbb{R}^M \rightarrow \mathbb{R}^N$ where the function Φ maps vector coordinates of dimensions M to the distance at those coordinates with a vector dimension of N. and the actual amplitude values $f(t)$ over all the domain values Ω is calculated and minimized as shown below:

$$L = \int_{\Omega} ||\Phi(t) - f(t)||dt \tag{3.5}$$

The SIREN architecture performs better than other implicit neural representation networks having different activation functions such as ReLu and ReLu with positional encoding.

This can also be shown in images or videos training by mapping (x, y) pixel coordinates or (x, y, t) pixel coordinates respectively to color values.

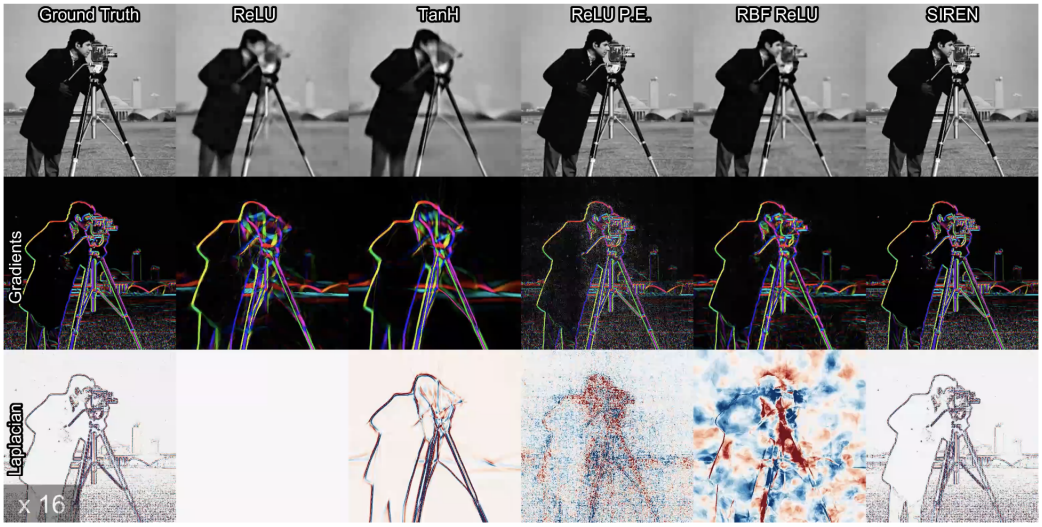

Figure 3.8: The Reconstruction of Images, Their Gradients and Their Second Derivatives by Architectures with Different Activation Functions (Sitzmann *et al.* (2020))

But when there is no access to the grayscale or RGB pixel values of the image and there is only access to the gradient values at each coordinate, reconstruction of the image in the primal domain can be achieved by supervising the derivative of SIREN with the gradient at that coordinate. We can implement this by understanding the concept of automatic differentiation in pytorch from Paszke *et al.* (2017). This helps train a SIREN function parameterizing the original image. The only difference in the loss function would be that the loss is calculated over the gradients of the Φ and f functions. The loss function can be written as:

$$L = \int_{\Omega} ||\nabla\Phi(t) - \nabla f(t)||dt \tag{3.6}$$

When SIREN is compared with architectures with other non-linear activation functions, it can be seen from the Fig. 3.8 that SIREN fits the gradient and laplacian

of the images quite well when compared to other architectures. SIREN can also be used to learn the Eikonal equation in a similar manner.

3.4.3 SIREN for Trajectory Prediction

Over the long run, it is desirable to learn priors over the space of implicit functions. In the case of images, in the paper by Sitzmann *et al.* (2020), it is tested on Celeba dataset. Here, the input is a set of pixels and an encoder produces a latent code from them. This latent code is then given as an input to the hypernetwork which then outputs the parameters of the respective SIREN representation of the image. The architecture thus learns the prior over the space of functions that parameterize many images or signals. This allows the reconstruction of images because the priors learnt by the hypernetwork allow it to generate the parameters of the corresponding SIREN network.

In the case of trajectory prediction, the input to the hypernetwork would be the latent vector encoded by the encoder and the hypernetwork would then translate it into parameters of the SIREN network which models the specific trajectory. The encoder takes the past trajectory as the input and produces a latent vector. This process of generating the parameters of SIREN network can also be viewed as sampling from a function space where the different functions correspond to different trajectories. The priors of the trajectories learnt by the hypernetwork play an important role in generating the implicit representation of the respective trajectory. Now, this network can be used to predict the values in the whole trajectory.

Chapter 4

EXPERIMENTS AND RESULTS

4.1 Implementation Details

Conditional variational autoencoder(CVAE) is implemented on various datasets for trajectory forecasting.

4.1.1 CVAE

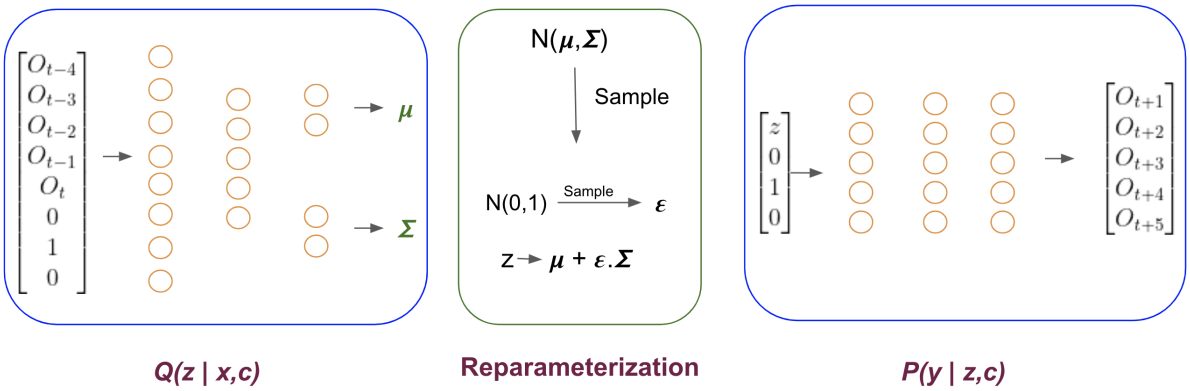

Figure 4.1: Generic Overview of Conditional Variational Autoencoder Architecture Implementation

As shown in the fig. 4.1, the network has 3 parts namely the encoder or $Q(z|x,c)$, sampling from the gaussian distribution latent space and the decoder or $P(y|z,c)$. The input to the network would be a set of observations $[O_t, O_{t-1}, ..O_{t-4}]$ and conditioned on a label which might be a number, vector or a one-hot encoded vector. Then the encoder outputs the mean and variance of the gaussian distribution to be considered for sampling in the latent space.

Now, the reparameterization trick is used to sample from a normal distribution using which we can indirectly sample from the gaussian distribution. This is because

the sampling operation from a gaussian distribution whose parameters are μ and Σ is not differentiable which would become a problem during backpropagation. Now this sample is given as input to the decoder which predicts the next set of observations in the trajectory $[O_{t+1}, O_{t+2}, ..O_{t+5}]$. It can be seen that a similar kind of approach taken in the work by Gomez-Gonzalez *et al.* (2020).

4.1.2 Loss function of CVAE

The loss function contains 2 main terms which are the reconstruction loss and regularization loss. In the case of trajectory prediction, the reconstruction loss is the mean squared loss(MSE) between the actual and predicted future trajectories y and $\hat{y}$ respectively while the regularization term is the KL-divergence(KL) loss between the gaussian distribution choice of the encoder $N(\mu, \Sigma)$ and normal distribution $N(0, I)$. The loss function is as shown below:

$$L = MSE(y, \hat{y}) + KL(N(\mu, \Sigma)||N(0, I)) \tag{4.1}$$

In the equation 4.2, $y, \hat{y}$ are the actual and predicted values.

4.1.3 SIREN

As shown in 4.2, the model has 3 components namely the convolutional image encoder, the hypernetwork and the SIREN network. The input to the network is an incomplete image of a letter in an unknown handwriting. The convolutional image encoder then encodes the image by propagating the image data through a series of convolutions and ReLu layers. The output from the encoder is then passed through a hpernetwork. The hypernetwork is a multi-layered perceptron. This hypernetwork then outputs the weights of the SIREN network.

The SIREN network can be viewed as the image trajectory function corresponding

to the input image. Thus, by passing the pixel coordinates to the SIREN network, the missing component or the incomplete part of the input image can be predicted. An important component in the training of this network is to generate sparse images from fully complete ones to train the network. These are generated with by masking the image with a mask generated by getting all the elements in the mask from bernoulli distribution. This allows some elements of the mask to be 0 and the others to be 1 which is used to perform element-wise multiplication with the original image to get the sparse image.

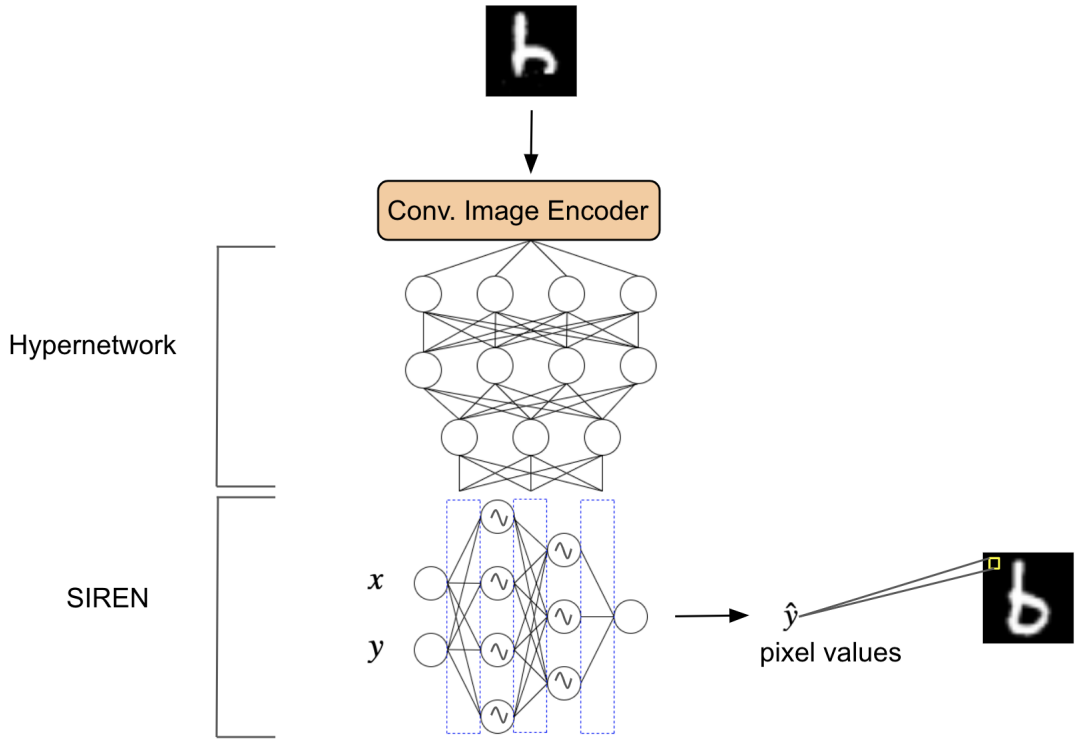

Figure 4.2: Generic Overview of Siren Architecture Implementation

4.1.4 Loss function of the SIREN-based model

The loss function contains 3 main terms namely the mean squared loss between the ground truth and predicted image pixel vectors, the squared loss of the weights of

SIREN predicted by the hypernetwork and the latent vector loss or the squared loss of the latent vector produced by the encoder. They correspond to the 3 components in the model. The loss function is as shown below:

$$L = MSE(y, \hat{y}) + \lambda_1 ||W_{siren}||_F^2 + \lambda_2 ||L_v||_2^2 \tag{4.2}$$

In the equation 4.2, $y, \hat{y}$ are the actual and predicted image pixel vector values and the first term is the mean squared loss between them while λ_1 and λ_2 are the weights of the frobenius and L2 regularization terms. W_{siren} and L_v are the weights of the siren network and the latent vector produced by the image encoder respectively.

4.2 Plots

4.2.1 Stock market price Prediction

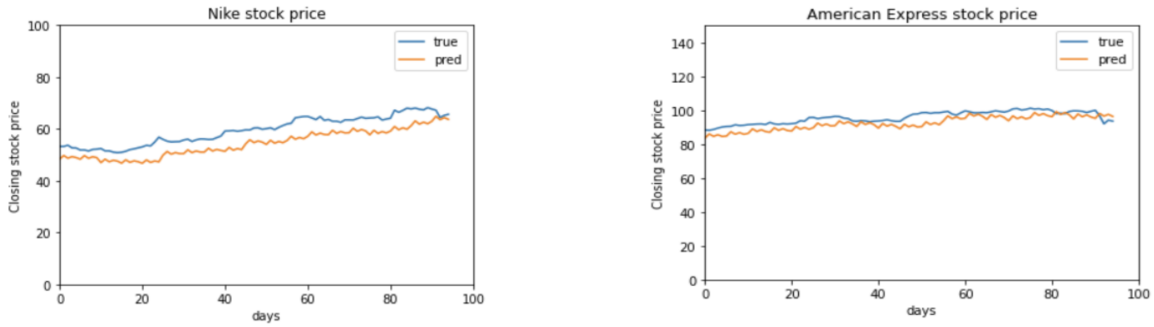

Figure 4.3: Nike and American Express Stock Price Prediction

Figure 4.4: Bank of America Stock Price Prediction

The dataset consisted of features of 500 companies taken from the S&P 500 Index. There are a total of 7 features representing each datapoint. Some of the features included are opening stock price, closing stock price, high, low etc. This dataset was taken from Kaggle. Then the 30 companies which have the Dow Jones Industrial Average were separated and used to train the CVAE model. The dataset contains 5 years of stock data of the companies from 2013 to 2018.

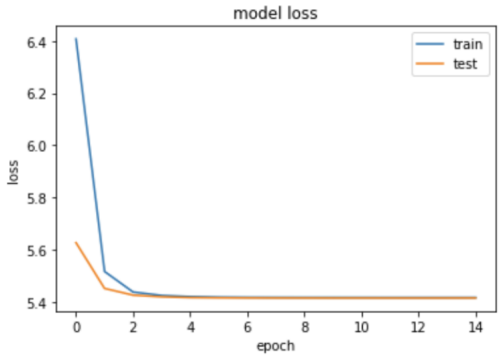

Figure 4.5: Model Loss Plot

After some feature engineering, some extra features were added and the resulting dataset was used for training and testing. A total of 11 features were used for training and testing the model. The encoder and the decoder of the CVAE model consisted of 3 layers each. The hidden layers had a total of 8 dimensions and the latent space size was taken as 4. The binary cross-entropy loss and the KL divergence loss added together made up the loss function which was optimized using the Adam optimizer. The input to the CVAE included features as well as the label which was the stock symbol identifier of the company. The data was divided into batches of batch size of 50 and was run for 40 epochs during training.

The train and test loss of the model are plotted and shown in Fig. 4.5. It can be seen from the figure that there is a fast convergence in the loss values. Fig. 4.4 shows the plots of model reconstructed(predicted) values of bank of america stocks as well as the true values of its stocks for the last 100 days in the dataset. The values

plotted are those of closing stock price values. We can see that the predicted curve is between the control limits of the other curve implying it to be reasonably good.

4.2.2 Sine curve trajectory prediction

The trajectory predictions in the stock market price prediction show that the model is able to detect the underlying trend in the closing stock price values and the predicted values is able to increase and decrease with the values. But it is still not very accurate since the stock market price fluctuations are too volatile and encoding their behavior is very tough. Hence, the models are tested on simple sine curve trajectories to see how good the models are in generating trajectories given a set of previous observations.

Here, the dataset consists of points on the sine wave 3° apart from each other. Now, the model is trained to predict the values of the next 60° of the sine curve given the previous 20° values of the curve. The values are taken from a large domain of sin(t) where t ranges from [-2000,2000]. Also, it is trained on sine curves of multiple frequencies. In the case of CVAE model, the model is conditioned on the frequency values. So, the input to the model are the data points concatenated with the frequency of the sine curve. This model performs with a high accuracy as seen from the Fig. 4.6, 4.7, 4.8.

Also, it is evident from Fig. 4.9 that during inference, if the same model is given the wrong frequency along with the correct datapoints as the input, it outputs a completely different trajectory. Since the frequency of sin(8t) is higher than sin(4t), the generated curve is also of higher frequency. This also shows that the model learns the underlying patterns associated with the frequencies in the latent space.

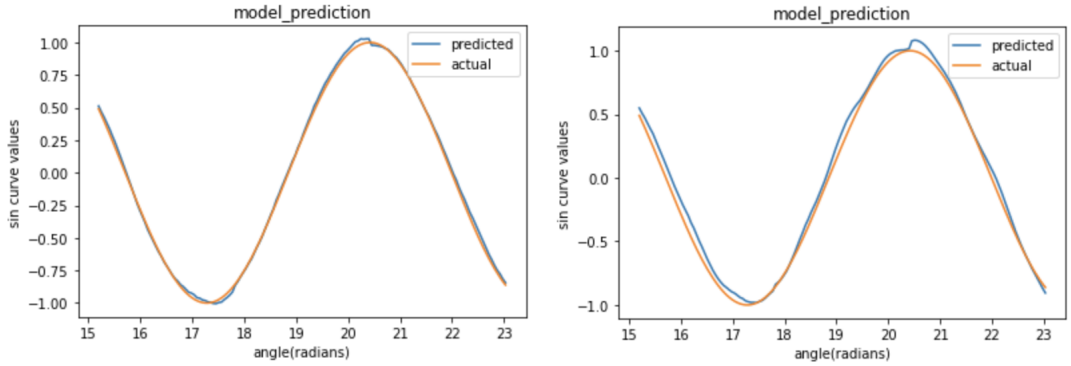

Figure 4.6: Sin(t) Curve Trajectory Predictions. The Orange Line Shows the Actual Trajectory While the Plot in Blue Is the Predicted Trajectory by Cvae. The Figure on the Left and Right Are Predictions Taking a Kl Divergence Loss Weightage Of .01 And .1 Respectively

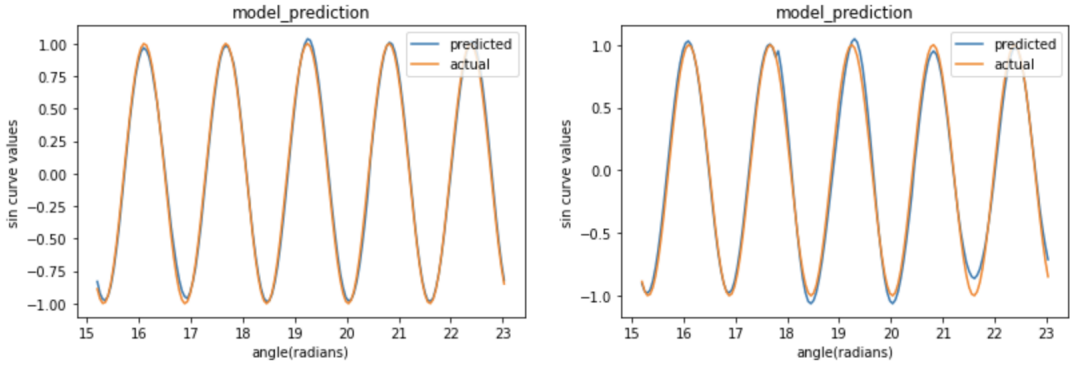

Figure 4.7: Sin(4t) Curve Trajectory Predictions. The Orange Line Shows the Actual Trajectory While the Plot in Blue Is the Predicted Trajectory by Cvae. The Figure on the Left and Right Are Predictions Taking a Kl Divergence Loss Weightage Of .01 And .1

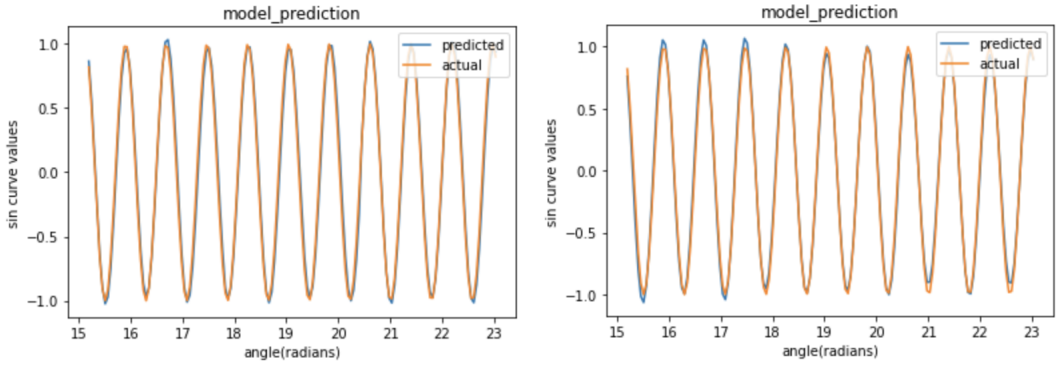

Figure 4.8: Sin(8t) Curve Trajectory Predictions. The Orange Line Shows the Actual Trajectory While the Plot in Blue Is the Predicted Trajectory by Cvae. The Figure on the Left and Right Are Predictions Taking a Kl Divergence Loss Weightage Of .01 And .1 Respectively

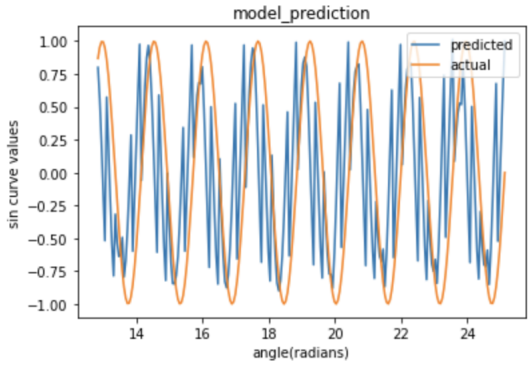

Figure 4.9: Sin(4t) Curve Trajectory Predictions by Cvae When Conditioned on a Wrong Class of 8 Instead of 4

4.2.3 Human Handwriting trajectory prediction

Finally, the models are tested on a handwriting dataset to learn the human behaviors. Using this learned behavior, future trajectories of human handwriting motion while writing letters are predicted. Since the model would predict the future writing trajectory, it is possible for the machine or the robot to help the human fill out the rest of the letter once the individual starts writing by grasping his behavior. Thus, this becomes a very good goal to achieve in human-robot interaction.

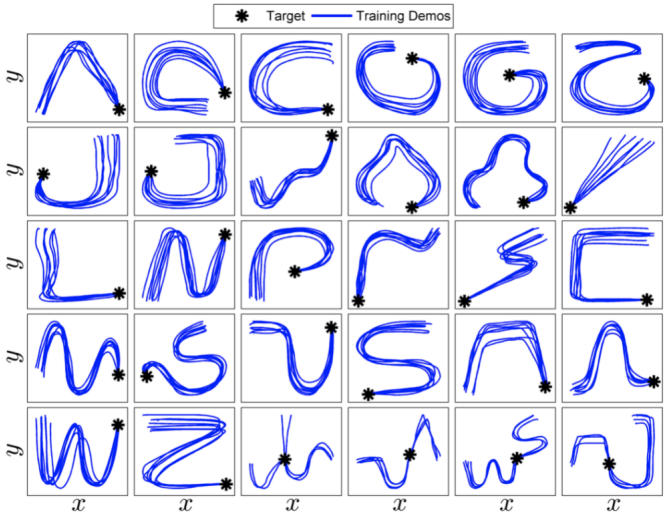

Figure 4.10: All the Different Letters and Curves in the Lasa Handwriting Dataset

The CVAE model is trained on the Lasa handwriting dataset shown in Fig. 4.10.

This dataset consists of letters and other curves each letter having 7 different behavioral trajectories. Each curve has 1000 datapoints or coordinates and the model is trained such that it takes 20 points as input to predict the next 60 points or coordinates thus generating a trajectory. The individual corresponding to the letter writing is encoded in the form of a number and the model is conditioned on the individual by also giving this number as the input to the model. is The loss function of the CVAE model then will have 3 terms as shown below:

$$L = MSEloss + MSE_vloss + KLloss \tag{4.3}$$

The first term $MSEloss$ is the reconstruction loss which is also the mean squared loss between the predicted and actual future trajectory. The second term MSE_vloss is the mean squared loss between the last point in the trajectory input and the first point in the output trajectory. This is important because this term ensures the smooth continuity of the future trajectories without being broken. The third term $KLloss$ is the regularization loss which ensures a smooth future trajectory.

Fig. 4.11, 4.12, 4.13 show the predictions of the model on different letters on 2 different human behaviors. It can be seen that the model is able to generate trajectories quite well for different human writings.

The SIREN-based model is trained on the Extended-MNIST(EMNIST) dataset shown in Fig 4.14. This dataset consists of digits and letters. This was published by Cohen *et al.* (2017).

The dataset is derived from the NIST special database 19 as shown in 4.14. It contains handwriting letters of 3600 people. Handwritten letters were extracted from the EMNIST dataset which were used to learn the handwriting behaviors of the people in general helping to understand the underlying human behavior which can then be used to fill the letters in a way as similar as possible to the individual writing the

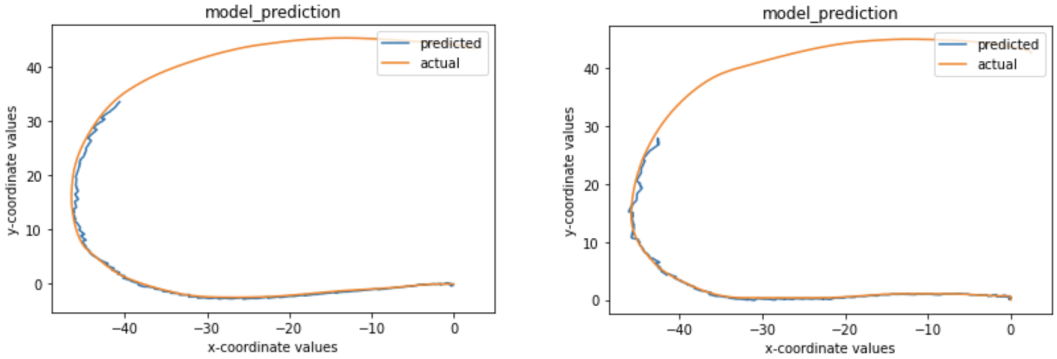

Figure 4.11: C Shape Curve Trajectory Predictions of 2 Different Behaviors on the Left and Right Respectively.The Blue Curve Is the Predicted Trajectory While the Orange Curve Is the Actual Human Motion Trajectory

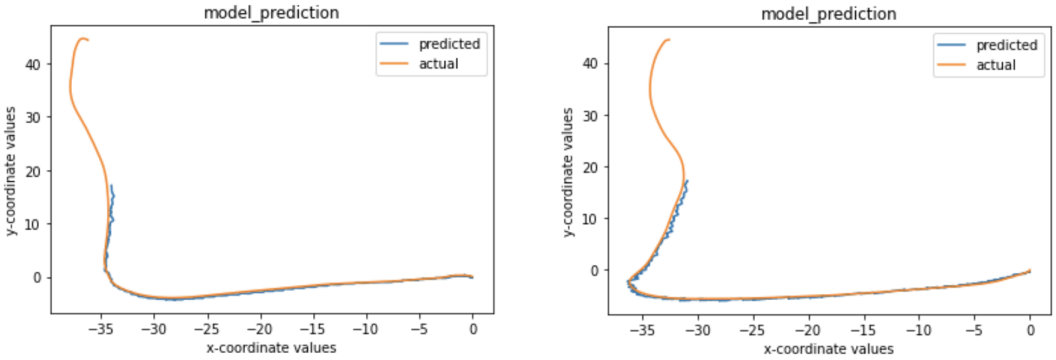

Figure 4.12: L Shape Curve Trajectory Predictions of 2 Different Individual Behaviors on the Left and Right Respectively. The Blue Curve Is the Predicted Trajectory While the Orange Curve Is the Actual Human Motion Trajectory

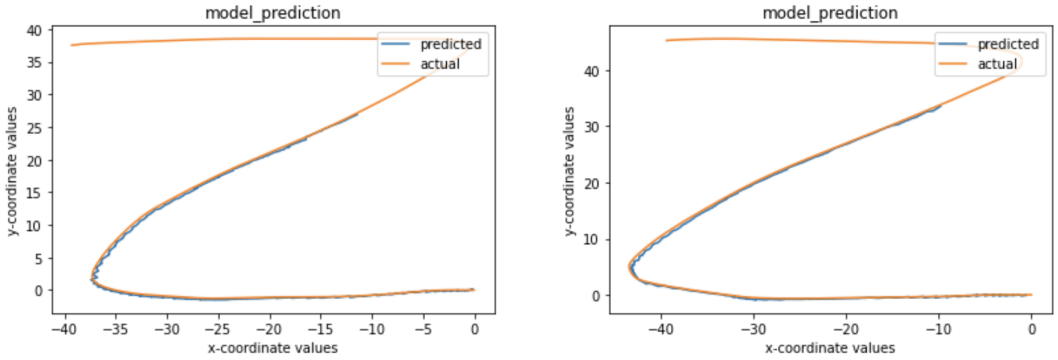

Figure 4.13: Z Shape Curve Trajectory Predictions of 2 Different Individual Behaviors on the Left and Right Respectively. The Blue Curve Is the Predicted Trajectory While the Orange Curve Is the Actual Human Motion Trajectory

letter.

In Fig. 4.15, the 3 pairs of images are the input images to the model and the

Figure 4.14: Depiction of Some Letters in the EMNIST Dataset (Cohen *et al.* (2017))

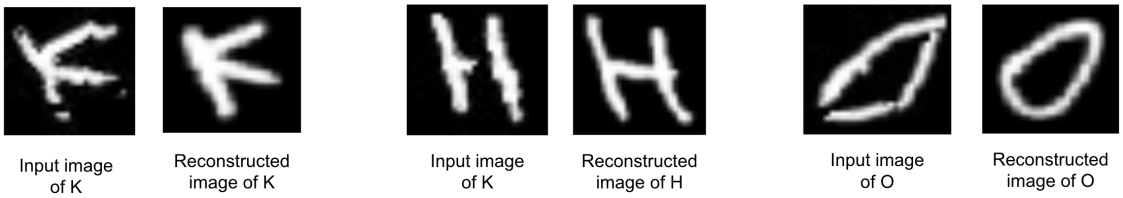

Figure 4.15: Reconstructed Letters from Sparse Images by Siren-based Model

model output on the left and right respectively. It can be seen that the sparse images are reconstructed quite well though not perfectly . The predicted image of K in the first pair is quite thick in stroke but it still gets the angle, font and a unique style from the image to its left. The reconstructed H image on the other hand is quite sharp and the angle also matched the sparse image input. This is also true in the case of the letter O. This especially highlights the capability of the model to learn the angle of the letter very accurately.

4.16 contains 5 sets of test cases where partially drawn images of a completely new individual are given as input and the model tries to predict the letter the person wants to draw and outputs the supposed complete image of the letter the user wants to draw. It can be observed that the model is quite good at predicting what the letter is by understanding the human writing and trying to replicate the style of the

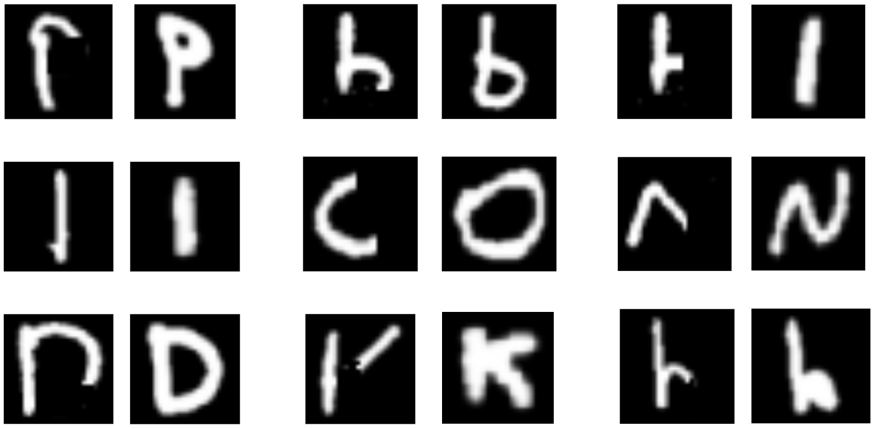

Figure 4.16: Predicted Letters from a Different User's Partially Written Letter Images by Siren-based Model

person drawing it. But from the right most pair of images, it can be seen that the predictions are strongly dependent upon how long of a future trajectory it can predict. It is expected that the model would predict and output the letter h or b. Instead, it just predicted it as the letter I.

It can also be observed that the prediction in the centre could have been the letter C or the letter O. But the model predicts it as an O. Also, in the last prediction, the actual letter to be predicted is the lower case letter h. But the model predicts it as something very unclear between h and the lower case letter b. In general, it is seen that the models generates outputs which are blurred as can be seen for almost all the predictions except one or two predictions. Thus, the model predictions are not very accurate and the sparsity of the input image plays a key role to get accurate predictions.

Chapter 5

DISCUSSION AND FUTURE WORK

This work has used and compared the performance of conditional variational autoencoders and SIREN models for trajectory prediction and reconstruction. For this, human hand writing motion trajectories are generated. In general, in real-life scenarios, people write completely or partially in cursive style. But the proposed models are not yet tested on whole cursive words and sentences which is a potential future direction in testing and improving the models. This would also help in knowing how deeply the models can encode the human behavior from the given data.

Another future direction of work would be to modify the SIREN-based model in which it learns priors in a better and more accurate manner. This can especially be seen as a way of figuring out a more systematic and maybe even a parametric approach in controlling how the model samples from the SIREN function space. There is presently no method interfering or intervening in this process of sampling.

Finally, the current model still has fidelity issues and doesn't generate accurate parameters of the SIREN network consistently. One of the ways to improve this would be researching on the function sampling process as mentioned previously. Alternately, the encoding process could also be improved. Currently, the SIREN-based model encodes the human writing sample using a convolutional image encoder which is connected to a hypernetwork followed by a SIREN network. So, the type of encoding plays a major role in getting the correct predictions from the model. A more sophisticated encoder model might be preferred in such cases.

REFERENCES

Alahi, A., K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", (2016).

Berg, J. and K. Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries", Neurocomputing **317**, 28–41 (2018).

Bilan, I. and B. Roth, "Position-aware self-attention with relative positional encodings for slot filling", arXiv preprint arXiv:1807.03052 (2018).

Chen, R. T., Y. Rubanova, J. Bettencourt and D. Duvenaud, "Neural ordinary differential equations", arXiv preprint arXiv:1806.07366 (2018).

Cohen, G., S. Afshar, J. Tapson and A. Van Schaik, "Emnist: Extending mnist to handwritten letters", in "2017 International Joint Conference on Neural Networks (IJCNN)", pp. 2921–2926 (IEEE, 2017).

Cui, H., V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks", in "2019 International Conference on Robotics and Automation (ICRA)", pp. 2090–2096 (IEEE, 2019).

Dong, B., H. Liu, Y. Bai, J. Lin, Z. Xu, X. Xu and Q. Kong, "Multi-modal trajectory prediction for autonomous driving with semantic map and dynamic graph attention network", arXiv preprint arXiv:2103.16273 (2021).

Gomez-Gonzalez, S., S. Prokudin, B. Schölkopf and J. Peters, "Real time trajectory prediction using deep conditional generative models", IEEE Robotics and Automation Letters **5**, 2, 970–976 (2020).

Ivanovic, B. and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs", in "Proceedings of the IEEE/CVF International Conference on Computer Vision", pp. 2375–2384 (2019).

Kingma, D. P. and M. Welling, "Auto-encoding variational bayes", arXiv preprint arXiv:1312.6114 (2013).

Ma, Y., X. Zhu, S. Zhang, R. Yang, W. Wang and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 33, pp. 6120–6127 (2019).

Mescheder, L., M. Oechsle, M. Niemeyer, S. Nowozin and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 4460–4470 (2019).

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in pytorch", (2017).

Phan-Minh, T., E. C. Grigore, F. A. Boulton, O. Beijbom and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 14074–14083 (2020).

Rhinehart, N., R. McAllister, K. Kitani and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings", in "Proceedings of the IEEE/CVF International Conference on Computer Vision", pp. 2821–2830 (2019).

Salzmann, T., B. Ivanovic, P. Chakravarty and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data", in "Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16", pp. 683–700 (Springer, 2020).

Sitzmann, V., J. Martel, A. Bergman, D. Lindell and G. Wetzstein, "Implicit neural representations with periodic activation functions", Advances in Neural Information Processing Systems **33** (2020).

Sohn, K., H. Lee and X. Yan, "Learning structured output representation using deep conditional generative models", in "Advances in neural information processing systems", pp. 3483–3491 (2015).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is all you need", in "Advances in neural information processing systems", pp. 5998–6008 (2017).

Vemula, A., K. Muelling and J. Oh, "Social attention: Modeling attention in human crowds", in "2018 IEEE international Conference on Robotics and Automation (ICRA)", pp. 4601–4607 (IEEE, 2018).