University of Wolverhampton
Department of Mathematics and Computer science

Master's Degree in Artificial Intelligence
Academic Year 2021/2022

Multi-Modal Deep Learning With Local Credit Assignment

Supervisor:
Prof. Adeel Ahsan

Student:
Nazeh Abel

Academic Year 2021/2022

I declare that the undersigned as well as the author of document is responsible for its content, and for the parts taken from other works, these are expressly declared citing the sources.

Acknowledgement

I am thankful to professor Adeel Ahsan for his simulating guardians and support during the development and experiments of this study and to Leandro Aparecido Passos Junior, Ahmed Khubaib, Mohsin Naseem for their valuable discussions and support.

Contents

Abstract

Deep learning has brought a significant contribution to the world of artificial intelligence inspired and motivated by neurophysiology. However, there is serendipity such as whether learning could be achieved in the human brain, computationally efficient algorithms of learning, etc. To simulate the mammal brain, gradient-based learning is adopted where each neuron plays its role in adjusting the error and the weight in a manner that can influence the output result of the model which shadows the back-propagation implementation in deep learning by weight sharing or back-propagation popularly known as a credit assignment problem. However, neuroscience learning models tend to share their information such as errors across layers effectively without weight sharing being necessitated. This study is an extension of the model of Guergiuev et al. **(Guergiuev et al., 2016)** a deep learning algorithm that uses multi-compartment neurons in optimizing the neocortex cost function in solving the credit assignment problem.

The neurons in this model feed on the sensory information and high-order feedback in the electronically segregated compartments of the model just like **(Guergiuev et al., 2016)** which allows the neurons in different layers to update their synaptic weights in harmony thus allowing the model to make a classification or to predict its target. The learning ability is affected by the degree of model dendritic segregation, and the form of synaptic feedback connection on the model. Furthermore, this will demonstrate how the model Towards deep learning with segregated dendrites **(Guergiuev et al., 2016)** can be extended into a multi-model multi-layer architecture to pinpoint practical higher-order representations—the characteristic of deep learning.

1 Introduction

The world is full of complexities starting from humans down to the environment and that makes it difficult for machines to understand and interpret such complexities. The brain processes information through powerful representation in multiple layers of neurons **(T. P. Lillicrap et al., 2014)**. One of the ways that machines can learn these patterns is through deep learning. Deep learning is an artificial intelligence technique that employs multiple layers of processing units which are designed to take advantage of these multi-layer network architectures to generate a hierarchical representation that allows each ensuing layer to determine the increasingly abstract, pertinent variables for a given task **(LeCun et al., 2015,Bengio and LeCun, 2007, Blake A. R, et al. 2016).** Deep learning methods are representational-learning techniques with multiple levels of representations achieved by composing simple but non-linear modules that each transform the representation at one level (raw input) into a higher and more abstract layer **(LeCun et al., 2015)**. The training of a deep neural network requires error back-propagation via a feedback pathway which allows the hidden layer to communicate error information and how it is calculated effectively **(Yang et al., 2021, Alber et al., 2018)**. Deep learning generates representations which filter relevant information related to the assigned task while ignoring extraneous information **(Yang et al., 2021).** The network learns how to assign blame using the random synaptic weights and this allows the network to extract useful information via the signal transported from the random weight connections. In deep learning each neuron is assigned a credit that allows it to contribute to the final output **(Bengio et al., 2015; Lillicrap et al., 2016, Yang et al., 2021)** known as the credit assignment problem. However, in neuroscience, there is still a major challenge on how conventional error back-propagation is demonstrated in the memory akin to deep learning **(Guergiuev et al., 2016, Yang et al., 2021, credit20)** as there is still a gap in the knowledge of deep learning and how memory and learning in neuroscience **(Guergiuev et al., 2016).**

The credit assignment problem is due to its difficulties in a multi-layer network as the result of the hidden layer neuron's dependency on the downstream synaptic connections **(Guergiuev et al., 2016)** which take advantage of the multi-layer architectural design in developing complex real-world solutions **(LeCun et al., 2015; Bengio et al., 2015)**. However, this problem has gained less attention in the field of neuroscience which is associated to function history of biological studies of synaptic plasticity **(Guergiuev et al., 2016).** Deep learning tends to mimic

the pattern and activities that occur in the human brain, however, not all features of deep learning are consistent with how the brain works due to the simplicity of the neurons in deep learning.

Artificial neurons with two compartments similar to roots and branches were designed by **(Guerguiev Jordan, 2016)**, to simulate how deep learning can be biologically feasible. And the model learned and make classification on the MNIST dataset and in a similar manner **(Guergiuev et al., 2016)** designed a neural network that was built on **(Guerguiev Jordan, 2016)** process. This study will evolve to build its shape on the process of Guergiuev et al. **(Guergiuev et al., 2016)** proposed model Towards deep learning with segregated dendrites which demonstrated how multi-compartment neurons can be used in deep learning by adopting electronical segregated compartments higher-order feedback to receive sensory information which allows it to coordinate synaptic weight updates and identify userful high-order representations.

2 Network Architecture and Algorithm

Supervised deep learning techniques' is a learning procedure that leverage local weight updates expecting every neuron to receive signals that can be used as their credit signal to determine the output of the model **(Guergiuev et al., 2016).** The error back-propagation algorithm is the basis that is used in credit assignment **(Rumelhart et al., 1986, Werbos, 1982, Linnainmaa, 1976)** The back-propagation requires an exact symmetric between the forward and feedback weight pathways known as weight problem transport. However, some studies have proved that this can be mitigated in supervised learning tasks by using random weight feedback **(T. P. Lillicrap et al., 2014).**

The model in this study was built on the inspirations of **(Guergiuev et al., 2016)** on how biology and machine learning interwind. In demonstrating the learning ability of the model, the MNIST dataset was employed. The dataset consists of handwritten digits images divided into 10 categories i.e 0–9, and each image has an associated label. The dataset is composed of 70,000 instances of 28x28 pixels each, out of which 60,000 were used as training set and 10,000 as the testing sets.

Additionally,to test if the concept of multi-modal architecture could work on the model of **(Guergiuev et al., 2016)** a shallow network of a single hidden layer was implemented using sparse feedback weight and the prediction performance of 6.10% on average was achieved.

The MNIST data input was used in setting up the spiking rate of $\ell = 392$ Poisson point process in the input layer as against the model of **(Guergiuev et al., 2016)** which is then projected to 500 neurons (m) on the hidden layer which is composed of an apical compartment with a vector of $V^{0a}(t)$= $[V_1^{0a}(t)$, $V_2^{0a}(t)$,...,$V_m^{0a}(t)]$; Basal compartment with a voltage vector of $V^{0b}(t) = [V_1^{0b}(t)$, $V_2^{0b}(t)$,...,$V_m^{0b}(t)]$; and Somatic compartments with a voltage vector $V^0(t)$= $[V_1^0(t)$, $V_2^0(t)$,...,$V_m^0(t)]$ similar to the model of **(Guergiuev et al., 2016)** on each of the sub-network.

,

$$\frac{dV_i^0(t)}{dt} = V_i^0(t) + \frac{gb}{gl}(V_i^{0b}(t) - V^0(t)) + \frac{ga}{gl}(V_i^{0a}(t) - V^0(t)) \tag{1}$$

The voltage in the dendrites compartment is the weighted mean of the last spike trains to the i^{th} hidden layer neuron:

$$V_i^{0b}(t) = \sum_{j=1}^{l} W_{ij}^0 s_j^{input}(t) + b_i^0 \tag{2a}$$

$$V_i^{0a}(t) = \sum_{j=1}^{n} Y_{ij} s_j^1(t) \tag{2b}$$

where the W_{ij}^0 and Y_{ij} are the synaptic weight from the input and output layer respectively. additionally, b_i^0, s^{input}, s^1 are the bias and the filtered spike trains of the input and output layer neurons respectively as it's in the model of **(Guergiuev et al., 2016).**

Parameter	Units	Value	Description
dt	ms	1	Time step resolution
ϕ_{max}	Hz	200	Maximum spike rate
τs	ms	3	Short synaptic time constant
τL	ms	10	Long synaptic time constant
Δts	ms	30	Settle duration for calculation of average voltages
g_b	S	0.6	Hidden layer conductance from basal dendrites to the soma
g_a	S	0, 0.05, 0.6	Hidden layer conductance from apical dendrites to the soma
g_d	S	0.6	Output layer conductance from dendrites to the soma
g_l	S	0.1	Leak conductance
V^R	mV	0	Resting membrane potential
C_m	F	1	Membrane capacitance
P_0	–	$20/\phi max$	Hidden layer error signal scaling factor
P_1	–	$20/\phi^2 max$	Output layer error signal scaling factor

Table 1: List of hyper-parameters and parameters values used in the network simulations of the models. Adopted from Guergiuev et al., 2016.

The somatic compartment generates spikes of a vector $\phi^0(t) = [\phi_1^0(t), ..., \phi_m^0(t)]$ using the Poisson process similar to what was used in the model of **(Guergiuev et al., 2016)** and a non-linear sigmoid $\sigma(.)$ is applied to the i^{th} hidden layer neurons of the somatic voltage to generate this rate of fire and the output layer is having 10 neurons fashioned into two-compartments neurons similar to the deep learning model using segregated dendrites **(Guergiuev et al., 2016).** Both the somatic and dendrites voltages $V^1(t) = [V_1^1(t), V_2^1(t), ..., V_n^1(t)]$ and $V^{1b}(t) = [V_1^{1b}(t), V_2^{1b}(t), ..., V_n^{1b}(t)]$ respectively, are updated in the same manner with the hidden layer basal and soma as done in the model of **(Guergiuev et al., 2016).**

A similar teaching signal $I_i(t)$ is maintained on the output layer that tends to force the output layer towards the right target similar to the model of **(Guergiuev et al., 2016).** However,no evidence of such teaching signal that has been mirrored into the real brain but there are studies that prove an animal's desired behaviour output can be influenced by the internal goal representation **(Guergiuev et al., 2016).** The model hidden layer neuron computation such as the plateau and transmission are similar to the model of **(Guergiuev et al., 2016).**

The proposed network can take advantage of the multi-layer architecture to enhance the learning performance, which is a critical characteristic of deep learning **(Bengio et al., 2015, LeCun et al., 2015)** and it can also exhibit the characteristics of a multi-model as demonstrated in this study using the MNIST dataset.

2.1 Model Architecture

For artificial intelligence to properly evolved to maturity, AI needs to understand its world and how to interpret and reason about multi-modal messages **(Ngiam et al., 2011).** The multi-model machine learning aims to build a model that has the capacity and ability to process data and information from multiple modalities **(Ngiam et al., 2011).**

The multi-model is an emerging field that has real-world applications in healthcare, robotics, autonomous driving, video and audio processing, etc. However, this comes with some challenges such as: how the data changes across the modalities, known as the Translation problem; How

the elements from different modalities are related, known as the Alignment problem; How the information coalesced from different modalities to perform any prediction task, known as fusion problem; and lastly how the knowledge transfer occurs between modalities, known as a co-learning problem. The architecture in this study is a multi-modal model built on a multi-layer model of **(Guergiuev et al., 2016)** which takes halves (left and right as shown in. figure 1) and pass them on to each sub-network. Every sub-network is expected to contribute its learning(Θ) toward the network prediction at the end of each cycle.

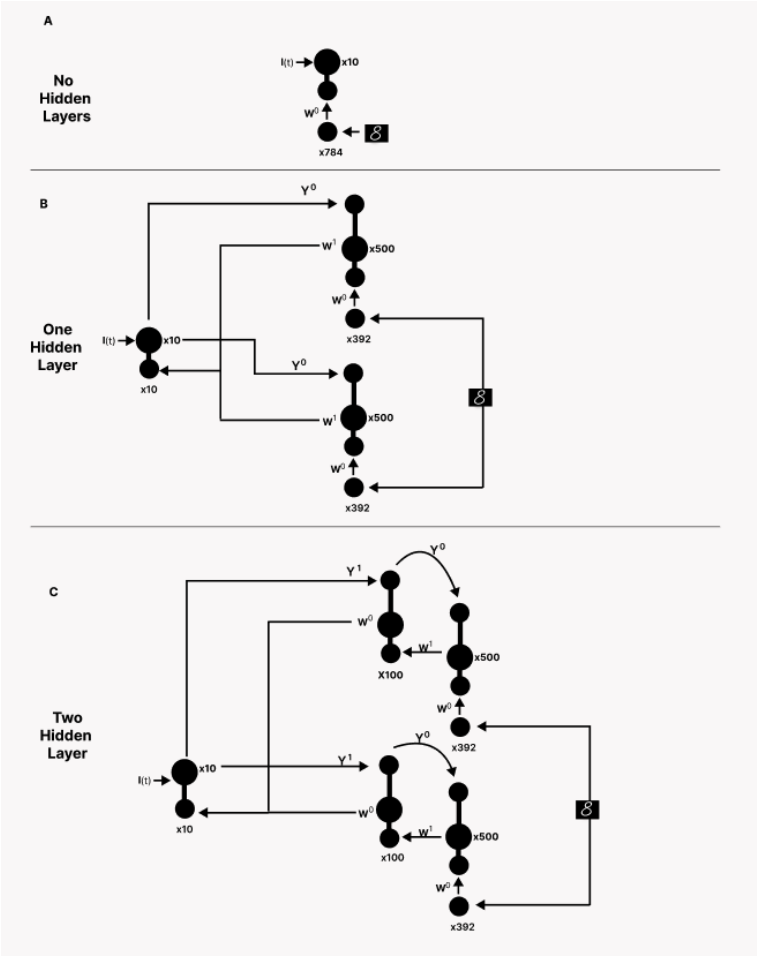

Figure 1: Illustration of the network architecture of three models used in the simulations. **(A)** Shallow network with no hidden layer i.e only input layer and output layer, **(B)** A network with one hidden layer, and **(C)** A network with two hidden layers. The network is designed to act as independent sub-networks that receives their feedback from the output layer through independent synaptic path with random weight Y^0 and Y^1. The hidden layers somata output are concatenated i.e weight W^1 of each sub-network are project to the output layer basal compartments while feedback from the output layer somata are projected to the hidden layers apical compartments through weight Y^1 for two hidden layers and Y^0 for one hidden layer.

2.2 Plateau Potentials and Weight Updates

The plateau computation is similar to the model of **(Guergiuev et al., 2016)** which uses the apical voltage to predict the plateau potentials by applying a non-linear function in the mean of the recent 20 - 30 ms period of the hidden layer neuron. The plateau potentials are numeric

values that are used in credit assignment and these are transported to the basal dendrites once they occur. However, they are stored temporarily for 0 - 60 ms and used in calculating the synaptic weight as done in the model of **(Guergiuev et al., 2016).**

The network dendrites uses gradient descent approximation to update the weights in the output layer:

$$W^1 \rightarrow W^1 - \eta^1 P^1 \frac{\delta L^1}{\delta W^1} \tag{4a}$$

$$b^1 \rightarrow b^1 - \eta^1 P^1 \frac{\delta L^1}{\delta b^1} \tag{4b}$$

where the η is the learning rate, P^1 is the rate of fire function scaling factor defined as:

$$L^0 = ||\phi^{0*} - \phi_{max}\sigma(V^{0^-f})||_2^2 \tag{5}$$

where $\phi^{0*} = ||\phi_1^{0*}, \phi_2^{0*}, ..., \phi_m^{0*}||$ such that ϕ^{0*} is equal to equation (9) where $\alpha^f = [\alpha_1^f, \alpha_2^f, ..., \alpha_m^f]$ and $\alpha^t = [\alpha_1^t, \alpha_2^t, ..., \alpha_m^t]$ are the plateau potentials of the forward and target phase. The basal weights are updated using a gradient descent approximation:

$$W^0 \rightarrow W^0 - \eta^0 P^0 \frac{\delta L^0}{\delta W^0} \tag{6a}$$

$$b^0 \rightarrow b^0 - \eta^0 P^0 \frac{\delta L^0}{\delta b^0} \tag{6b}$$

with the assumption of all the activities on the target phase to be fixed relative to the voltage firing rate.

3 Learning With Feedback Driven Plateau Potentials

3.1 Coordinating local learning across layers

In this simulation, a loss function was defined to solve the credit assignment on the hidden and output layer without any weight being transported. However, there's a consideration of how the hidden layer neurons impact the output layer of the network during the update **(Guergiuev et al., 2016).** This is achieved by the target firing rate of the output neurons ϕ^{1*}= [ϕ_1^{1*}, ..., ϕ_n^{1*}] as the mean of the firing rate during the target phase (Guergiuev et al., 2016).

$$\phi^{1*} = \frac{1}{\Delta t_2} \int_{t_1+\Delta t_s}^{t_2} \phi^{1*}(t)dt \tag{7}$$

A loss function at the output layer computes the average of the difference between the output of the forward phase and target phase as described in 3.3. The loss function is evaluated to zero if the mean firing rate of the output layer forward phase is equal to the target i.e the target phase mean firing rate; the closer its to zero the mean is, the closer the images match the output pattern imposed by the teaching signal.

The plateau potential integrates the top-down feedback which guarantees that the hidden loss function is zero if the output layer loss function is zero thus credit assignment is assured by updating the synaptic weight of the hidden layer to reduce the local loss function on the hidden layer a stochastic gradient descent is used after every round of the target and forward phase **(Guergiuev et al., 2016).**

Demonstrating how the network utilizes the multi-layer architecture is done whereby the loss function definitions are defined in a manner that the hidden layer L^0 reduces by updating W^0, which will equally reduce the L^1 resulting in what the sub-network hidden layer learns implies that the output layer has learned the same similar to the model of **(Guergiuev et al., 2016).**

3.2 Calculating credit signal

The network is trained in two phases: The forward phase is a phase where the input image is presented to the network without any teaching signal at the output layer ($I(t)_i = 0, \forall_i$), which occurs between t_0 - t_1 follows by plateau potential calculation in the neurons of the hidden layer $\alpha^f = [\alpha_1^f, ..., \alpha_m^f]$. At the end of this phase at t_1 the second phase begins known as the target phase, that takes the input image with a teaching signal on the output layer which forces the correct output layer neuron to have a max firing rate while others neurons are silenced this occurs until t_2 when plateau potentials $\alpha^t = [\alpha_1^t, ..., \alpha_m^t]$ in the hidden layers are calculated again, similar to the model in **(Guergiuev et al., 2016)** as shown in equation (8b). The plateau potential of the hidden layer target and forward phase are generated using the same equation in deep learning segregated dendrite **(Guergiuev et al., 2016).**

$$\alpha_i^f = \sigma\left(\frac{1}{\Delta t_1}\int_{t_1-\Delta t_1}^{t_1} V_i^{o_a}(t)dt\right) \tag{8a}$$

$$\alpha_i^t = \sigma\left(\frac{1}{\Delta t_2}\int_{t_2-\Delta t_2}^{t_2} V_i^{o_a}(t)dt\right) \tag{8b}$$

where t_1 and t_2 are the end times of forward and target phases respectively, and $\Delta t_1 = t_1 - (t_0 + \Delta t_s)$, $\Delta t_2 = t_2 - (t_1 + \Delta t_s)$, and the settling constant of the voltage $\Delta t_s = 30ms$ that allows the network to settled before the plateaus are integrated, this is similar to the model of **(Guergiuev et al., 2016).** The network synapses are updated using the plateau potentials values α^f and α^t to influence the credit assignment on the network.

However, this is achieved after training the weight matrices W^0 and W^1 on a stochastic gradient descent on both the hidden and output layer local functions respectively while preserving the state of the weights of during the process similar to the model of **(Guergiuev et al., 2016).**

The target firing rate of the hidden layer neuron $\phi^{0*} = [\phi_i^{0*}, ..., \phi_n^{0*}]$ is defined by the equation (9) below:

$$\phi^{1*} = \phi^{0^{--^f}} + (\alpha_i^t - \alpha_i^f) \tag{9}$$

The cost function of the hidden layer is the difference between the target firing mean of the forward (α^f) and target (α^t) phase:

$$L^0 = ||\phi^{1*} - \phi^{-f}||_2^2 \tag{10}$$

$$\Delta W^1 = -\eta_1 \frac{\delta L^1}{\Delta W^1} \tag{11a}$$

$$\Delta W^0 = -\eta_0 \frac{\delta L^0}{\Delta W^0} \tag{11b}$$

where η_i is the learning rate and ΔW^i is the weight matrix

The synaptic weight on the hidden layer using the stochastic gradient descent at the end of every target phase is updated using equation (11b) and (11a).

The purpose of adopting this type of learning is to force the output layer activity with the teaching signal to be the same as the layer with the teaching signal thus giving the network dynamism and the ability to predict appropriate output weight matrices using stochastic gradient descent on the local function which are used in training the hidden and output layer respectively.

Weight is updated at the end of every target phase similar to the model of **(Guergiuev et al., 2016).** as shown in figure 2.

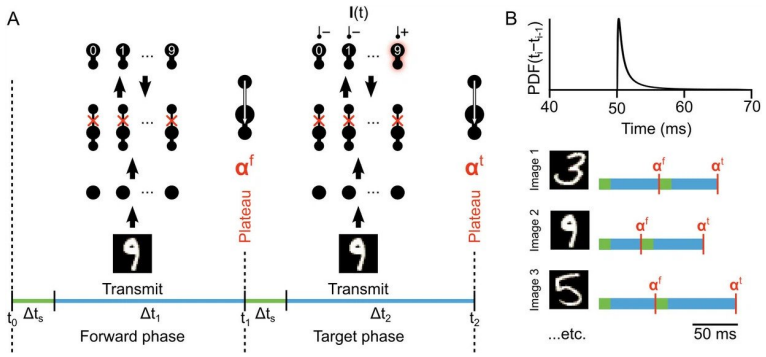

Figure 2: Adopted from Guergiuev et al., 2016 model.**(A)** Illustrate the network sequence learning phases i.e forward phase where $I_i(t) = 0$ that last upto t_1 and target phase that last upto t_2 during this phase $I_i(t)$ (equation (12)) is passed to force the neuron i to max-fire or silent based on the image category for each of the training sample. This is illustrated with '9' been the image that is showned to the network output layer which forces other neurons to be silent except '9'.The plateau potential at the end of forward and target phase is calculated using (8a) and (8b) respectively.**(B)** Illustrates the stochastic length of the sampled phase which is randomly drawn from a minimum of $50ms$ shifted inverse Gaussian distribution.

$$I_i(t) = gE_i(t)(E_E - V_i^1(t)) + gI_i(t)(E_I - V_i^I(t)) \tag{12}$$

3.3 Coordinate local learning to mimic back-propagation of error

Back-propagation of error is the primary algorithm that is used in supervised deep neural networks **(Guergiuev et al., 2016).** Considering some of the previous studies **(Linnainmaa, 1976, T. P. Lillicrap et al., 2014)** that demonstrated learning with random weight can be matched with synaptic weight updates of a back-propagation algorithm which allows the model to approximate credit assignment similar to back-propagation. However, in this study, the hidden layer activity was divided into two halves (sub-networks) to be trained as independent models both during the target and forward phase and the two models are concatenated and then passed to the output layer for prediction.During the training, the input dataset was split into two halves (left and right) and then projected into the hidden layers to train with each having the same amount of hidden neurons and to independently update it weights W, Y, b and c which is then concatenated and used as a single input to the output layer.

3.4 Deep Learning with Dendrites

As demonstrated in the findings of **(Guergiuev et al., 2016)** on credit assignment in a manner that the output layer error is assigned to hidden layer neuron which makes local weight update causing the network exhibit deep learning properties. As examined by the models of **(Guergiuev et al., 2016,Li et al., 2016)** to see if the network is having the deep learning characteristics whereby including the extra hidden layers on the network could increase the model classification accuracy.

In the training network with 60,000 images for 60 epochs on each of the networks. The first network had an average error of 84.10% while the network with one hidden layers had an average error of 6.10% and the third network with two hidden layers achieved an average error of 5.4% on the 10,000 test data at the end of the 60^{th} epoch as shown in table 3.

However, some of the characteristics during the learning in the model of **(Guergiuev et al., 2016)** were also observed in these networks.

During the experiment as it were in the model of **(Guergiuev et al., 2016)** where the data was shuffled but in this study the shuffling of the data tends to impair the learning ability of the model.

However, it was observed that when extra layers are added to the network there was a significant improvement in the model accuracy which gives it the ability to form a deep neural network by leveraging multi-layer architecture in the learning.

3.5 Condition for feedback weights

The model demonstrated in this study adopted the same approach as the model of **(Guergiuev et al., 2016).** However, in this study, the hidden layers were maintained for each sub-network (left and right) of the input data and each sub-network (left and right) is to update its respective weight, bias and target both during the target phase and forward phase.

Validating the previous finding **(T. Lillicrap et al., 2017)** the models were trained with sparse feedback weight and symmetric weight similar to the model of **(Guergiuev et al., 2016).** The learning with the sparse weight had a mean test error on the 10,000 samples of 3.2% at the end of the 60^{th} epoch while the symmetric weight had a mean error rate of 6.1% at the end of the 60^{th} epoch. This demonstrated that there was a substantial credit assignment in the network.

3.6 Network Hyper-parameters

This study network simulation used the learning rates of 0.19 for the network without any hidden layer; 0.21 for the network with one hidden layer for both the output and hidden layer while the learning rates for network with two hidden layer was 0.23 and 0.12 for the hidden and output layer respectively. The network with the single hidden layer had 500 neurons while the network with two hidden layers layers had 500 and 300 neurons on the first and second layer respectively.

3.7 Training paradigm

The deep neural network simulation in this study were conducted using the MNIST data set of 28x28 pixel images. At the beginning of the training 10,000 test sample are shown to the network in sequence for classification. Each of the image had a Poisson spike of 392 input neurons to represent the pixel equivalent of the image with it firing rate proportional to the pixels intensity between $[0-\phi_{max}]$ where the ϕ_{max} is the max spiking rate, in each of the test sample, the network undergoes a forward phase after receiving the spiking activity from the 392 neurons of the hidden layer then at the end of this phase the output layer gives the network the classification of the input image with the greatest spike rate and the network will classify and compare with the given output.

The training of the model was done in an online manner where all the 60,000 images were presented to the network at the start of the training epoch. The image is then presented in a sequence to the network undergoing forward phase with a plateau potential at the end and, a target phase with another plateau potential at the end. The network updates the feed-forward weights at the end of the target phase.

The 60,000 images are shown to each of the sub-network for each epoch and 10,000 samples are tested on the network classification performance at the end of each epoch and the classification error is given based on the images that were classified accurately. This was repeated for 60 epochs in each of the experiments.

However, it's worth noting that both the test and training dataset were never shuffled at any point in time during the testing and training at it were in the model of **(Guergiuev et al., 2016).** Shuffling the data set was impairing the learning ability of the model.

3.8 Network Simulation Results and Plots Samples

The results comparison and plots of the network simulations are shown below

No of layers	Multi-model	Uni Model
No Hidden Layer	78%	91.23%
One Hidden Layer	6.10%	91.23%
Two Hidden Layers	6.93%	91.23%

Table 2: This shows the final error of the multi-model after 60 epochs of training and the existing uni-model. The uni-model data was ingested as a full image but keeping half of the image during training corrupted by changing it values to 0's but passed the full image during test.

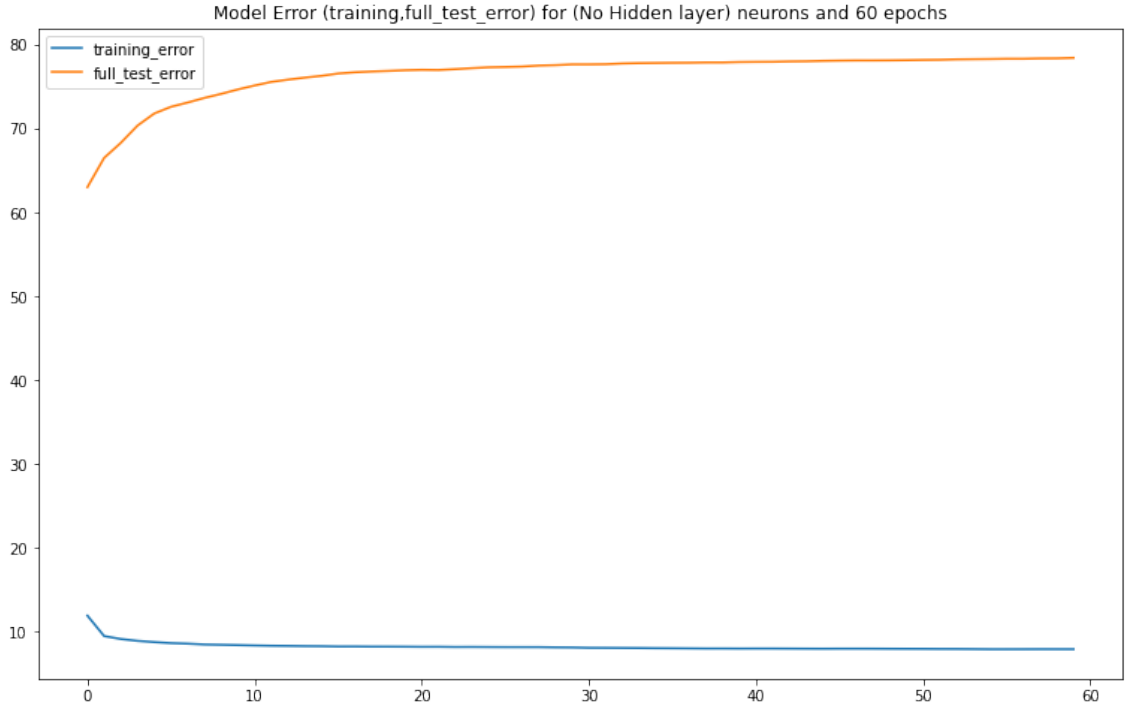

Figure 3: The training and test errors plot of the simulated model with no hidden layer showing training error less while full test error is high after 60 epochs.

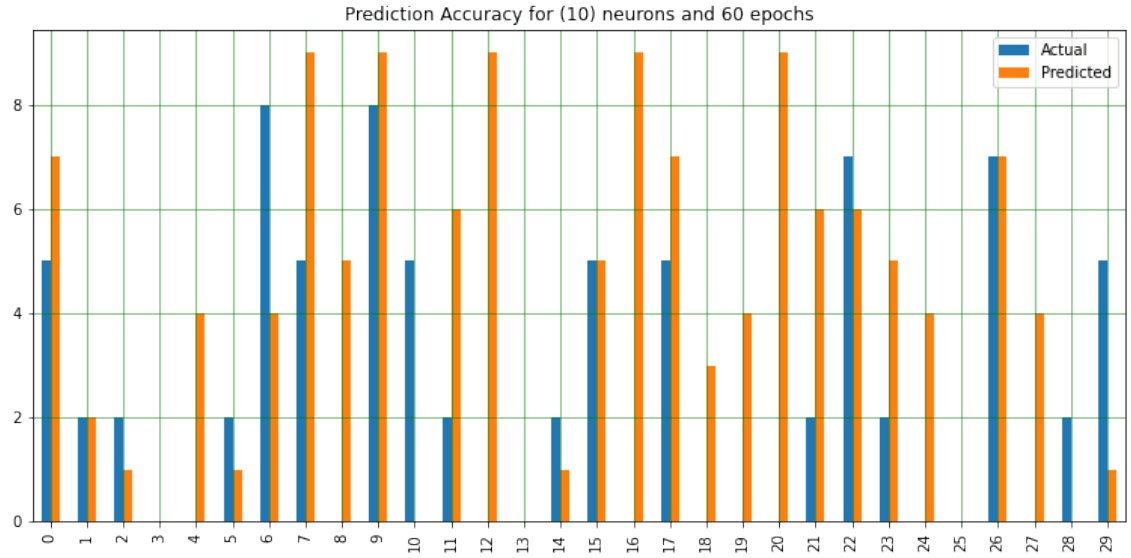

Figure 4: This shows the first 30 predictions of the model after 60 epochs with no hidden layer of the simulated networks

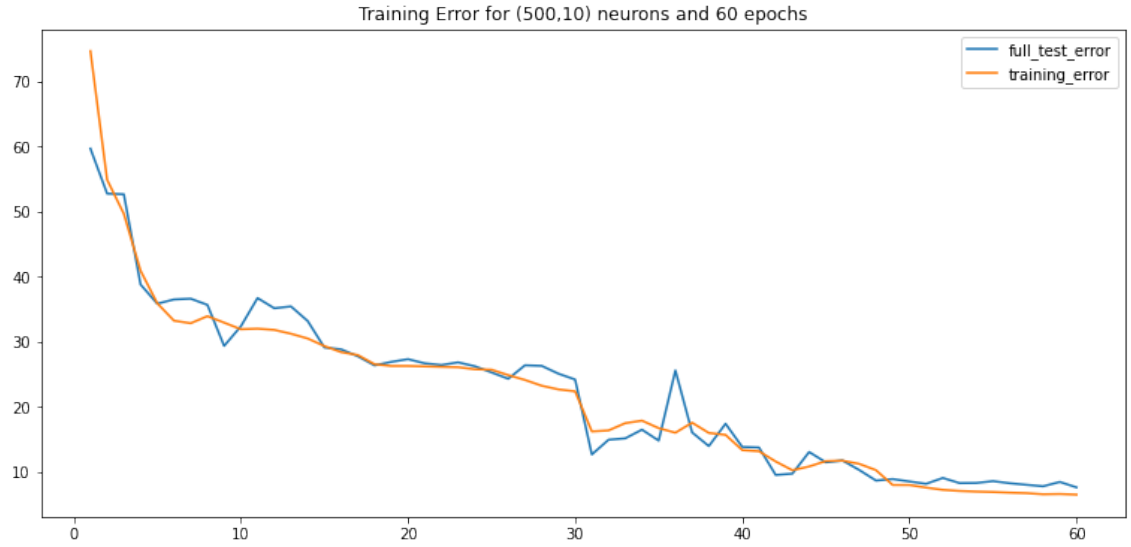

Figure 5: The training errors and test errors plot of the three models that were simulated after 60 epochs. The error appears to be less in training for all models and abit more in the final tests.

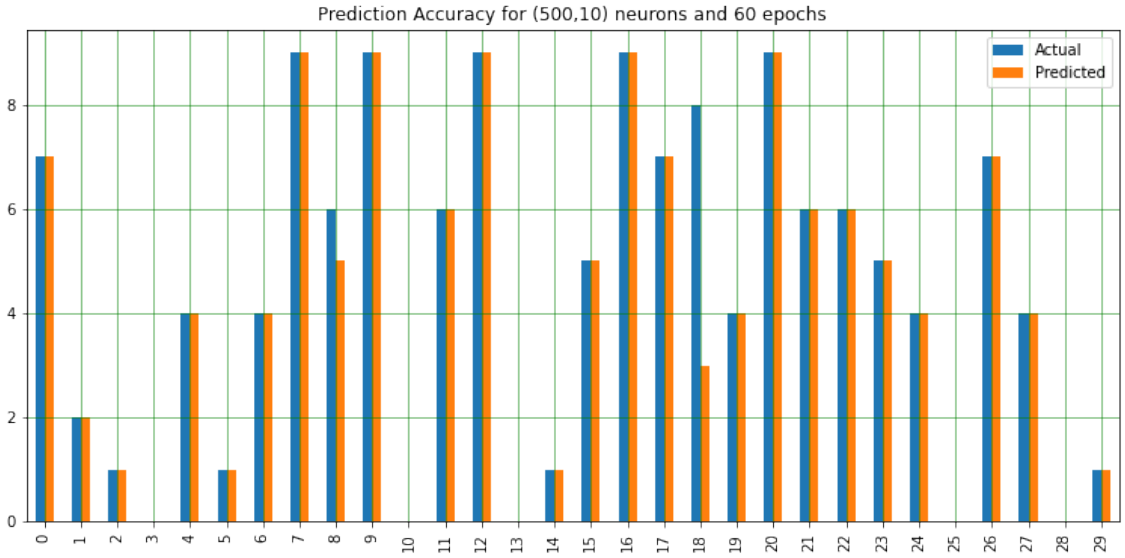

Figure 6: The training errors and test errors plot of the three models that were simulated after 60 epochs. The error appears to be less in training for all models and abit more in the final tests.

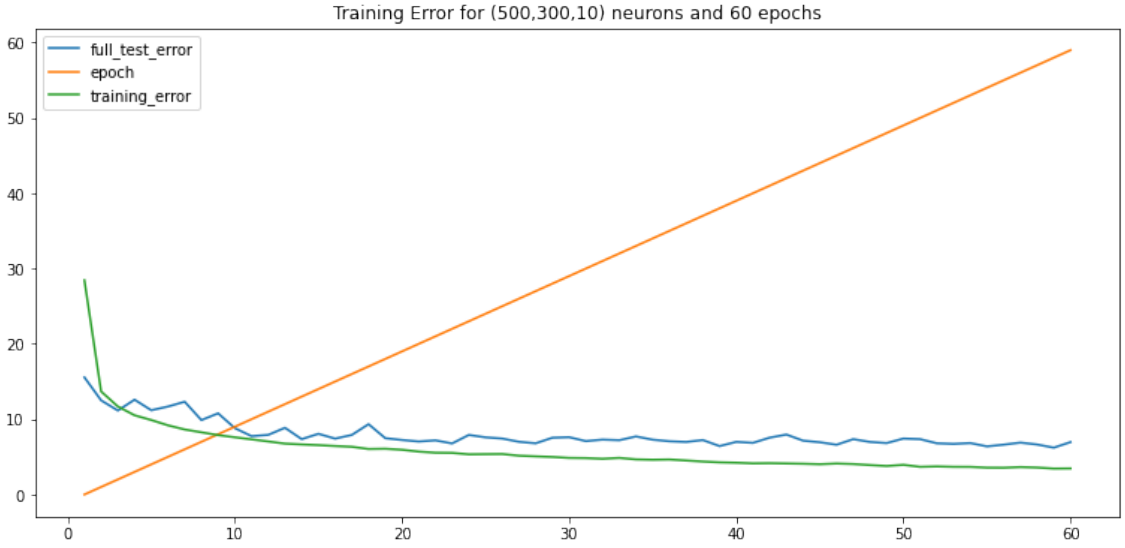

Figure 7: The training errors and test errors plot of the three models that were simulated after 60 epochs. The error appears to be less in training for all models and a bit more in the final tests.

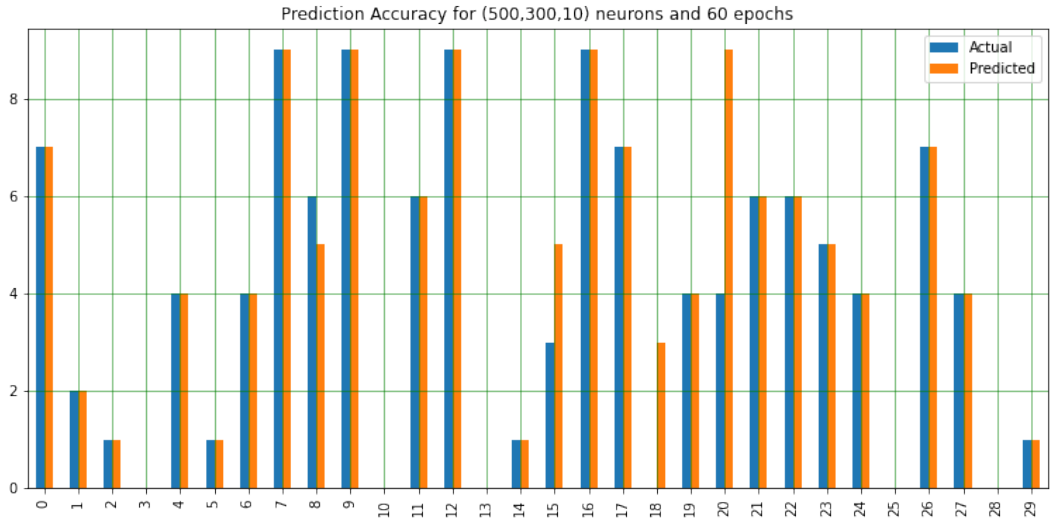

Figure 8: The first 30 predictions of the simulated network without any hidden layer. This shows how many of the images were missed-classified by the network after 60 epochs of training.

No of layers	**Training Errors**	**Test Errors**
No Hidden Layer	7.8%	68%
One Hidden Layer	3.2%	6.1%
Two Hidden Layers	3.1%	5.4%

Table 3: The comparison of all the simulated models test and training errors showing the training error less while the test error high after 60 epochs.

3.9 Network Simulation Information

During the training, the length of the forward and target training phase is determined by adding their minimum length to the additional length term randomly selected from Wald distribution with an average of $2ms$ and a scaling factor of 1. The mean forward and target phase voltage were calculated after the settling period of $\Delta ts = 30ms$ from the start of the phase. In a simulation using a randomly selected plateau potential time , the time at which the plateau potential of each neuron occurred is from a folded normal distribution ($\mu = 0$, $\sigma^2 = 3$) randomly selected ($max = 5$), a plateau potential between 0 and $5ms$ occurred before the start of the next phase.

The network was updated bottom-to-top starting from the hidden layer to the output layer for each of the time-step of $dt = 1ms$. For each of the layers in the network the dendritic potential is updated then the somatic potentials and end with the spike activity similar to the model of **(Guergiuev et al., 2016).**

The data used for network training was taken from the National Institute of Standards and Technology (MNIST) database, a modified version of the original National Institute of Standards and Technology database **(Lecun et al., 1998).** While the code for this simulation was written in Python programming language (Python) 3.9.X, NumPy 1.20.3, and SciPy 1.7.1 libraries. The code for this simulation is available at https://github.com/Nazehs/multi-model-segragated-dendrites-model .The MNIST database is located at http://yann.lecun.com/exdb/mnist/

4 Conclusion

This study presents a multi-model multi-layer feed-forward deep neural network architecture that used segregated dendrites with local credit assignment and a two-phase learning scheme.And this demonstrates deep learning ability using local credit assignments. And this demonstrates how dendrite segregation and synaptic feedback play a significant role in how the model learning performance can be impacted. Additionally, this study also, demonstrated the possibility of how machine learning and neuromorphic learning occur which can be applied in real-world applications.

References

Alber, M., Bello, I., Zoph, B., Kindermans, P., Ramachandran, P., & Le, Q. V. (2018). Backprop evolution. *CoRR*, *abs/1808.02822*. http://arxiv.org/abs/1808.02822

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards ai.

Bengio, Y., Lee, D., Bornschein, J., & Lin, Z. (2015). Towards biologically plausible deep learning. *CoRR*, *abs/1502.04156*. http://arxiv.org/abs/1502.04156

Guergiuev, J., Lillicrap, T. P., & Richards, B. A. (2016). Towards deep learning with segregated dendrites. https://doi.org/10.48550/ARXIV.1610.00161

Guerguiev Jordan. (2016, May 11). *Segregated dendrite deep learning* (Version 23f2c66). https://github.com/jordan-g/Segregated-Dendrite-Deep-Learning

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

Li, Y., Li, H., Xu, Y., Wang, J., & Zhang, Y. (2016). International conference on intelligent data engineering and automated learning.

Lillicrap, T., Cownden, D., Tweed, D., & Akerman, C. (2017). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, *7*(13276), 1–10.

Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2014). Random feedback weights support learning in deep neural networks. https://doi.org/10.48550/ARXIV.1411.0247

Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, *16*, 146–160.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. Y. (2011). Multimodal deep learning. *ICML*, 689–696. https://icml.cc/2011/papers/399_icmlpaper.pdf

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. In R. F. Drenick & F. Kozin (Eds.), *System modeling and optimization* (pp. 762–770). Springer Berlin Heidelberg.

Yang, S., Gao, T., Wang, J., Deng, B., Lansdell, B., & Linares-Barranco, B. (2021). Efficient spike-driven learning with dendritic event-based processing. *Frontiers in Neuroscience*, *15*. https://doi.org/10.3389/fnins.2021.601109