Forward-Forward :
Is it time to bid adieu to BackProp?

Sai Anuroop Kesanapalli
Shashank Rangarajan
Anukaran Jain
Avtaran Jain

“The Gradient Gangsters”

Goal

- To implement and study the Forward-Forward (FF) algorithm, and compare it with the traditional back-propagation (BackProp) framework

- Study the architectural differences of FF and BackProp and explore new architectures

- Analyze system performance of FF and BackProp

FF

- Every layer has its own objective (goodness) function
- Layers are trained separately
- Gradients are computed
- No flow of gradients across layers

BackProp

- There is a global objective (loss) function
- Layers are trained jointly
- Gradients are computed
- Flow of gradients across layers

Exp 1: Comparison of Baseline FF with BackProp

FF

```
layers = [
    Linear_layer(28*28, 2000, device=device),
    Linear_layer(2000, 2000, device=device),
    Linear_layer(2000, 2000, device=device),
    Linear_layer(2000, 2000, device=device)
  ]
net = Net(layers, 10)
```

BackProp

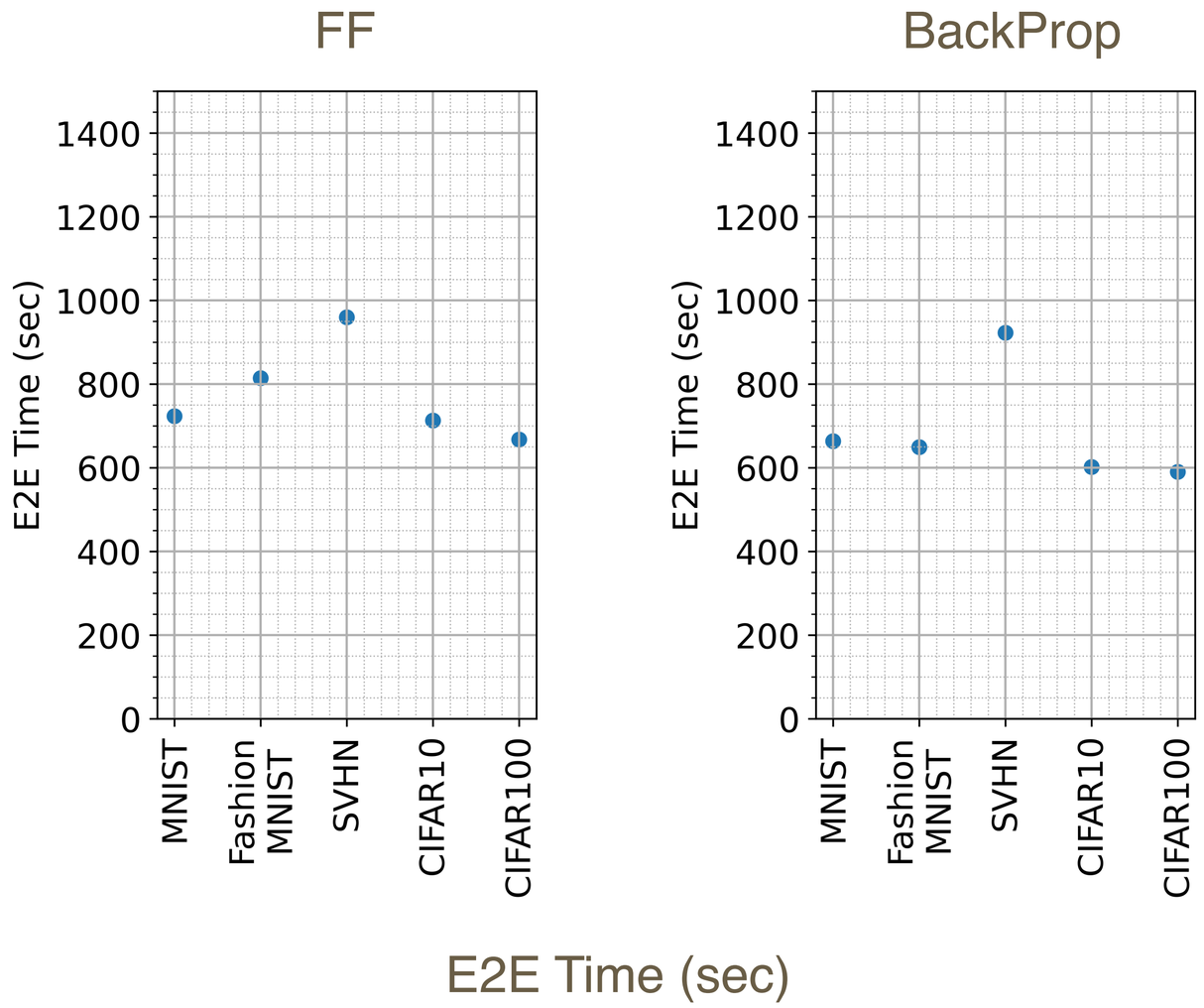
```
Net(
  (linear1): Linear(in_features=784, out_features=250, bias=True)
  (relu1): ReLU()
  (linear2): Linear(in_features=250, out_features=110, bias=True)
  (relu2): ReLU()
  (linear3): Linear(in_features=110, out_features=10, bias=True)
)
```

Exp 1: Comparison of Baseline FF with BackProp

Algo\Data	MNIST	Fashion MNIST	SVHN	CIFAR 10	CIFAR 100
FF	97.14	85.47	64.93	46.13	10.33
BackProp	98.04	89.43	80.52	53.88	23.75

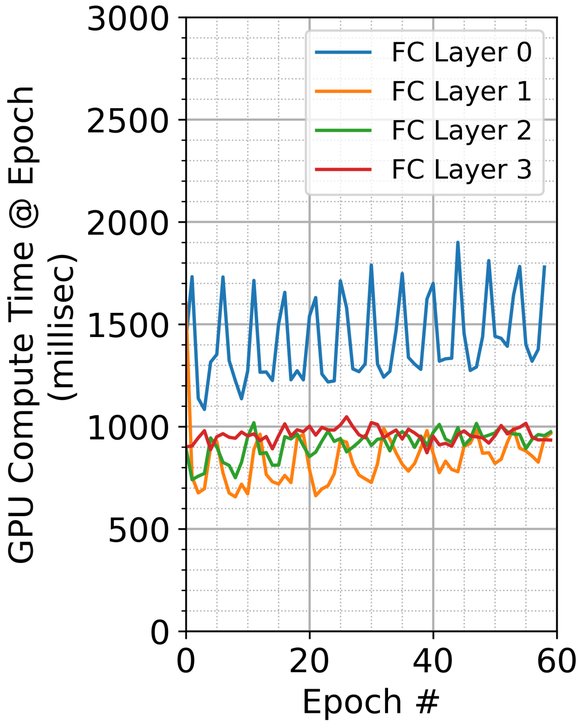
Test Accuracy (%)

Exp 1: Comparison of Baseline FF with BackProp

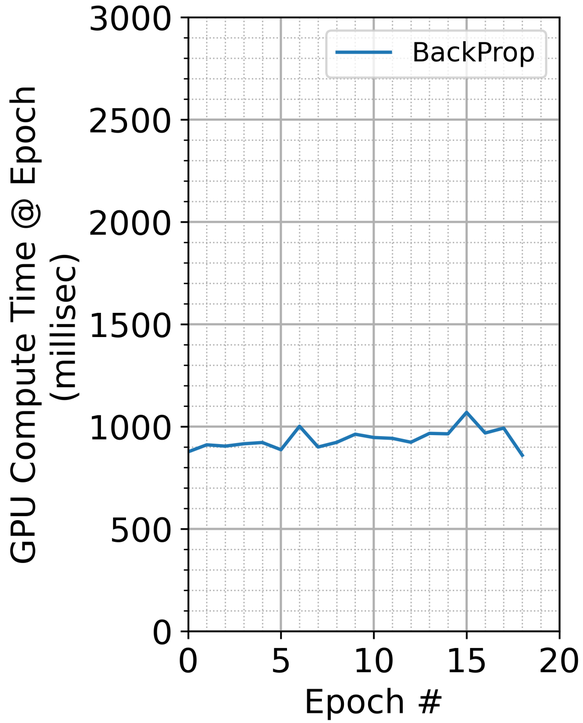


Exp 1: Comparison of Baseline FF with BackProp

FF

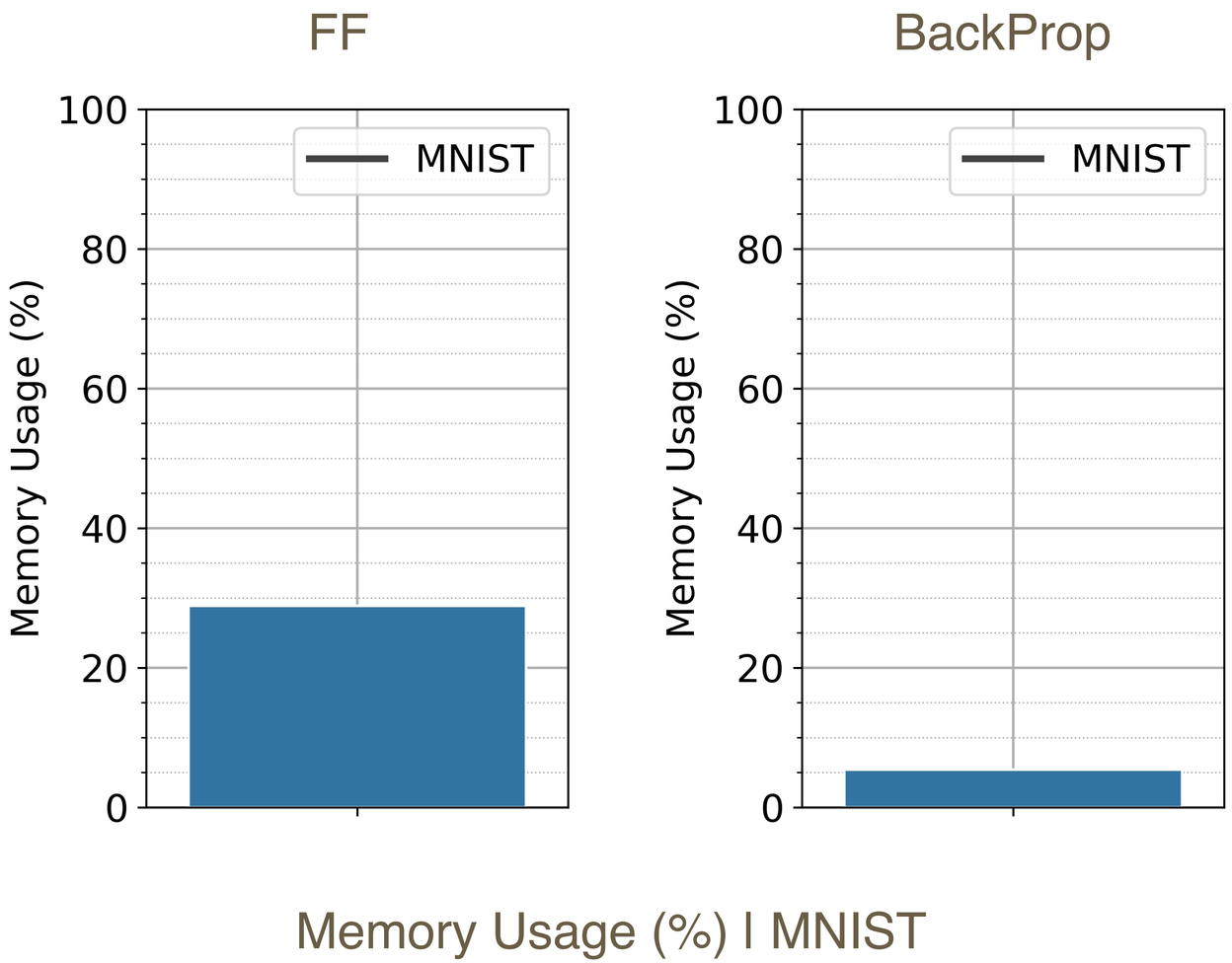


BackProp

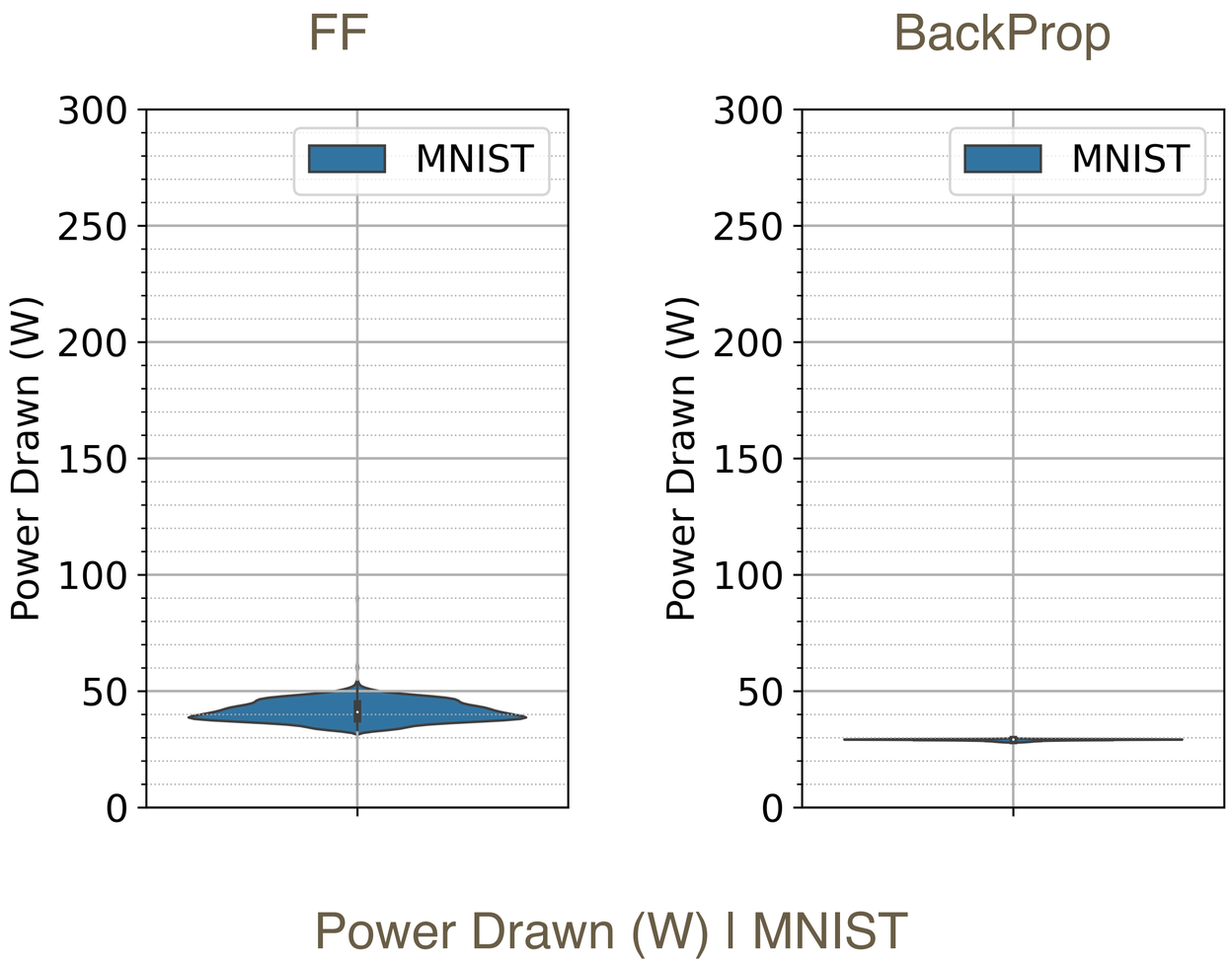


GPU Compute Time | CIFAR100

Exp 1: Comparison of Baseline FF with BackProp



Exp 1: Comparison of Baseline FF with BackProp



Exp 2: Hybrid FF + BackProp

Dataset: MNIST

- **Exp 2 a):** Testing BackProp performance with FF as weight initializer
  - ➡ Method: Train the network using FF for 60 epochs followed by BackProp for 20 epochs with reduced learning rate 1e-4
- **Exp 2 b):** Testing FF performance with BackProp as weight initializer
  - ➡ Method: Train the network using BackProp for 20 epochs followed by FF for 60 epochs with reduced learning rate 1e-4

Observation: Abysmal performance (< 10% test accuracy) for both experiments!

Conclusion: FF and BackProp have different objective functions that do not gel well together

Exp 3: CNN with FF

```
layers = [
    Convolutional_layer((28, 28, 1), 6, kernel_size = 5, padding=2),
    MaxPool_layer(kernel_size=2, stride=2),
    Flatten_layer(),
    Convolutional_layer((14,14,6),16, kernel_size=5, padding =0),
    MaxPool_layer(kernel_size=2, stride=2),
    Flatten_layer(),
    Linear_layer(400, 2000, device=device),
    Linear_layer(2000, 2000, device=device)
]
net = Net(layers, 10)
```

Architecture

Exp 3: CNN with FF

Dataset: MNIST

Method: Train the network using FF for 1000 epochs with learning rate 2e-2

Observation: Poor performance (10.75% test accuracy)

Conclusion: Our observation bolsters the view that CNN is not feasible with FF due to weight sharing

Exp 4: FF with Self-Attention

```
layers = [
    Attention_layer((28, 28, 1), 2000, 4),
    Linear_layer(2000, 2000, device=device),
    Linear_layer(2000, 2000, device=device),
    Linear_layer(2000, 2000, device=device)
  ]

net = Net(layers, 10)
```

Architecture

Exp 4: FF with Self-Attention

Method: Train the network using FF for 60 epochs with learning rate 1e-3 and 2e-2 for Attention and Linear layers respectively

Observation: Did not observe performance comparable to baseline FF (60%)

Conclusion: Label overlay method used in FF does not work well with networks emphasizing spatial locality

Next Steps

- Explore sample complexity for FF vs BackProp
- Try larger datasets such as GLD23k

Future Work

- Layer-wise parallelized implementation of FF
- Passing overlay information to Attention layer so that embeddings are well-formed