Writemate Project Report

Mohammed Bhadsorawala,Sanika Kumbhare
Mentored by Lakshaya Singhal and Advait Dhamorikar

November 2023

Contents

1 Introduction

This document gives an overview of what we have learned,what we have implemented,what challenges we faced and how we solved them during the two month **EKLAYVA** mentorship programme organized by **SRA VJTI** ***(SOCIETY OF ROBOTICS AND AUTOMATION,Veermata Jijabai Technological Institute)***

We are extremely grateful to our mentors **Lakshaya Singhal** and **Advait Dhamorikar** for their support and guidance throughout the duration of the project.We can never thank them enough for how much they have helped us and how much we have learned from them.
Thank You for teaching us how to learn.

Abstract

Section 1 describeswhat we have learned about Linear Algebra,git and Github.Section 2 talks about Deep learning,Standard Neural Networks and implementing it on MNIST Handwritten digit recognition.Section 3 emphasizes on Convolutional Neural Networks,relevent papers in that domain and implementing Facial Emotion recognition model using the same.Section 4 is based on Recurrent Neural Networks,its varients,Attention mechanism and why it works. Section 5 explains implementation of *Writemate, a model which converts input text into Handwriting.*

2 Git and Linear Algebra

Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers who are collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows. Wait Wait now What is version control? **Version control** is a system that records changes to a file or set of files over time so that you can recall specific versions later. If you are a graphic or web designer and want to keep every version of an image or layout (which you would most certainly want to), a Version Control System (VCS) is a very wise thing to use. It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS also generally means that if you screw things up or lose files, you can easily recover. In addition, you get all this for very little overhead.

Linear algebra is built on two basic elements, the matrix and the vector.
What is a vector:
Physics student: Arrow with a length and direction
CS student: A list consisting of numbers [x y z]
Math student: Well it's both of the above.Both the forms are interconvertible
BASIS: Set of vectors of unit magnitude which are scaled by the scalars .Any vector in plane or space can be represented by linear combination of scaled basis. For v=xi+yj here i and j are basis and scaled by x and y respectively to form vector v
SPAN: Set of vectors that can be obtained by linear combination of basis.

For any 2 basis that are not parallel the span is whole 2D space For any 3 basis such that they are linearly independent is whole 3D space If three basis are such that no vector can be expressed as linear combination of two or does not lie in the plane formed by the other two,they are said to be linearly independent Transformation of any vector can be described by the transformation of the basis of the system only.Transformation of the basis vectors is represented by a matrix each column correspond to co ordinates of resultant basis.
MATRICES GIVES A LANGUAGE TO DESCRIBE THESE TRANSFORMATIONS

So why did we learn about Linear Algebra?? Machines or computers only understand numbers. And these numbers need to be represented and processed in a way that lets machines solve problems by learning from the data instead of learning from predefined instructions (as in the case of programming).All types of programming use mathematics at some level. Machine learning involves programming data to learn the function that best describes the data.The problem (or process) of finding the best parameters of a function using data is called model training in ML.

Therefore, in a nutshell, machine learning is programming to optimize for the best possible solution – and we need math to understand how that problem is solved. The first step towards learning Math for ML is to learn linear algebra. Linear Algebra is the mathematical foundation that solves the problem of representing data as well as computations in machine learning models

3 Deep Learning and Neural Networks

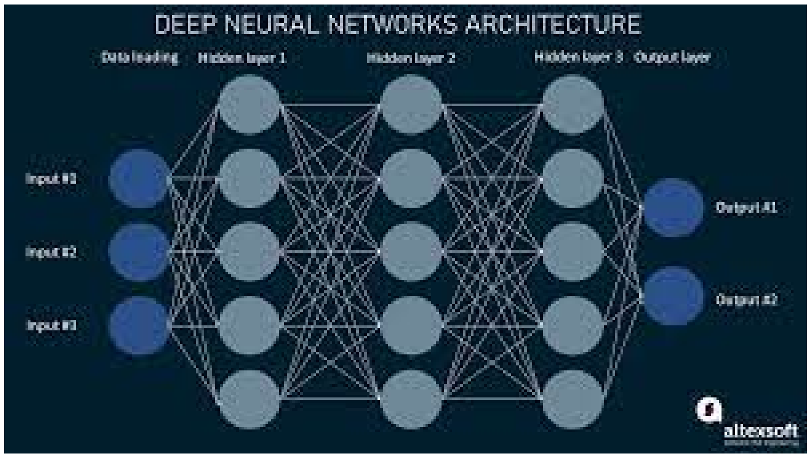
Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.
Deep learning is said to be analogous to human brain but how a neuron in brain computes a decision and how it learns is still uncertain, receiving electrical signals from other neurons and sending signals ahead.
Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars). In neural network each node is called as a NEURON, takes input from previous step and computes some function and gives output to further neurons
FACTORS LEADING TO BETTER FUNCTIONING OF NEURAL NETWORKS:

1. Increased data:better training

2. Computational speed: With faster speed,correction and optimization becomes faster and productivity increases

3. Better algorithms: increase computational speed

Machine learning and deep learning models are capable of different types of learning as well, which are usually categorized as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning utilizes labeled datasets to categorize or make predictions; this requires some kind of human intervention to label input data correctly. In contrast, unsupervised learning doesn't require labeled datasets, and instead, it detects patterns in the data, clustering them by any distinguishing characteristics. Reinforcement learning is a process in which a model learns to become more accurate for performing an action in an environment based on feedback in order to maximize the reward.



For learning,We did Andrew Ng Deep Learning Specialization courses on Coursera.In Course 1,we learned about standard deep neural networks,the notation used,vectorization used for computation,forward propagation,backward propagation,taking derivatives of varaibles and optimizing model's parameters. In course 2,we learned about how to tune the hyperparamaters of our network,various techniques for regularization,better optimization technique than standard gradient descent and techniques to speed up the training process of the model.

Creating a Handwritten digit recognition model

Using the knowledge we gained from neural networks and under the guidance of our mentors,we developed a 3 layer dense neural network from scratch first in numpy and later in JAX to train the network much more faster on GPUs and we used MNIST dataset consisting of 70000 images,60000 used for training and 10000 used for testing The dataset is termed at an easy start for creating more complex NN architectures further.
We acheived an accuracy of 98.093% on training set after 2000 iterations and 97.19% on the test set.

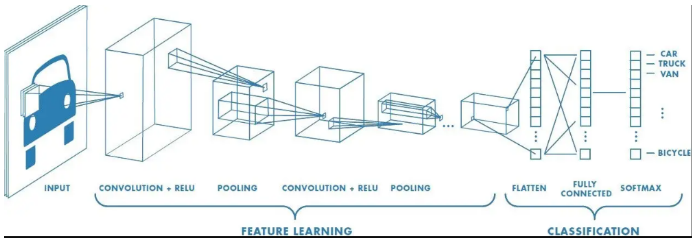


Figure 1: CNN architecture

4 Convolutional Neural Networks

In mathematics (in particular, functional analysis), convolution is a mathematical operation on two functions (f and g) that produces a third function ( ) that expresses how the shape of one is modified by the other.
A **Convolutional Neural Network (CNN)** is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data. They are predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers. The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.Convolutional Layers are used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred ad feature maps.

After going through course 4 on CNN, we read papers of AlexNet and VGG CNN architecture.Then we implemented a Facial Emotion recognition model based on ResNet architecture using Fer2013 dataset to get image data for training and testing.
On Resnet 50 after 30-35 epochs we achieved around 50% accuracy and ResNet 18 accuracy was also capped to around 50% but in fewer epochs

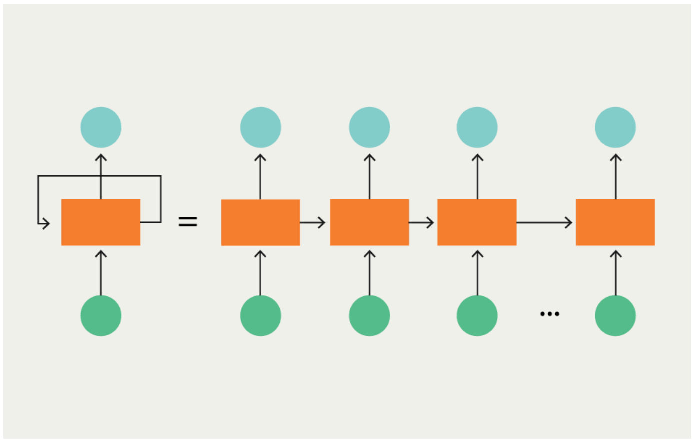


Figure 2: RNN architecture

5 Recurrent Neural Networks

A **recurrent neural network (RNN)** is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate. Like feedforward and CNNs, recurrent neural networks utilize training data to learn. They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. While future events would also be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.

Another distinguishing characteristic of recurrent networks is that they share parameters across each layer of the network. While feedforward networks have different weights across each node, recurrent neural networks share the same weight parameter within each layer of the network. That said, these weights are still adjusted in the through the processes of backpropagation and gradient descent to facilitate reinforcement learning.

RNNs have many varient architectures, we would here discuss LSTMs breifly If the previous state that is influencing the current prediction is not in the recent past, the RNN model may not be able to accurately predict the current state. **LSTM** Long-Short-term-Memory cells have "cells" in the hidden layers of the neural network, which have three gates–an input gate, an output gate, and a forget gate. These gates control the flow of information which is needed to predict the output in the network. LSTM is a type of RNN with higher memory

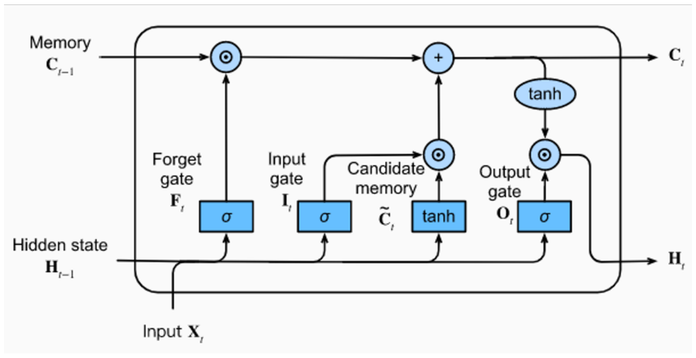

Memory
$\mathbf{C}_{t-1}$
$\mathbf{C}_t$
tanh
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Candidate memory $\tilde{\mathbf{C}}_t$
Output gate $\mathbf{O}_t$
σ
tanh
Hidden state $\mathbf{H}_{t-1}$
$\mathbf{H}_t$
Input $\mathbf{X}_t$

Figure 3: LSTM architecture

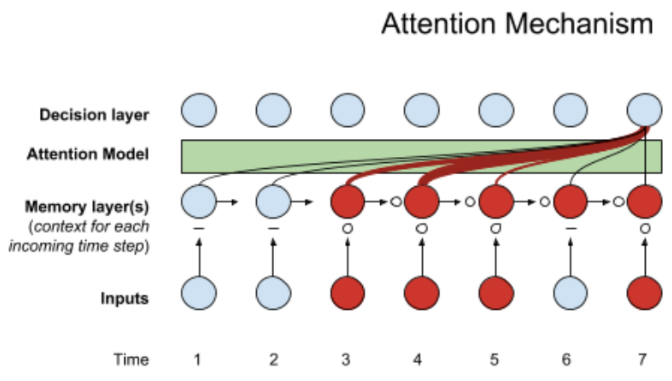


Figure 4: Attention Mechanism

power to remember the outputs of each node for a more extended period to produce the outcome for the next node efficiently.

Attention mechanism

The attention mechanism was introduced to improve the performance of the encoder-decoder model for machine translation. The idea behind the attention mechanism was to permit the decoder to utilize the most relevant parts of the input sequence in a flexible manner, by a weighted combination of all the encoded input vectors, with the most relevant vectors being attributed the highest weights.The attention mechanism addresses the bottleneck problem that arises with the use of a fixed-length encoding vector, where the decoder would have limited access to the information provided by the input. This is thought to become especially problematic for long and/or complex sequences, where the dimensionality of their representation would be forced to be the same as for shorter or simpler sequences.

6 WriteMate

Afer learning all these topics,We had to create a sequential model which takes in text input and prints handwriting for the same.We refered to Alex Grave‘s paper, Generating Sequences with Recurrent Neural Networks.
The paper described the architecture and dataset to be used to train the model on handwriting data.We used Online handwriting data and online in this context means that the writing is recorded as a sequence of pen-tip locations. IAM-OnDB consists of handwritten lines collected from 221 different writers using a ‘smart whiteboard’. The writers were asked to write forms from the Lancaster-Oslo-Bergen text corpus, and the position of their pen was tracked using an infra-red device in the corner of the board.

The original input data consists of the x and y pen co-ordinates and the points in the sequence when the pen is lifted off the whiteboard.IAM-OnDB is divided into a training set, two validation sets and a test set, containing respectively 5364, 1438, 1518 and 3859 handwritten lines taken from 775, 192, 216 and 544 forms.We treat each line as a separate sequence (meaning that possible dependencies between successive lines were ignored). In order to maximise the amount of training data, we used the training set, test set and the larger of the validation sets for training and the smaller validation set for early-stopping. The lack of independent test set means that the recorded results may be somewhat overfit on the validation set; however the validation results are of secondary importance, since no benchmark results exist and the main goal was to generate convincing-looking handwriting. The principal challenge in applying the prediction network to online handwriting data was determining a predictive distribution suitable for real-valued

6.1 Mixture Denisty Output

The idea of mixture density networks is to use the outputs of a neural network to parameterise a mixture distribution. A subset of the outputs are used to define the mixture weights, while the remaining outputs are used to parameterise the individual mixture components.Mixture density network are trained by maximising the log probability density of the targets under the induced distributions.
Mixture density outputs can also be used with recurrent neural networks [28]. In this case the output distribution is conditioned not only on the current input, but on the history of previous inputs. Intuitively, the number of components is the number of choices the network has for the next output given the inputs so far.

The intuition behind using mixture density output instead of softmax or any other activation function is that when we are prediciting handwriting strokes,it can be anywhere in the proximity of current stroke so we mixture of components,in our case 20,where each component has a normal distribution and predicts a part of our task.

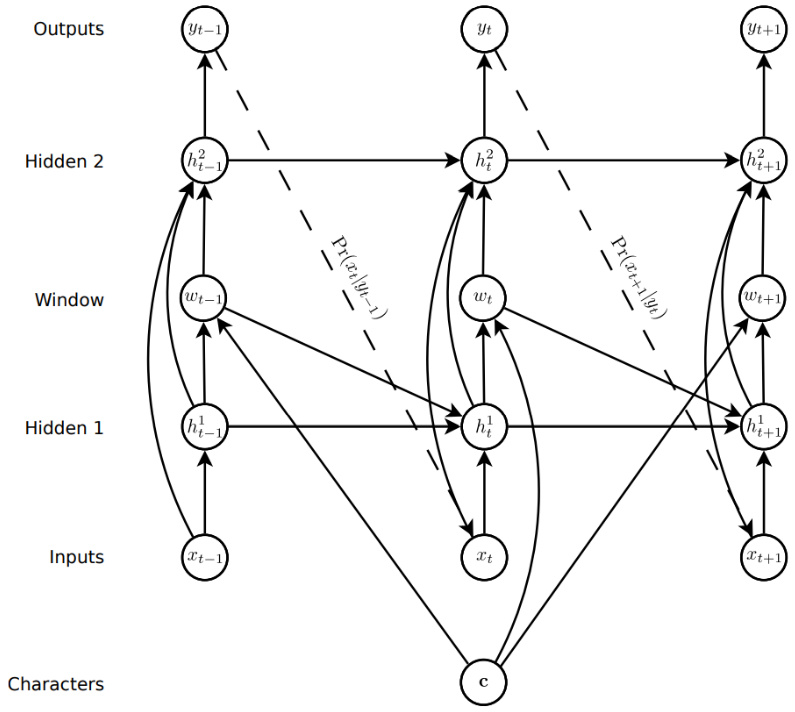
6.2 Architecture

Each point in the data sequences consisted of three numbers: the x and y offset from the previous point, and the binary end-of-stroke feature. The network input layer was therefore size 3. The co-ordinate offsets were normalised to mean 0, std. dev. 1 over the training set. 20 mixture components were used to model the offsets, giving a total of 120 mixture parameters per timestep (20 weights, 40 means, 40 standard deviations and 20 correlations). A further parameter was used to model the end-of-stroke probability, giving an output layer of size 121. The network architectures consists of 3 layers of 400 LSTM cells.The network was retrained with adaptive weight noise, with all std. devs. initialised to 0.075. Training with fixed variance weight noise proved ineffective, probably because it prevented the mixture density layer from using precisely specified weights.

This works for random Handwriting prediction but for sampling Handwriting conditioned on text input,we need to add a layer for passing text as input to our network.The main challenge in conditioning the predictions on the text is that the two sequences are of very different lengths (the pen trace being on average twenty five times as long as the text), and the alignment between them is unknown until the data is generated. This is because the number of co-ordinates used to write each character varies greatly according to style, size, pen speed etc.

So a **'soft window'** is convolved with the text string and fed in as an extra input to the prediction network. The parameters of the window are output by the network at the same time as it makes the predictions, so that it dynamically determines an alignment between the text and the pen locations. Put simply, it learns to decide which character to write next.

Here we used the knowledge of Attention model where we apply a soft window after the first LSTM layer and the window for each ouput from !st LSTM evaluates the attention that should be given to each part in the input sequence.Output of this window is passed to second and third LSTM layer subsequently.Below figure shows the architecture of our model including LStM layers,soft window and skip connections.


Outputs
Hidden 2
Window
Hidden 1
Inputs
Characters
y_{t-1}
y_t
y_{t+1}
h^2_{t-1}
h^2_t
h^2_{t+1}
w_{t-1}
w_t
w_{t+1}
h^1_{t-1}
h^1_t
h^1_{t+1}
x_{t-1}
x_t
x_{t+1}
$\mathbf{c}$
$\Pr(x_t|y_{t-1})$
$\Pr(x_{t+1}|y_t)$

6.3 Implementation

Our project was to build a model from scratch in JAX which is build over and is very similar to numpy.We created the model for handwriting prediciton in jax writing from scratch code for LSTM layers, MDN and loss and for back-propagation we used *jax.grad* which takes input a function calling the model and returning the loss, and grad returns the gradients of loss with respect to all parameters of the model

Later when starting with synthesis network, we used tensorflow,a deel learning framework which makes creating models very easy,giving readymade functions to call various architectures.
We faced problems while writing code for Attention at it was not straight forward and was not mentioned in paper.
Later we decided to define a custom class for the model writing custom functions for layers in tensorflow.We refered several repositories implementing the model on Github.
Yet when we wrote the code for ourselves, we got errors related to architecture.

6.4 Future work

Now we are working on stabilizing our code and once its ready,we would train our model and check its accuracy and reliability.
We think of making a user freindly interface for interacting with our model and viewing results.Also with synthesis can be extended to generate various beautiful handwritings and calligraphy.

7 References

Coursera courses on Deep Learning by Andrew Ng
Generating Sequences With Recurrent Neural Networks Paper by Alex Graves
Our project github Repository
Our reference repository for making our project
Clicking above links redirect to mentioned references