Deep Learning Project

Topic: Feedback Alignment Methods

Vraj Thakkar(202103052)
Pranav Patel(202103040)
Jainil Patel(202101416)

Artificial Neural Network -

Neurons in the Brain:

- The brain consists of billions of neurons that communicate through synapses.
- Neurons process information and transmit signals in the brain's neural networks.

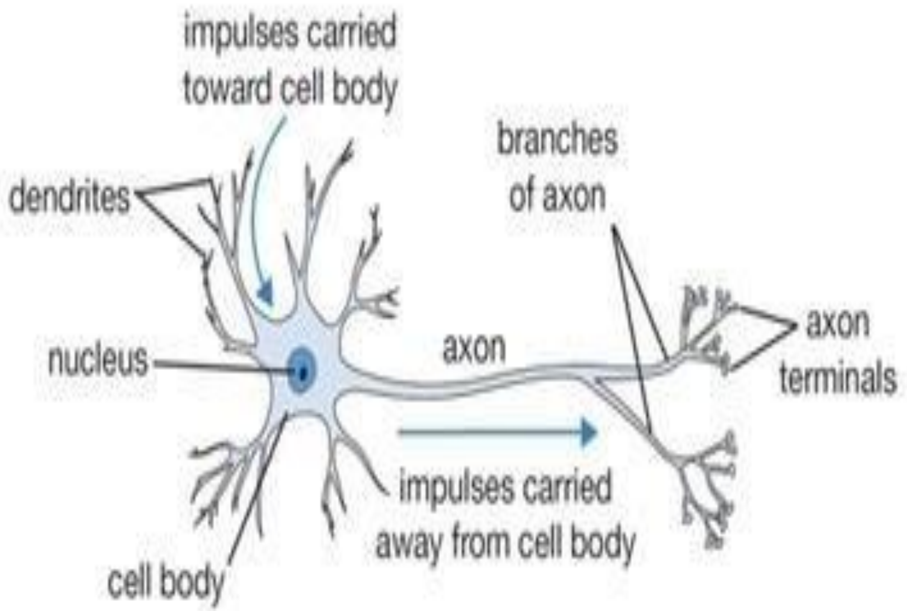
Artificial Neurons:

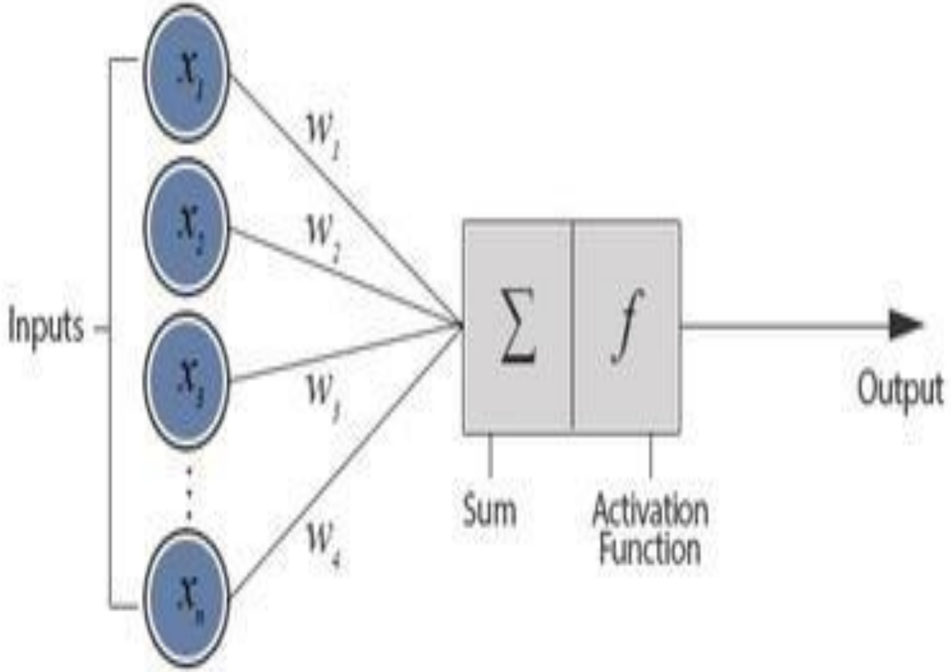
- In artificial neural networks (ANNs), artificial neurons mimic the behavior of biological neurons.
- Artificial neurons process inputs, apply weights to them, and produce an output signal.

Neuron architecture in Brain\ANN

- ➔ *The brain's neurons are organized into layers, including input, hidden, and output layers .*
- Neurons in the brain are connected to each other through synapses.
- In ANNs, artificial neurons are connected through weighted connections, similar to synapses.

Biological Neuron versus Artificial Neural Network


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body



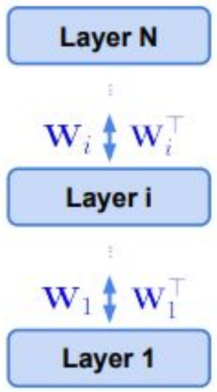
The need for backpropagation

- Key algorithm used in training ANNs. It is used to adjust the weights of connections between neurons in the network in order to minimize the difference between the actual output of the network and the desired output, typically measured by a loss function.
- To compute the gradients for the preceding layer, the gradients from the current layer are multiplied by the transpose of the weight matrix.

Math Behind it

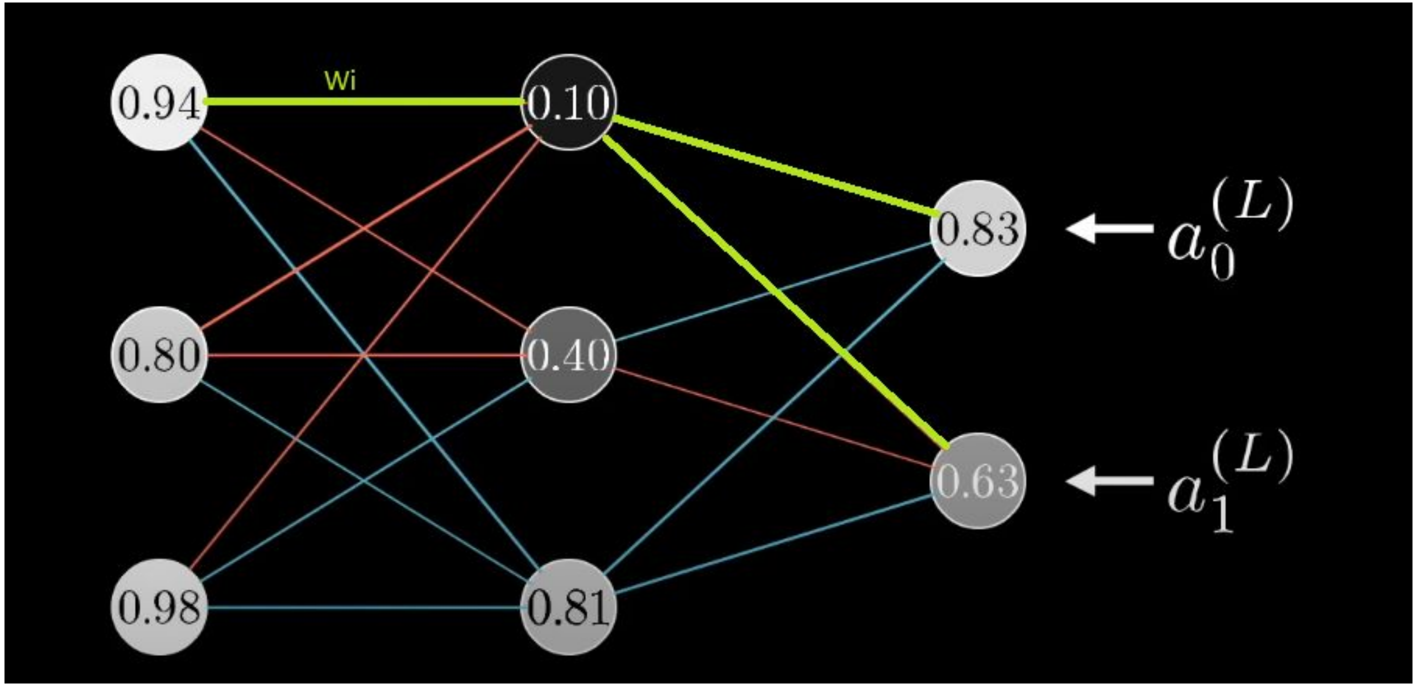
$$\delta \mathbf{W}_i = -\left(\left(\mathbf{W}_{i+1}^\top \delta \mathbf{z}_{i+1}\right) \odot \phi'(\mathbf{z}_i)\right) \mathbf{y}_{i-1}^\top, \text{ with } \delta \mathbf{z}_{i+1} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_{i+1}}$$

$$\mathbf{y}_i = \phi(\mathbf{z}_i), \text{ with } \mathbf{z}_i = \mathbf{W}_i \, \mathbf{y}_{i-1} + \mathbf{b}_i.$$


Layer N
$\mathbf{W}_i$ $\mathbf{W}_i^\top$
Layer i
$\mathbf{W}_1$ $\mathbf{W}_1^\top$
Layer 1

(a) BP

Crucial Step



Why Backpropagation is an issue?

1. Such a system requires backward pass to update the weights in each iteration. In brain such a system cannot take place as neurons in the brain are unidirectional.
2. One of the main drawbacks of this method is that it does not promote parallelization as the method is sequential.
3. Symmetric weight updation.
4. Vanishing or exploding gradient issues are always concerning.
5. Model can overfit to the data, and can therefore be prone to adversarial attacks.
6. Computationally this method is very expensive and there is need for faster methods for weight updation.

Biologically plausible algorithms

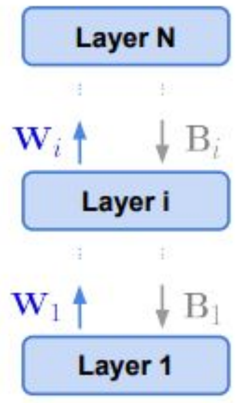
1. We are not sure if there’s a symmetry in the neuron connections and also the neurons are unidirectional.
2. Due to aforementioned issues backpropagation is certainly not the way we learn things.
3. We need more algorithms which are robust and biologically plausible.
4. Computationally also we need algorithms that are faster, avoid so many calculations at each layer.
5. We need algorithms that are not sequential and we can successfully parallelize.

Feedback alignment

1. While calculating the gradient at each step, the memory of previous layers is not used.
2. Weight memory from previous steps are replaced by a random matrix multiplication which is used to update the weights at each layer.
3. Increased randomness in data makes the model more protective against adversarial attacks.
4. Computationally efficient and can be parallelized.
5. Eliminate the need of symmetricity in updation in the network.

How the Math changes?

$$\delta \mathbf{W}_i = -\left((\mathbf{B}_{i+1}\, \delta \mathbf{z}_{i+1}) \odot \phi'(\mathbf{z}_i)\right) \mathbf{y}_{i-1}^{\top}, \text{ with } \delta \mathbf{z}_{i+1} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_{i+1}}$$



(b) FA

Drawbacks of FA

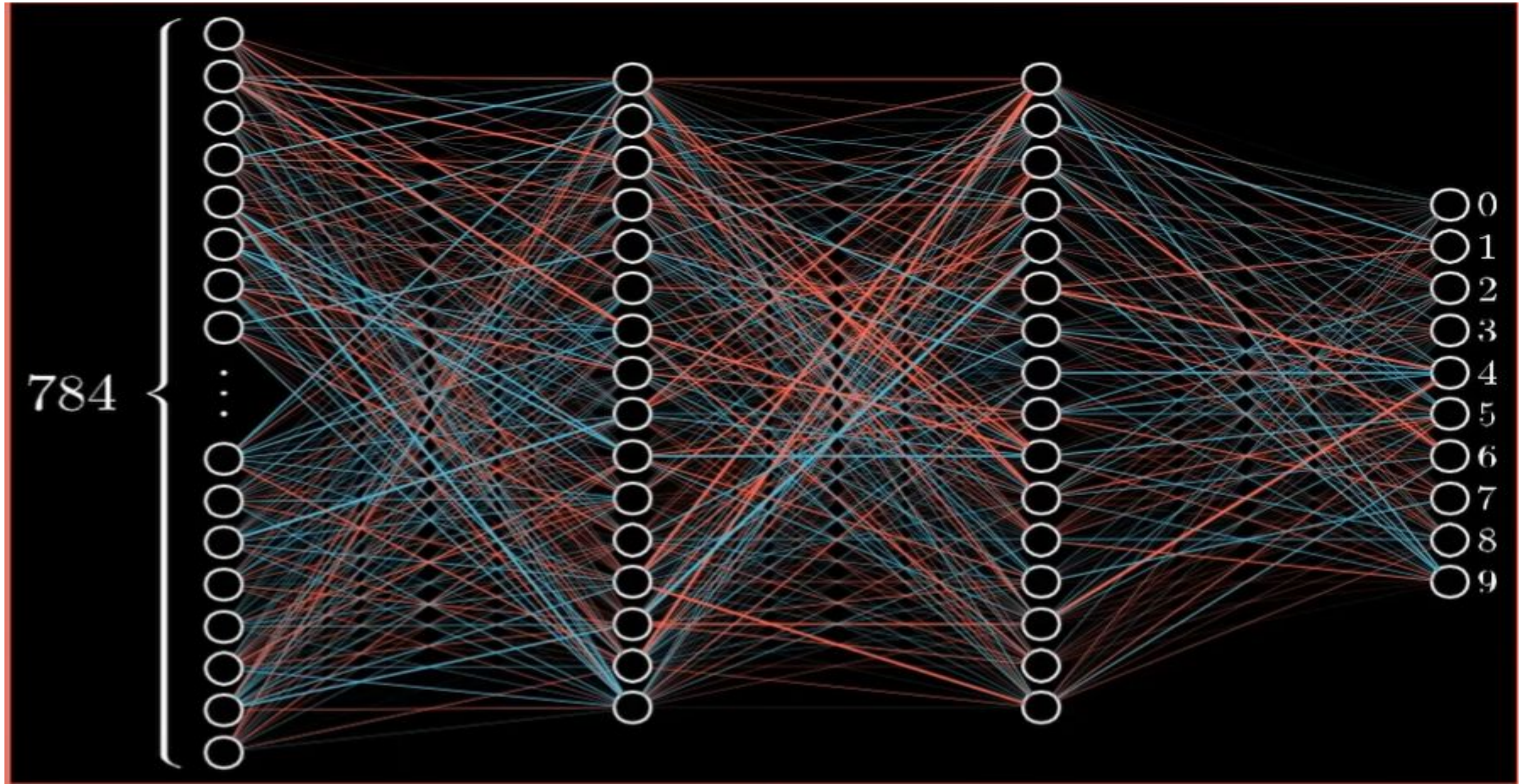
1. Needs excessive amount of data due to the inherent randomness in the network.

2. Does not converge in most cases due to lack of direct error information encoding in the network.

3. Works best on shallow networks.

4. Extremely sensitive to initialization of random matrices and weights

What happens inside?

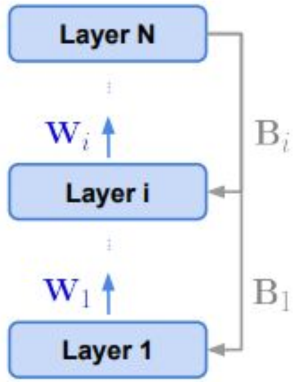


Direct Feedback alignment

1. Improvement to the architecture of FA, as the weight updates are dependent on loss in the final layer.
2. Higher level intuition is that we require the knowledge of only the direction of weight change and not the magnitude exactly.
3. Model converges way better than normal FA.
4. Controlled behaviour decreases the exploding, vanishing gradient problem

How the Math changes?

$$\delta \mathbf{W}_i = -\left(\left(\mathbf{B}_i\, \delta \mathbf{z}_N\right) \odot \phi'(\mathbf{z}_i)\right) \mathbf{y}_{i-1}^{\top}, \text{ with } \delta \mathbf{z}_N = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_N}.$$



(c) DFA

Drawbacks of DFA

1. We don’t know if it’s really biologically plausible.
2. Slower convergence in deep neural networks due to increased randomization
3. Gradients are not captured effectively as precise error is not back propagated in each iteration.

Biotorch Library

- BioTorch is an open-source python library which allows you to train biologically plausible learning algorithms.
- It uses PyTorch in the background.
- BioTorch holds implementation of all the feedback alignment methods (FA, DFA..)

https://github.com/jsalbert/biotorch

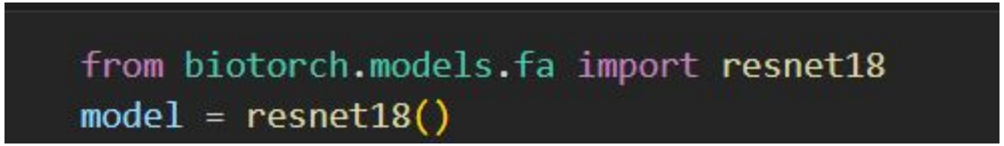
BioTorch Interface

BioTorch’s intuitive interface, allows the user to create neural network models with feedback alignment methods in an effortless manner.

Functionalities :

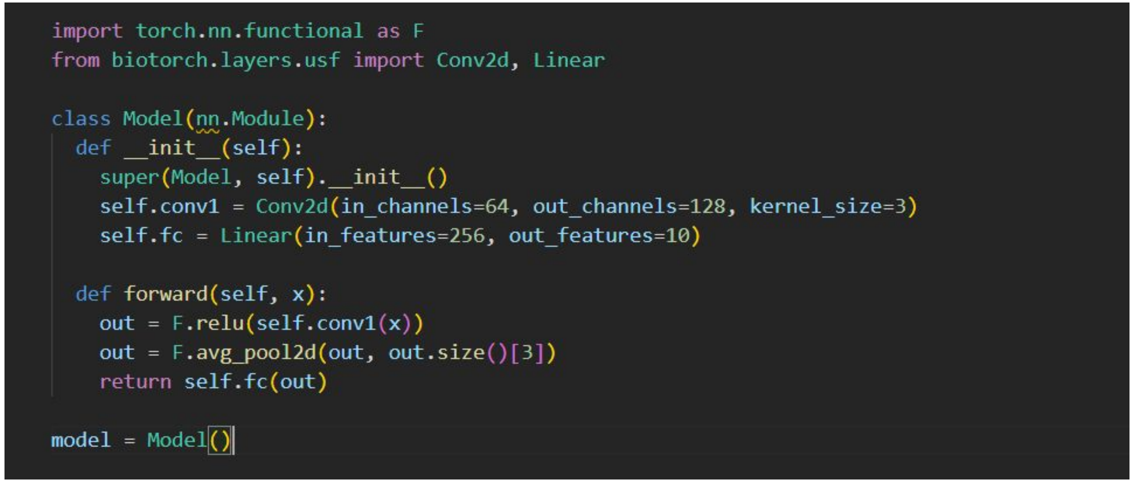
1. Model importing : predefined 25 state-of-the-art architectures, such as (LeNet, ResNet-50, DenseNet-161)
2. Custom model creation using alignment layers
3. Automatic conversion of an existing model architecture

1. Model importing :



```python
from biotorch.models.fa import resnet18
model = resnet18()
```

2. Custom model creation using alignment layers



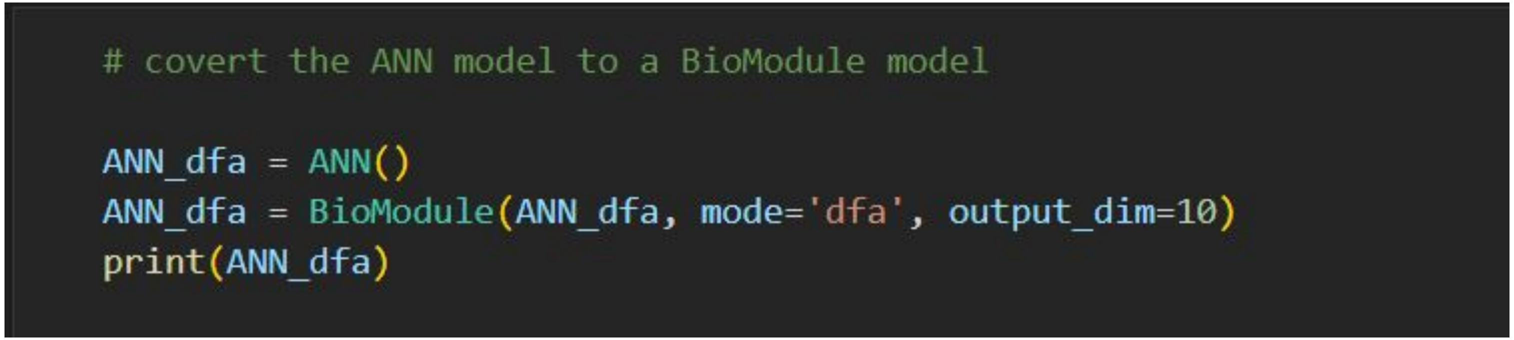
```python
import torch.nn.functional as F
from biotorch.layers.usf import Conv2d, Linear

class Model(nn.Module):
  def __init__(self):
    super(Model, self).__init__()
    self.conv1 = Conv2d(in_channels=64, out_channels=128, kernel_size=3)
    self.fc = Linear(in_features=256, out_features=10)

  def forward(self, x):
    out = F.relu(self.conv1(x))
    out = F.avg_pool2d(out, out.size()[3])
    return self.fc(out)

model = Model()
```

3. : Automatic conversion of an existing model architecture



```
# covert the ANN model to a BioModule model

ANN_dfa = ANN()
ANN_dfa = BioModule(ANN_dfa, mode='dfa', output_dim=10)
print(ANN_dfa)
```

Results on ANN

- ANN with 1-hidden layer with 50 neurons
- Dataset : MNIST

Methods	Test Accuracy	Loss
BP	93.49 %	0.2523
FA	94.12 %	0.9604
DFA	95.0 %	0.7926

Results on CNN

CNN with 3 convolutional layers and 2 linear layers

Dataset : MNIST

Methods	Test Accuracy	Loss
BP	97.0 %	0.084
FA	90.34 %	157.07
DFA	92.87 %	31.87

Implementation File:

CNN_bp

CNN

ANN

CNN_fa

LeNetMNIST