Equilibrium Propagation for MNIST Classification

Indian Institute of Technology, Bombay

Shambhavi Shanker
Electrical Engineering
21D070066

Swadhin Dash
Electrical Engineering
210020142

Parth Bansal
Electrical Engineering
21D170028

I. Motivation

Spike-based neuromorphic computational systems have, in recent years, demonstrated outstanding energy efficiency on inference tasks. Implementing the training of deep neural networks in such systems remains, however, a considerable challenge, as backpropagation does not apply directly to spiking networks. Equilibrium spiking when implemented in silicon neuromorphic technology can reduce the energy consumption of inference and training respectively by three orders and two orders of magnitude compared to GPUs expanding the horizon of shifting learning to analog computers. Ongoing research efforts therefore investigate how the error backpropagation algorithm can be mathematically modified to make it appropriate for our purpose.

II. Objective

Backpropagation is a widely used algorithm for training artificial neural networks, but it has certain limitations when applied to biological models. Being a computational approach, it may function efficiently in digital computers, but since it heavily relies on calculating precise gradients, it can be challenging for biological systems that lack such precision. The brain operates in an analog fashion which makes it difficult for Backpropagation's weight updates to align with the learning mechanisms observed in the brain. Backpropagation also requires access to the network's entire architecture during training, which is inconsistent with the distributed and local learning processes in the human brain.

Hence, to address the above limitations we explore Equilibrium propagation which frames learning as an energy-minimization problem. Here, each neuron's activity is considered a state variable, and the network seeks an equilibrium state where the energy function is minimized. Learning occurs by minimizing the difference between the current state and the target state through an iterative process. It neither has error gradients nor requires storing activations in external memories, and synapses can be directly updated through neuronal events.

III. Basic Overview

In this paper, we present the concrete realization of the equilibrium propagation concept within the context of solving a specific problem - the classification of handwritten digits in the MNIST dataset

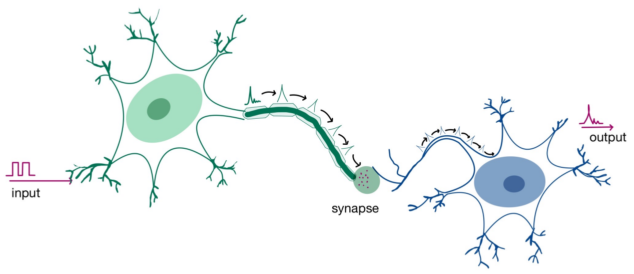

Fig. 1. Synaptic Transmission across two neurons

A. Neural Network Structure

Here, we have a digital neural network, like a simplified brain, with multiple artificial neurons arranged in layers, each of which neuron can either "spike" (send a signal) or stay silent, just like real neurons in our brain. The output layer is just 10 neurons each representing a number from 1 to 10 and the neuron spiking the most "vigorously" represents the prediction of the network for that particular image input.

B. Data Input

We feed this neural network with the MNIST dataset which consists of several images of handwritten numbers. These images are represented as patterns of spikes in the input layer of our network and the spiking pattern of each neuron in the input layer represents a pixel in the image. Information from the input layer travels down the network as a current that generates a cascade of spikes in the middle-layer neurons.

C. Energy Minimization

The goal is to teach the network to recognize the number in each image. To do so, the network tries to find the most efficient way to represent the image by changing the spiking patterns in its layers. This is done by adjusting the "weights" between different neurons. The more the "weight", the more the current input from the previous layer neuron and the more the chances of spiking in the former layer. It's like a puzzle where the network adjusts itself to match the input image as closely as possible while minimizing a type of "energy." The energy in this context is a theoretical concept that represents how far off the neural network's current state is from the desired state.

D. *Iterative Process*

The neural network makes an initial guess, and if that guess is incorrect, it continues to fine-tune the ”weights” through an iterative process based on a certain ”penalty” function. This involves striving to minimize the gap between its current patterns of neural activity and the desired patterns that would correspond to the right answer.

E. *Learning and Feedback*

During these iterations, the network learns which spikes need to be adjusted and accordingly adjusts the weights. This learning process happens through feedback loops.

F. *Classification*

As the network is trained on many examples, it learns to recognize new handwritten numbers, by adjusting its spiking patterns to match the input as closely as possible to a known image of a number and the final state of the network is the recognized digit.

IV. Hypothesis

A. *The Energy Model*

Neuronal configuration serves as an explanation for sensory data. In an energy-based model, the network units adjust towards lower energy states that align with sensory input and the current understanding of the world within the model's parameters.

The set of units of the network is denoted by u, and $\theta = (W, b)$ denotes the set of free parameters to be learned, which includes the synaptic weights W_{ij} and the neuron biases b_i. The units are continuous-valued and correspond to averaged voltage potential across time, spikes, and possibly neurons in the same minicolumn. Finally, ρ is a nonlinear activation function such that $\rho(u_i)$ represents the averaged firing rate of unit i.

The above continuously valued network of neurons is called a Hopfield model and its energy function is defined as:

$$E(u) := \frac{1}{2}\sum_i {u_i}^2 - \frac{1}{2}\sum_{i \neq j} W_{ij}\rho(u_i)\rho(u_j) - \sum_i b_i \rho(u_i) \quad (1)$$

The set of all units in the network is $u = \{x, h, y\}$ where x represents the input layer neurons which are always

”clamped”, y represents the output layer which are partially ”clamped” and h represents the hidden layer neurons. We define a quadratic cost function **C**

$$C := \frac{1}{2}\|y - \hat{y}\|^2 \quad (2)$$

which provides a form of ”external force” to the output layer neurons and drives them towards their targets i.e $\hat{y}$.

A novelty force total energy function F is defined as

$$F := E + \beta C \quad (3)$$

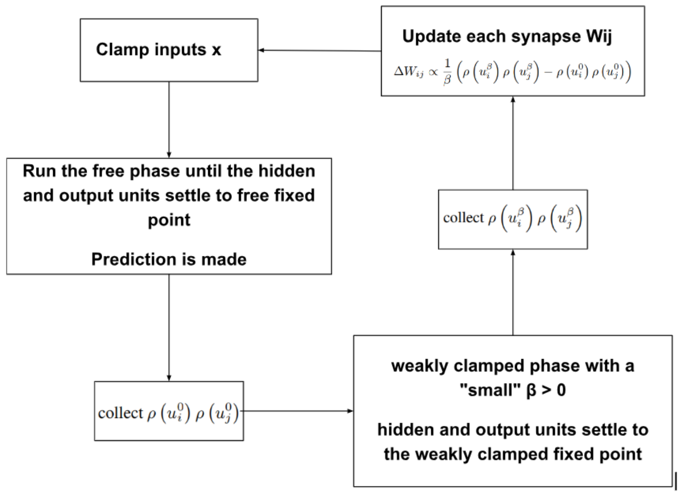
Clamp inputs x
Update each synapse Wij
$\Delta W_{ij} \propto \frac{1}{\beta}\left(\rho\left(u_i^\beta\right)\rho\left(u_j^\beta\right) - \rho\left(u_i^0\right)\rho\left(u_j^0\right)\right)$
Run the free phase until the hidden and output units settle to free fixed point
Prediction is made
collect $\rho\left(u_i^\beta\right)\rho\left(u_j^\beta\right)$
collect $\rho\left(u_i^0\right)\rho\left(u_j^0\right)$
weakly clamped phase with a "small" β > 0
hidden and output units settle to the weakly clamped fixed point

Fig. 2. Learning Flow Diagram

where $\beta \geq 0$ represents the ”weakly clamped phase” of the output layer and $\beta = 0$ represents the free phase where our network makes a prediction without any ”external force”.

B. *The Dynamics Model*

We denote $s = \{h, y\}$ which are the un-clamped states and we assume its dynamics follow the equation

$$\frac{ds}{dt} = -\frac{\partial F}{\partial s} \quad (4)$$

$$\frac{dF}{dt} = \frac{\partial F}{\partial s}\frac{ds}{dt} = -\|\frac{\partial F}{\partial s}\|^2 \leq 0 \quad (5)$$

This guides the fact that the total energy of the system keeps decreasing until it reaches a fixed point governed by the equation

$$\frac{dF}{dt} = 0 \iff \frac{ds}{dt} = 0 \iff \frac{\partial F}{\partial s} = 0 \quad (6)$$

The net force can be viewed as a sum of two forces acting on any neuron:

$$\frac{ds_i}{dt} = -\frac{\partial E}{\partial s_i} - \beta\frac{\partial C}{\partial s_i} \quad (7)$$

where the first term is the ”internal” force induced by the model to drive the system to a state of energy minima

$$-\frac{\partial E}{\partial s_i} = \rho'(s_i)\left(\sum_{j \neq i} W_{ij}\rho(u_j) + b_i\right) - s_i \quad (8)$$

and the second term is the ”external force” driving the output layer towards its target

$$-\beta\frac{\partial C}{\partial h_i} = 0 \quad \text{and} \quad -\beta\frac{\partial C}{\partial y_i} = \beta(y_i - \hat{y}_i) \quad (9)$$

So our training can operate in two phases, a ”free” phase where there is no external force and the network settles to a fixed point u^0 which is its prediction of the current input, and another where $\beta \geq 0$ which represents the nudging of y or the

”weakly clamped” phase and the network settles to the state u^{β}. The hidden layers don’t feel the external force directly but it propagates as time progresses because the ”nudging” of the output layer exerts a force on the h units eventually in the ”free” phase.

The weakly clamped phase implements back-propagation of error through synaptic weight updates given by

$$\Delta W_{ij} \propto \lim_{\beta \to 0} \frac{1}{\beta} \left(\rho(u_i^{\beta})\rho(u_j^{\beta}) - \rho(u_i^{0})\rho(u_j^{0}) \right) \tag{10}$$

V. Methods

In this section, we describe the implementation of the above energy-based model and train it on the classification of hand-written digits. The MNIST dataset of handwritten digits consists of 60,000 training examples and 10,000 test examples. Each example x in the dataset is a gray-scale image of 28x28 pixels and comes with a label $y \in \{0, 1, ..., 9\}$. We use the same notation $\hat{y}$ for the one-hot encoding of the target, which is a 10-dimensional bit vector.

We train a neural network with no skip-layer or lateral connections with only symmetric weights. The flow of the training goes as per Fig 12. The input layer is clamped ❶ with the handwritten digit image. The free phase is run until the network settles to a fixed point ❷ and the $\rho(u_i^0)\rho(u_j^0)$ values are collected for each synapse. The prediction ❸ y_0 is the output neuron whose activation is the highest among the 10. We have to minimize the square of the prediction error given by

$$C = \frac{1}{2}\|y_0 - \hat{y}\|^2 \tag{11}$$

The weakly clamped phase ❹ is run with a very small $\beta > 0$ until the network settles to a fixed point and the $\rho(u_i^{\beta})\rho(u_j^{\beta})$ values are collected for each synapse. Each synapse weight is updated ❺ according to equation (10). This is a single step of gradient descent on the total energy F with a step size ϵ. In our experiments, we adopt the ReLU(Rectified Linear Unit) activation function $\rho(s_i) = 0 \vee s_i$, where $\vee$ denotes the minimum. With this choice of ρ, as $\rho_0'(s_i) = 0$ for $s_i < 0$, it follows from equations 8 and 9 that if $h_i < 0$, then $\frac{\partial F}{\partial h_i} = -h_i < 0$ i.e. the force prevents the values from going below zero.

A. Update Rule

$$s_i \leftarrow 0 \vee \left(s_i - \epsilon \frac{\partial F}{\partial s_i}(\theta, v, \beta, s) \right) \tag{12}$$

B. Choice of step size

Fixing other parameters, we vary ϵ and observe the classification accuracy. What significantly impacts the outcome is the total duration of relaxation, $\Delta t = n_{\text{iter}} \times \epsilon$, where n_{iter} represents the number of iterations. Essentially we want to find a value of ϵ that prevents unnecessary computations with diminished returns.

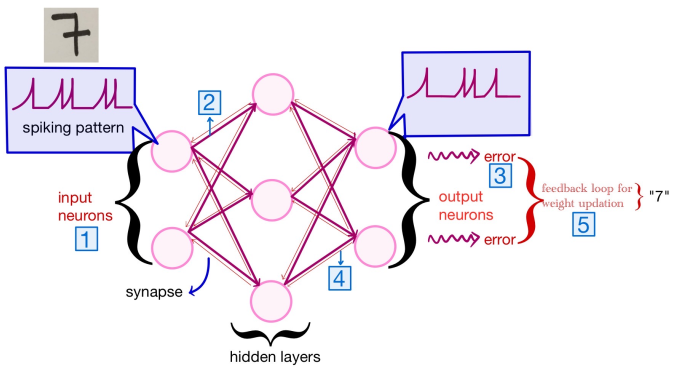

Fig. 3. Spiking neural networks (SNNs) architecture

C. Choice of β and Learning rates

We vary different values of β and draw observations on how small a β is good enough. We use different learning rates for different layers to maintain the normalized change in the weight of a synapse in the same order, irrespective of the fact that theory didn’t support it possibly because numerical optimization performs a stochastic gradient descent that is not along the steepest gradient, thus different learning rate for different layers is optimum.

D. Duration of the free phase relaxation

We tried to observe how different values of n_{iter1} affect the accuracy, keeping all other parameters fixed. The free phase time guides the propagation of input spiking information to the inner layers and guides the convergence.

E. Duration of the weakly clamped phase

We just need to start the error back-propagation and it’ll eventually travel. One good heuristic that we will test is that if the time that we should allot to this phase is N where N is the number of hidden and output layers since the time constant is the time taken by the signal to propagate significantly from one layer to the other. Hence the number of iterations will be $\frac{N}{\epsilon}$. Essentially we need to draw some parallel between the two phases of learning.

F. Experimental Setup and Network Architecture

The entire code has been written from scratch without the use of any State-Of-Art Neural Network model or optimization techniques. We have used only one hidden layer with 1024 neurons. It performed better than one hidden layer with 512 neurons.

Weights are initialized using Glorot-Bengio initialization to ensure stable training by controlling the variance of weights based on the layer’s input and output neurons. The weights of neural network layers are set by sampling from a distribution that considers the number of input and output neurons, aiming to prevent issues like vanishing or exploding gradients during training.

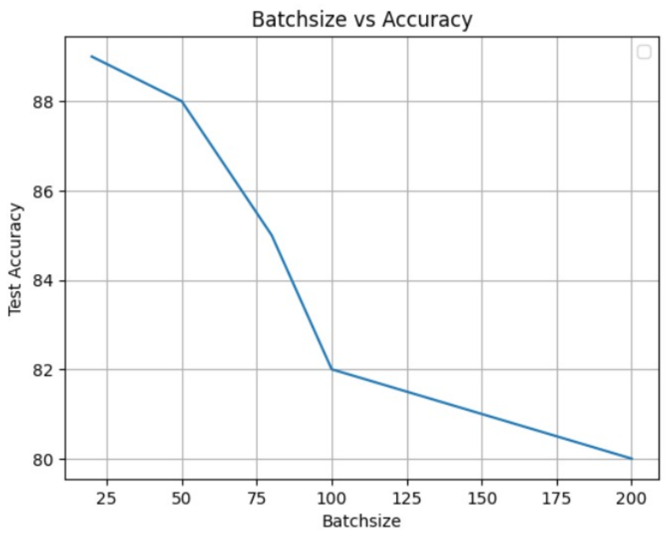

Fig. 4. Accuracy vs Batch Size

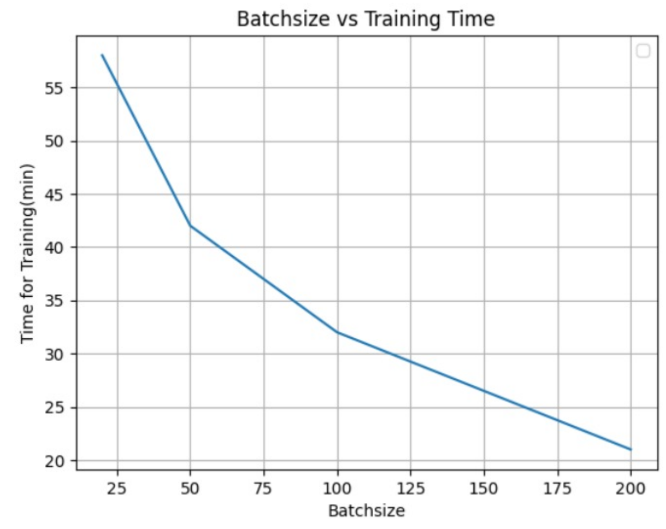

Fig. 5. Total Time Taken vs Batch Size

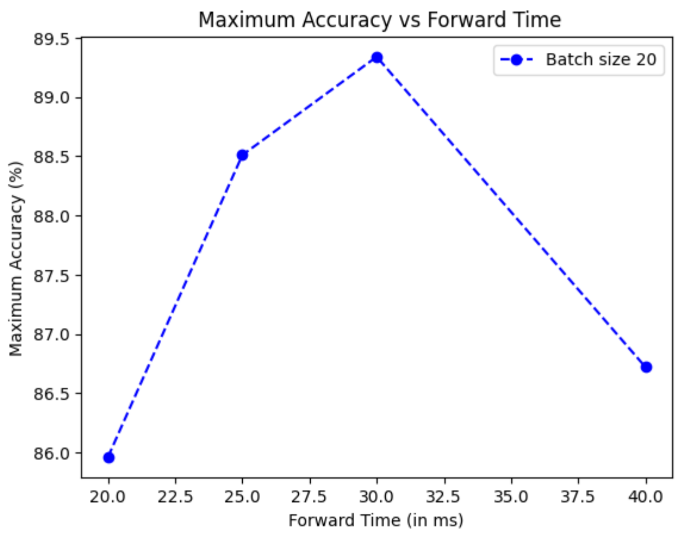

Fig. 6. Maximum Accuracy vs Forward Time

VI. Experimental Results and Conclusions

There are 5 important hyper-parameters namely Batch Size, Forward Pass Time, Backward Pass Time, Clamping Factor, and Learning Rates.

A. Batch Size

Firstly we play with different batch sizes. The batch update decreased the computation time by almost 100 times! Since the larger the batch size lesser the variance per update but more the iterations required to converge, there is scope for finding an optimum in computation time and convergence. We plot the performance of different batch sizes in terms of maximum accuracy achieved. Batch size of 20, as mentioned in the paper too, in practice performs the best(Fig4).

B. Forward Time

Next, we try to find the optimal forward pass time. Again we have the tradeoff - how much computation time is enough. We observed that if we run the forward pass for too long, there are chances that the particular image will ”overfit” the weights of the layers. Accordingly, we would need a long backward time and a larger number of epochs to ensure that each image doesn’t overfit too much. Our experiments suggest that assuming the synaptic time constant of the LIF neuron is 1 ms, the optimum forward time for best accuracy is around 30 ms i.e. 60 iterations $*$ 0.5 ms time-step(Fig6).

C. Backward Time

We have one heuristic idea that the time taken for the propagation of neuronal information is around the synaptic time constant. And only a single iteration of propagation is enough. The information will then continue to ”weakly” clamp the weights throughout future iterations without the need for a long backward time for each image. Thus we test different values of backward time around $N * tau = 2ms$ where N is the number of unclamped layers. We finally arrived at backward time = 3ms i.e. 6 iterations $*$ 0.5 ms time-step(Fig7). A larger backward time will ”overfit” the weights on the image and clamp or ”nudge” the weights towards the target more than expected, eventually overfitting the model on the image.

D. Time Step

We played around with the time-step around 0.5ms. The time-step essentially discretizes the numerical optimization and it acts more or less like a learning rate parameter. The smaller it gets, the more forward and backward time is required for the model to ”learn” a single image. And the larger it becomes, the more the discretization and the poorer the learning. So we had to find an optimum sandwiched between the trade-offs of computation time and accuracy. Our best performance is at dt = 0.5 ms(Fig8).

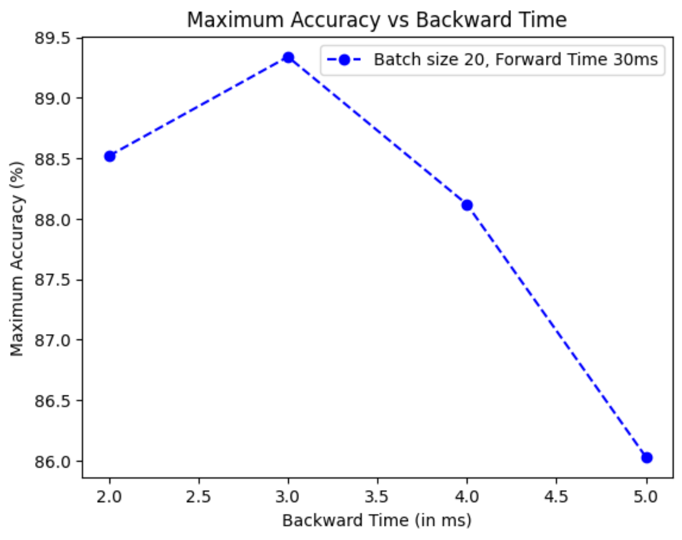

Fig. 7. Maximum Accuracy vs Backward Time

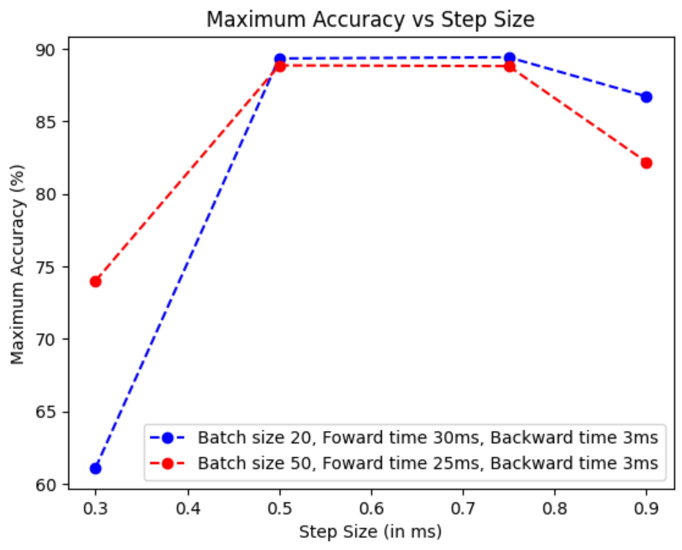

Fig. 8. Maximum Accuracy vs Time-Step at fixed forward and backward time

E. Clamping Factor

β is the clamping factor that drives the output layer towards the target in the weakly clamped phase. The larger the β the more the output layer is "driven" to the target and the more is the overfitting. According to the mathematics, Equation 10 claims that as β tends to zero, weight update can be written in a convergent manner since left and right derivatives converge. This is clearly beneficial for a digital computer and can be useless when it comes to an analog device where our observations conclude β tending to zero is optimal. Our experiments gave the maximum accuracy at $\beta = 0.5$ but again this is more of the device architecture that becomes a bottleneck, according to the above discussion and has no relation to the theory.

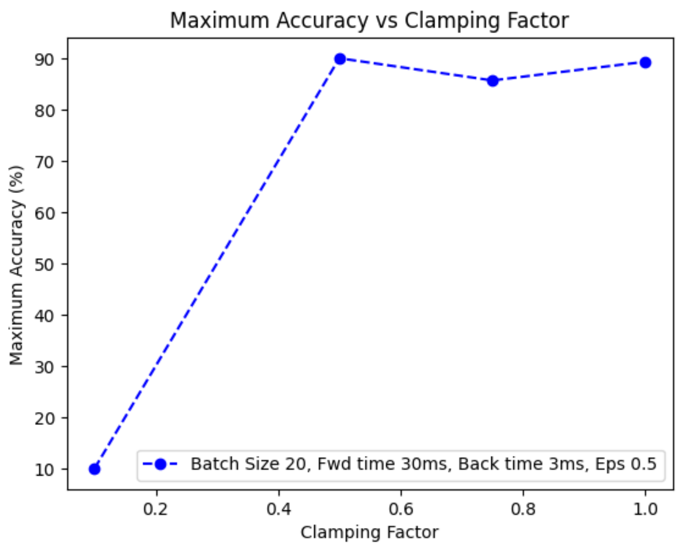

Fig. 9. Maximum Accuracy vs Clamping Factor

F. Energy Analysis

Spike-based neuromorphic computational systems when implemented on silicon neuromorphic technology can significantly reduce the energy consumption of inference and training respectively compared to classic backpropagation algorithms. We run EqProp on a Hopfield Neural Network with each neuron represented by a "spiking" number between 0 to 1. The total energy of the network is as per Equation 1. We plot the energy of our Hopfield network as a function of time and it does decrease continuously as expected(Fig10).

We compare the performance to an equivalent simple ANN using Back-Propagation with a single hidden layer of 1024 neurons and ReLU activation and plot its energy by drawing a parallel with the Hopfield model over time. It clearly shows orders of magnitude of difference in total energy of the network and increasing pattern unlike EqProp establishing the result that Spike-Based EqProp shows a clear advantage over simple BackProp in terms of energy efficiency and biological feasibility(Fig11).

G. An Aside on Energy

While running our experiments we noticed a that the energy of our model went to a local minima of around -0.3 at around an accuracy of 90(Fig12). Beyond this point energy would start increasing and push the model back to an accuracy of 90, which becomes an intrinsic upper limit to the performance of our model. Future work should involve working on making the energy minima of the model better and finding more effective activations and hyperparameters and maybe more number of hidden layers.

VII. Further Discussion

Equilibrium Propagation shares similarities with Contrastive Hebbian Learning and Contrastive Divergence, but it solves the theoretical issues of both algorithms by computing the gradient

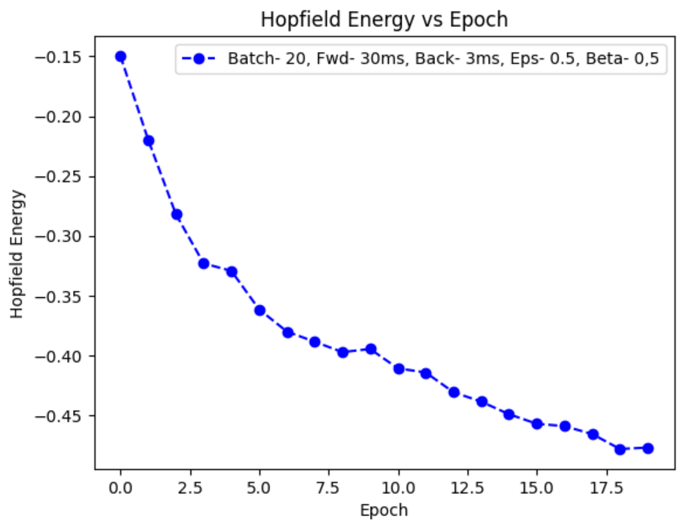

Fig. 10. EqProp - Hopfield Energy as a function of epochs

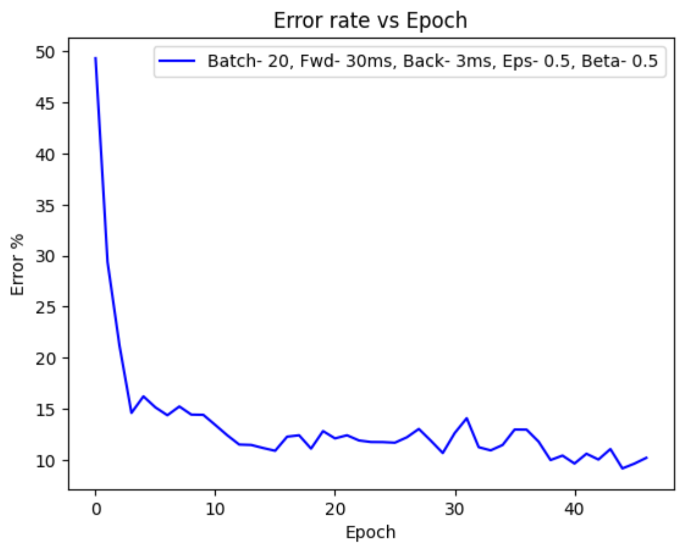

Fig. 12. Error Rate vs Epochs (Best Performance)

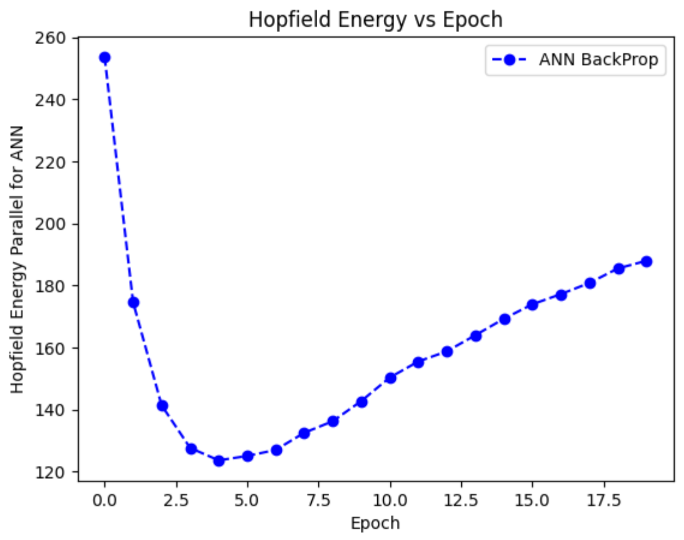

Fig. 11. BackProp - Hopfield Energy as a function of epochs

of a well-defined objective function. The second phase of Equilibrium Propagation corresponds to nudging the prediction towards a configuration that reduces prediction error. In a recurrent multi-layer supervised network, the output units are slightly nudged towards their target, and this perturbation propagates backward in the hidden layers. The signal ”back-propagated” during this phase encodes the gradient of the objective function.
The Hopfield model requires symmetric weights between units, but it is unclear how this symmetry arises from the learning procedure and whether the fact holds for biological models or not. One encouraging evidence is that Autoencoders and denoising autoencoders have shown potential in learning symmetric weights, suggesting a possible solution to the symmetry requirement.

The synaptic update in Equilibrium Propagation corresponds to a standard form of spike-timing-dependent plasticity, making it more plausible that a mechanism similar to Backpropagation could be implemented by brains.
The backpropagation algorithm for feedforward nets also works when feedback weights are random, indicating a tendency for feedforward weights to align with feedback weights. A stack of RBMs can help reduce the negative impact of a lengthy relaxation to a fixed point by making each layer a good autoencoder. The proposed update formula in synaptic plasticity involves the temporal derivative of postsynaptic activity, which differs from traditional Hebbian learning rules. Experimental results show that multi-layer recurrently connected networks with 1, 2, and 3 hidden layers can be trained by Equilibrium Propagation on the permutation-invariant MNIST task.

Future work should consider synaptic cooperativity, depolarization, and update rules based on spike timing statistics. Statistics of triplets or quadruplets of spike timing are being studied currently which will broaden synaptic cooperativity and might change our understanding of SNNs. Backpropagation through time is less plausible for time-varying inputs and since the brain is a network receiving time-varying inputs, computationally efficient estimators of the gradient can be obtained using a forward method i.e online noisy estimation of the gradient can eliminate the need to store the past states in the training sequence, making the time-varying inputs more feasible.

References

[1] Martin, Erwann, Maxence Ernoult, Jérémie Laydevant, Shuai-shuai Li, Damien Querlioz, Teodora Petrisor and Julie Grollier. “EqSpike: spike-driven equilibrium propagation for neuromorphic implementations.” iScience 24 (2020): n. pag.
[2] Scellier B, Bengio Y. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. Front Comput Neurosci. 2017 May 4;11:24. doi: 10.3389/fncom.2017.00024. PMID: 28522969; PMCID: PMC5415673.